

КОГДА ЗНАЕШЬ — ВСЕ ПРОСТО



Планета
Excel



Николай Павлов

Скульптор данных в Excel с Power Query

Как загружать в Excel любые данные
и приводить их в порядок

Николай Павлов

Скульптор данных в Excel с Power Query

Как загружать в Excel любые данные
и приводить их в порядок



DELIBRI

Москва
2019

УДК 004,42(075.8)
ББК 32.81-018,2*32.973.26
П 12

Павлов, Николай

П 12 Скульптор данных в Excel с Power Query / Николай Павлов. – М. : Де`Либли, 2019. – 332 с. : ил.

ISBN 978-5-519-50143-9

Это первая книга на русском языке, посвящённая настройке Power Query – мощному инструменту для работы с данными в Microsoft Excel. С её помощью можно легко решать множество задач, для которых раньше требовались сложные формулы или макросы. Подробно разбираются вопросы импорта данных в Excel из внешних источников (файлов разных форматов, баз данных, интернета и т. д.) и трансформации полученных таблиц с последующим их анализом.

Книга рассчитана на средних и продвинутых пользователей. Ко всем описанным в книге задачам в комплекте идут живые файлы-примеры, которые можно использовать в работе.

УДК 004,42(075.8)
ББК 32.81-018,2*32.973.26

ISBN 978-5-519-50143-9

© Николай Павлов, 2019
© Де`Либли, издание, 2019

https://t.me/it_boooks

Оглавление

ВВЕДЕНИЕ	8
Для КОГО ЭТА КНИГА	9
Новички в теме	9
Интересующиеся	9
Профи	9
ФАЙЛЫ С ПРИМЕРАМИ ИЗ ЭТОЙ КНИГИ	10
ТРЕНИНГИ	10
ВИДЕОУРОКИ И КАНАЛ НА YOUTUBE	11
НАЧАЛО РАБОТЫ С POWER QUERY	12
ТИХАЯ РЕВОЛЮЦИЯ	13
ЧТО МОЖЕТ И ЧЕГО НЕ МОЖЕТ POWER QUERY	15
ОТКУДА МОЖНО ЗАГРУЖАТЬ ДАННЫЕ В POWER QUERY	16
Базы данных	16
Интернет и облачные хранилища	16
Корпоративные программы	16
Файлы и папки	17
Другие источники	17
ВЕРСИИ И ОБНОВЛЕНИЯ	18
Отдельная надстройка для Excel 2010–2013	18
Встроенный функционал в Excel 2016–2019	18
Часть Microsoft Power BI Desktop	19
ОСНОВНЫЕ ПРИНЦИПЫ РАБОТЫ В POWER QUERY НА ПРИМЕРЕ ЗАГРУЗКИ ТХТ-ФАЙЛА	22
Постановка задачи	22
Загружаем файл	23
Окно редактора запросов	24
Наводим порядок в данных	25
Выгрузка результатов из Power Query обратно в Excel	30
ПОСТРОЕНИЕ СВОДНОЙ ТАБЛИЦЫ ПО РЕЗУЛЬТАТАМ ЗАПРОСА	33
ОБНОВЛЕНИЕ ЗАПРОСОВ	35
ИСХОДНЫЙ КОД ЗАПРОСА НА ЯЗЫКЕ M	37
ЗАГРУЗКА ДАННЫХ В POWER QUERY	38
ЗАГРУЗКА ДАННЫХ ИЗ ВНЕШНЕЙ КНИГИ EXCEL	39
ЗАГРУЗКА ДАННЫХ ИЗ ТЕКУЩЕЙ КНИГИ EXCEL	42
«Умная» таблица	42
Именованный диапазон	43
Универсальный способ с функцией Excel.CurrentWorkbook	43
ПОДКЛЮЧЕНИЕ К БАЗАМ ДАННЫХ	45
ЗАГРУЗКА ДАННЫХ ИЗ ИНТЕРНЕТА	47
Импорт данных с веб-страниц	47
Прямая загрузка Excel-файлов с веб-страниц	50
Загрузка данных из Facebook	50
ЗАГРУЗКА ИНФОРМАЦИИ ЧЕРЕЗ OPEN DATA PROTOCOL (ODATA)	52
ЗАГРУЗКА ДАННЫХ ИЗ ФАЙЛОВ XML	54
ЗАГРУЗКА И ВИЗУАЛИЗАЦИЯ ГЕОДАННЫХ ИЗ ФАЙЛОВ JSON	57
ЗАГРУЗКА ДАННЫХ ИЗ PDF ЧЕРЕЗ WORD	61
Шаг 1. Открываем PDF в Word	62
Шаг 2. Сохраняем документ как веб-страницу	63
Шаг 3. Загружаем файл в Excel через Power Query	64
ЗАГРУЗКА ДАННЫХ ПОЧТЫ И КАЛЕНДАРЯ ИЗ MICROSOFT EXCHANGE	67

СЛИЯНИЕ ЗАПРОСОВ	71
Типы слияния в POWER QUERY.....	72
<i>Добавление (Append)</i>	<i>72</i>
<i>Объединение (Merge).....</i>	<i>72</i>
ДОБАВЛЕНИЕ ДВУХ ТАБЛИЦ.....	74
ДОБАВЛЕНИЕ ТРЕХ И БОЛЕЕ ТАБЛИЦ С ЗАГРУЗКОЙ В МОДЕЛЬ ДАННЫХ.....	78
ОБЪЕДИНЕНИЕ ТАБЛИЦ: ЗАБУДЬТЕ ПРО ВПР.....	80
<i>Загружаем все таблицы как подключения.....</i>	<i>80</i>
<i>Выполняем слияние</i>	<i>80</i>
<i>Исправляем ошибки.....</i>	<i>82</i>
<i>Объединение в этом же запросе</i>	<i>83</i>
ОБЪЕДИНЕНИЕ ПО НЕСКОЛЬКИМ СТОЛБЦАМ.....	86
ПОДСТАНОВКА СРАЗУ ВСЕХ НАЙДЕННЫХ ЗНАЧЕНИЙ.....	88
ИНТЕРВАЛЬНЫЙ ВПР	91
СРАВНЕНИЕ ТАБЛИЦ ОБЪЕДИНЕНИЕМ РАЗНЫХ ТИПОВ.....	95
СРАВНЕНИЕ ТАБЛИЦ С ПОМОЩЬЮ УСЛОВНОГО СТОЛБЦА	99
НАСТРОЙКА УРОВНЕЙ КОНФИДЕНЦИАЛЬНОСТИ ИСТОЧНИКОВ ДАННЫХ	103
<i>Зачем нужны уровни конфиденциальности</i>	<i>103</i>
<i>Настройка уровней и ошибка Formula.Firewall.....</i>	<i>104</i>
<i>Проверить или нет?</i>	<i>105</i>
МАССОВАЯ ЗАГРУЗКА ДАННЫХ	106
ИМПОРТ ВСЕХ ТЕКСТОВЫХ ФАЙЛОВ ИЗ ПАПКИ	107
<i>Постановка задачи.....</i>	<i>107</i>
<i>Отбираем нужные файлы.....</i>	<i>108</i>
<i>Разворачиваем содержимое файлов</i>	<i>110</i>
<i>Выгружаем в Excel и ловим ошибки</i>	<i>113</i>
СБОР ДАННЫХ ИЗ ВСЕХ EXCEL-ФАЙЛОВ ЗАДАННОЙ ПАПКИ.....	115
<i>Постановка задачи.....</i>	<i>115</i>
<i>Формируем список файлов.....</i>	<i>116</i>
<i>Извлекаем содержимое каждого файла</i>	<i>117</i>
<i>Отбираем нужные листы</i>	<i>119</i>
<i>Разворачиваем таблицы и «причёсываем» результаты</i>	<i>119</i>
<i>Дополнительные улучшения для сводной</i>	<i>121</i>
ИМПОРТ ВСЕХ «УМНЫХ» ТАБЛИЦ ИЗ ТЕКУЩЕЙ КНИГИ.....	123
<i>Постановка задачи.....</i>	<i>123</i>
<i>Формируем список таблиц.....</i>	<i>123</i>
<i>Разворачиваем таблицы</i>	<i>125</i>
<i>Исключаем рекурсию.....</i>	<i>126</i>
ЗАГРУЗКА ВСЕХ ПРОСТЫХ ТАБЛИЦ С ЛИСТОВ ТЕКУЩЕЙ КНИГИ.....	128
ПРЕОБРАЗОВАНИЯ ТАБЛИЦ	131
ФИЛЬТРАЦИЯ	132
<i>Фильтрация разных типов данных.....</i>	<i>132</i>
<i>Опасная иллюзия с фильтрацией через поле «Поиск».....</i>	<i>133</i>
ТРАНСПОНИРОВАНИЕ.....	135
ЗАПОЛНЕНИЕ ПУСТЫХ ЯЧЕЕК	137
ГРУППИРОВКА СТРОК	139
<i>Простая группировка.....</i>	<i>139</i>
<i>Сложная группировка.....</i>	<i>140</i>
<i>Подсчет количества уникальных значений</i>	<i>141</i>
<i>Группировка с выводом всех значений</i>	<i>142</i>
<i>Извлечение уникальных значений при группировке.....</i>	<i>145</i>
<i>Первый/последний элемент в каждой группе</i>	<i>146</i>
СВЕРТЫВАНИЕ ТАБЛИЦ.....	150

<i>Простое свёртывание</i>	150
<i>Имитация сводной с текстом в значениях</i>	152
ТРАНСФОРМАЦИЯ СТОЛБЦА В ДВУМЕРНУЮ ТАБЛИЦУ	156
<i>Постоянный шаг в данных</i>	156
<i>Переменный шаг в данных</i>	159
ОТМЕНА СВЁРТЫВАНИЯ	162
<i>Зачем нужна отмена свёртывания</i>	162
<i>Отмена свёртывания простой таблицы</i>	163
<i>Отмена свёртывания таблицы с многоуровневыми подписями</i>	164
<i>Отмена свёртывания сразу нескольких таблиц</i>	167
ПОДТЯГИВАНИЕ ЗНАЧЕНИЙ К КРАЮ ТАБЛИЦЫ	172
ОПЕРАЦИИ С ТЕКСТОМ	173
ВАЖНОЕ ЗАМЕЧАНИЕ	174
ИЗМЕНЕНИЕ РЕГИСТРА	175
УДАЛЕНИЕ ЛИШНИХ ПРОБЕЛОВ И SUPERTRIM	177
ОЧИСТКА ТЕКСТА ОТ НЕПЕЧАТАЕМЫХ СИМВОЛОВ	179
РАЗДЕЛЕНИЕ «СЛИПШЕГОСЯ» ТЕКСТА	180
<i>Простой случай</i>	180
<i>Деление на строки вместо столбцов</i>	181
<i>Несколько строк в одной ячейке</i>	183
РАЗБОР БУКВЕННО-ЦИФРОВОЙ КАШИ	185
СКЛЕИВАНИЕ ТЕКСТА	188
<i>Команда «Объединить столбцы»</i>	188
<i>Склейка формулой</i>	188
<i>Склеивание текста и чисел</i>	190
<i>Склеивание текста и дат</i>	191
<i>Массовая склейка функцией Text.Combine</i>	191
СТОЛБЕЦ ИЗ ПРИМЕРОВ	193
ГЕНЕРАТОР ФРАЗ ДЕКАРТОВЫМ ПРОИЗВЕДЕНИЕМ	197
НЕЧЁТКИЙ ТЕКСТОВЫЙ ПОИСК	200
<i>Шаг 1. Создаем функцию коэффициента подобия</i>	200
<i>Шаг 2. Выполняем декартово произведение списков</i>	202
<i>Шаг 3. Ищем самые похожие пары</i>	204
ОБРАБОТКА ДАТ И ВРЕМЕНИ	206
РАСПОЗНАВАНИЕ ДАТ	207
<i>Формат даты для столбца</i>	207
<i>Использование локали для дат других стран</i>	208
<i>Столбцы с датами смешанного формата</i>	209
ПРЕОБРАЗОВАНИЕ ДАТ	211
НОМЕР НЕДЕЛИ ПО ISO	214
КОНВЕРТИРОВАНИЕ ДАТЫ В ТЕКСТ	216
ВЫЧИСЛЕНИЕ ДЛИТЕЛЬНОСТЕЙ	217
<i>Разница в полных днях</i>	217
<i>Продолжительность как тип данных</i>	217
<i>Вычисление возраста</i>	218
СДВИГ ДАТЫ НА N ПЕРИОДОВ	219
ПОИСК САМОЙ РАННЕЙ И САМОЙ ПОЗДНЕЙ ДАТЫ	220
<i>Во всем столбце</i>	220
<i>По каждой группе значений</i>	220
ЗАПОЛНЕНИЕ ПРОБЕЛОВ В ДАТАХ	222
РАБОТА С ЗАПРОСАМИ	226
ГРУППИРОВКА ЗАПРОСОВ	227
ЗАЩИТА ЗАПРОСОВ	228

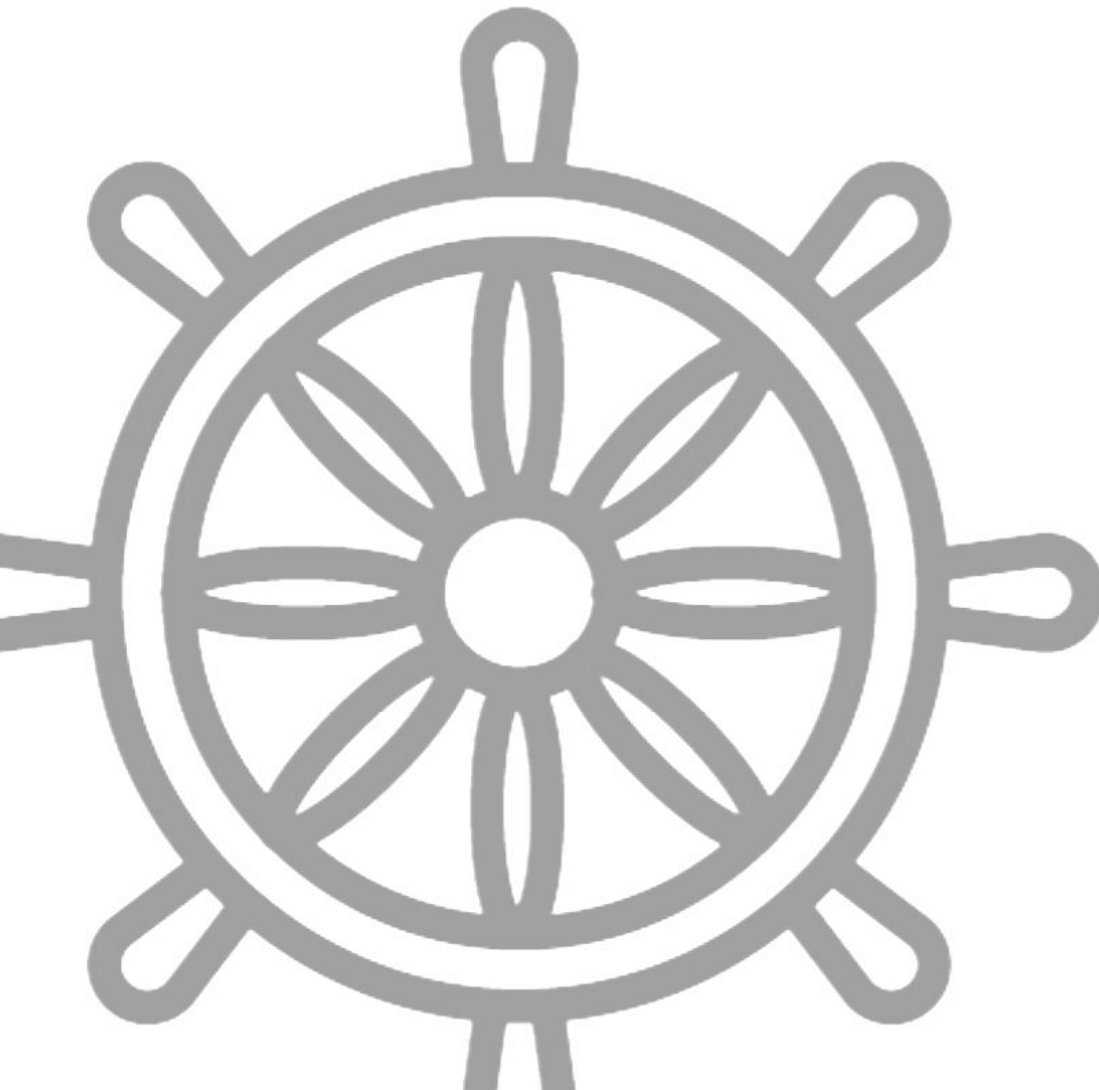
ПРОСМОТР ЗАВИСИМОСТЕЙ МЕЖДУ ЗАПРОСАМИ	229
КОПИРОВАНИЕ, ДУБЛИРОВАНИЕ И ССЫЛКА НА ЗАПРОС.....	230
<i>Дублирование</i>	230
<i>Ссылка</i>	231
<i>Копирование</i>	233
ДЕЛИМСЯ ЗАПРОСАМИ С ВНЕШНИМ МИРОМ	234
<i>Способ 1. Пересылка файла</i>	234
<i>Способ 2. Копирование и вставка запроса в другой файл</i>	234
<i>Способ 3. Экспорт файла подключения</i>	235
ОБНОВЛЕНИЕ ЗАПРОСОВ ПО РАСПИСАНИЮ	236
<i>Запускаем Планировщик</i>	236
<i>Создаем задачу</i>	237
<i>Добавляем макрос на открытие файла</i>	240
<i>Отключаем защиту</i>	241
POWER QUERY И VBA.....	242
<i>Удаление запросов макросом</i>	242
<i>Обновление запросов макросом</i>	242
<i>Создание запроса макросом</i>	243
<i>Загрузка «умных» таблиц в Power Query макросом</i>	244
<i>Загрузка запросов Power Query в Модель Данных Power Pivot макросом</i>	246
ЯЗЫК M	247
ОСНОВЫ СИНТАКСИСА ЯЗЫКА M	248
<i>Выражения</i>	248
<i>Оператор let</i>	249
<i>Комментарии</i>	251
<i>Последовательность выполнения</i>	251
ЛОГИЧЕСКИЕ ВЕТВЛЕНИЯ С IF ... THEN ... ELSE	253
ПРОСТЫЕ ТИПЫ ДАННЫХ	254
<i>Числовой (number)</i>	254
<i>Текстовый (text)</i>	255
<i>Продолжительность (duration)</i>	255
<i>Дата (date)</i>	256
<i>Время (time)</i>	257
<i>Логический тип (logical)</i>	258
<i>Тип null</i>	258
СТРУКТУРИРОВАННЫЕ ТИПЫ ДАННЫХ	260
<i>Список (list)</i>	260
<i>Запись (record)</i>	262
<i>Таблица (table)</i>	264
СПРАВКА ПО ВСТРОЕННЫМ ФУНКЦИЯМ	267
РЕДАКТОР M-КОДА NOTEPAD++ С ПОДСВЕТКОЙ СИНТАКСИСА	268
ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ	270
<i>Создание простой функции</i>	270
<i>Вызов пользовательской функции</i>	270
<i>Типы данных для аргументов и результата</i>	272
<i>Необязательные аргументы</i>	273
<i>Функция внутри запроса</i>	273
<i>Функция как аргумент для другой функции</i>	273
<i>Рекурсия</i>	275
КЛЮЧЕВОЕ СЛОВО EACH	276
<i>Пример 1. Арифметические операции с элементами списка</i>	276
<i>Пример 2. Обработка текстового списка</i>	276
<i>Пример 3. Фильтрация строк в таблице</i>	277
<i>Пример 4. Сложные фильтры</i>	278
ОБРАБОТКА ОШИБОК В ЗАПРОСАХ	279

ТОНКОСТИ ДЕЛЕНИЯ	281
<i>Ошибки (Error)</i>	281
<i>Пусто (Null)</i>	281
<i>Бесконечности (Infinity)</i>	281
<i>Нечисло (NaN)</i>	282
<i>Универсальный подход</i>	282
АБСОЛЮТНЫЕ И ОТНОСИТЕЛЬНЫЕ ССЫЛКИ В ЗАПРОСАХ.....	283
<i>Ссылка на конкретную ячейку в столбце</i>	283
<i>Ссылка на предыдущую/следующую ячейку</i>	284
ПАРАМЕТРИЗАЦИЯ ЗАПРОСОВ.....	286
ДОБАВЛЕНИЕ ПРОСТЫХ ПАРАМЕТРОВ К ЗАПРОСУ	287
ПАРАМЕТРИЗАЦИЯ ПУТЕЙ К ФАЙЛАМ ИСХОДНЫХ ДАННЫХ	290
<i>Постановка задачи</i>	290
<i>Создаем запрос к внешнему файлу</i>	291
<i>Находим путь к файлу в запросе</i>	292
<i>Вводим путь как параметр</i>	293
ПРЕОБРАЗОВАНИЕ ЗАПРОСА В ФУНКЦИЮ НА ПРИМЕРЕ ВЕБ-ЗАПРОСА КУРСА ВАЛЮТ.....	295
<i>Этап 1. Создаём простой веб-запрос</i>	295
<i>Этап 2. Преобразуем запрос в функцию</i>	297
<i>Этап 3. Применяем созданную функцию</i>	299
ЗАГРУЗКА «ПЛАВАЮЩЕГО» ФРАГМЕНТА ДАННЫХ.....	300
<i>Способ 1. Маркеры начала и конца</i>	300
<i>Способ 2. Вводим переменные</i>	302
ВЫБОРКА ФРАГМЕНТА ПРИ МАССОВОЙ ЗАГРУЗКЕ ФАЙЛОВ.....	306
<i>Шаг 1. Одиночный запрос</i>	306
<i>Шаг 2. Преобразуем запрос в функцию</i>	306
<i>Шаг 3. Собираем файлы и применяем нашу функцию</i>	308
ТАНЦЫ НА ГРАБЛЯХ	310
ДОБАВЛЕННЫЕ ИЛИ УДАЛЁННЫЕ СТОЛБЦЫ В ДАННЫХ.....	311
<i>Шаг 1. Идеальная таблица</i>	311
<i>Шаг 2. Добавляем запросы</i>	312
<i>Шаг 3. Убираем лишнее</i>	312
МУСОР В НАЗВАНИЯХ СТОЛБЦОВ	313
ПЕРЕИМЕНОВАНИЕ СТОЛБЦОВ.....	315
УДАЛЕНИЕ НЕСУЩЕСТВУЮЩИХ СТОЛБЦОВ	317
<i>Способ 1. Сохранять вместо удаления</i>	317
<i>Способ 2. Удалять по номеру, а не по имени столбца</i>	317
<i>Способ 3. Удалять по признаку</i>	318
ИЗМЕНЕНИЕ ПОРЯДКА СТОЛБЦОВ	319
ОПАСНЫЙ ФИЛЬТР.....	321
ВОПРОСЫ БЫСТРОДЕЙСТВИЯ.....	323
<i>Отключите фоновое обновление</i>	323
<i>Отключите проверку конфиденциальности</i>	324
<i>Используйте CSV-файлы вместо книг Excel</i>	324
<i>Откажитесь от прямых ссылок на ячейки</i>	324
<i>Используйте функции Table.Buffer и List.Buffer</i>	324
<i>Добавьте ключ при слиянии запросов</i>	325

Введение

Самая короткая глава, которая объясняет:

- для кого и зачем написана эта книга.
- как правильно её использовать.
- где скачать файлы для всех примеров из этой книги.



Для кого эта книга

*Каждый успешный человек – это прежде всего книги, которые он прочитал вовремя.
(Артёмий Лебедев)*

Предполагаю, что если вы это читаете, то вас, скорее всего, можно отнести к одной из трёх групп пользователей Microsoft Excel:

Новички в теме

Вы никогда не слышали о Power Query и не понимаете, при чём тут Excel? ОК. Если очень кратко, то Power Query – это надстройка для Excel, которая решает две основные задачи: импорт данных в Excel из любых (практически) источников во внешнем мире (файлов, программ, интернета) и трансформацию этих данных в любой нужный вам вид. Причем сделать всё это нужно лишь один раз, а потом запрос достаточно просто обновлять.

Начиная с 2016 версии Power Query стал уже неотъемлемой частью Excel. Уверен, что очень скоро к минимальному комплекту базовых знаний в виде «функция ВПР + сводные таблицы» добавятся навыки работы с этой надстройкой. Начните изучать её уже сегодня, освоите базовые приёмы и принципы – и вы станете на голову выше своих коллег и получите огромную фору по сравнению с остальными.

Интересующиеся

Если вы уже что-то слышали о Power Query, видели её в работе или даже попробовали делать в ней простые запросы самостоятельно, то вам возможности этой надстройки, думаю, рекламировать уже не надо: на работу с ней подсаживаются с первого раза. К хорошему привыкаешь быстро, да. Проблема лишь в том, что источников информации (особенно русскоязычных) по этой теме не очень много, да и на английском тоже пока одной руки хватит перечислить. Именно поэтому я и сел год назад писать эту книгу, чтобы сделать максимально подробный и понятный самоучитель-справочник по возможностям этой замечательной программы. Надеюсь, это (хотя бы отчасти) получилось и эта книга вам поможет.

Профи

Если вы уже серьёзно используете Power Query в своей работе, то мы с вами на одной волне. Просмотрите оглавление: возможно, найдутся темы, которые вас заинтересуют. Обработка данных в Power Query – это всегда немного импровизация, так что, возможно, вы сможете почерпнуть из этой книги пару фишек, до которых не докопались самостоятельно.































Если же, наоборот, вам есть что посоветовать, предложить или улучшить в описанных методиках, то я (и всё прогрессивное человечество) буду вам очень благодарны, если вы поделитесь вашим опытом.

Файлы с примерами из этой книги

*Человек усваивает 10% из того, что прочитал,
30% из того, что увидел, и 90% из того, что
сделал сам.
(Тренерская мудрость)*

Книга – это хорошо, но живой пример ещё лучше! Почти ко всем задачам из этой книги есть файлы, где можно посмотреть работающие запросы и формулы и даже скопировать их оттуда в свой проект. Не стесняйтесь.

Имена файлов повторяют названия глав, так что найти нужное будет несложно:

 Абсолютные и относительные ссылки в запросах - старт.xlsx	 Настройка уровней конфиденциальности.xlsx
 Абсолютные и относительные ссылки в запросах - финиш.xlsx	 Нечеткий текстовый поиск - финиш.xlsx
 Вычисление длительностей - старт.xlsx	 Номер недели по ISO - старт.xlsx
 Вычисление длительностей - финиш.xlsx	 Номер недели по ISO - финиш.xlsx
 Генератор фраз декартовым произведением - старт.xlsx	 Обработка ошибок в запросах.xlsx
 Генератор фраз декартовым произведением - финиш.xlsx	 Объединение по нескольким столбцам - старт.xlsx
 Группировка запросов - старт.xlsx	 Объединение по нескольким столбцам - финиш.xlsx
 Группировка запросов - финиш.xlsx	 Объединение таблиц забудьте про ВПП - старт.xlsx
 Группировка строк - старт.xlsx	 Объединение таблиц забудьте про ВПП - финиш.xlsx
 Группировка строк - финиш.xlsx	 Опасный фильтр - старт.xlsx
 Добавление двух таблиц - старт.xlsx	 Опасный фильтр - финиш.xlsx
 Добавление двух таблиц - финиш.xlsx	 Основы синтаксиса языка M - финиш.xlsx
 Добавление простых параметров к запросу - старт.xlsx	 Отмена свертывания простой таблицей - старт.xlsx
 Добавление простых параметров к запросу - финиш.xlsx	 Отмена свертывания простой таблицей - финиш.xlsx
 Добавление трех и более таблиц с загрузкой в Модель Данных - старт.xlsx	 Отмена свертывания сразу нескольких таблиц - старт.xlsx

130 файлов с примерами

Обратите внимание, что большинство файлов существуют в двух версиях: «старт» и «финиш».

Финишные файлы содержат готовые запросы и формулы. В стартовых есть лишь исходные данные, на которых можно тренироваться, выполняя все описанные в книге приёмы прямо по ходу изучения материала.

Где взять эти файлы:

- если вы купили эту книгу в **электронном виде**, то файлы примеров должны идти в комплекте архивом;
- если вы купили **бумажную версию**, то файлы можно скачать на моем сайте в разделе «**Книги**»: www.PlanetaExcel.ru/books

Лучший способ внедрить описанные в этой книге трюки и приёмы – это незамедлительно попробовать применить новые знания на практике и в работе, а ещё лучше – поделиться ими с другими (так усваивается лучше всего, я-то уж знаю, поверьте).

Тренинги

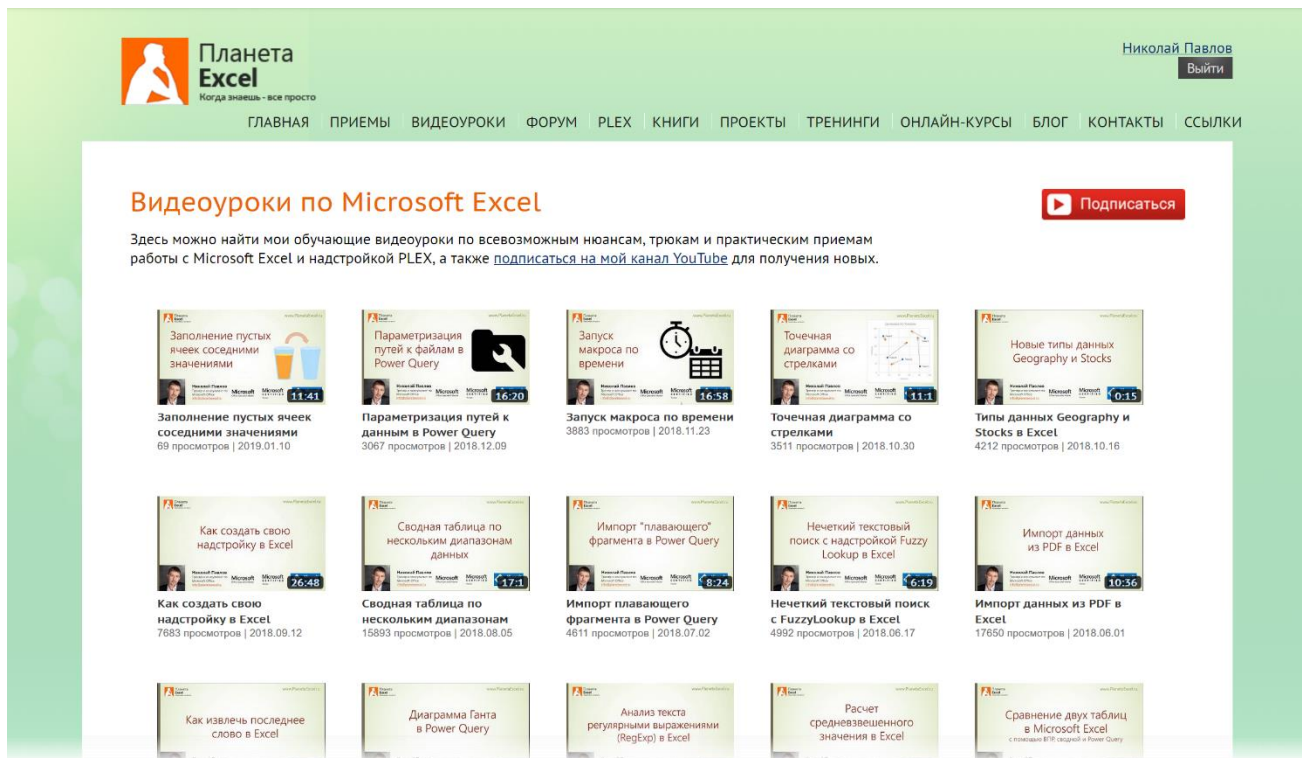
Уже больше 10 лет мы с моими коллегами-тренерами по команде регулярно проводим открытые и корпоративные тренинги для пользователей Microsoft Excel, обучая более полутора тысяч слушателей в год. В нашем портфолио больше десятка тренингов, в том числе и курсы по Power Query, Power Pivot, программированию макросов на VBA, прогнозированию и визуализации и т. д.

По сути, эта книга и выросла из одного из таких тренингов.

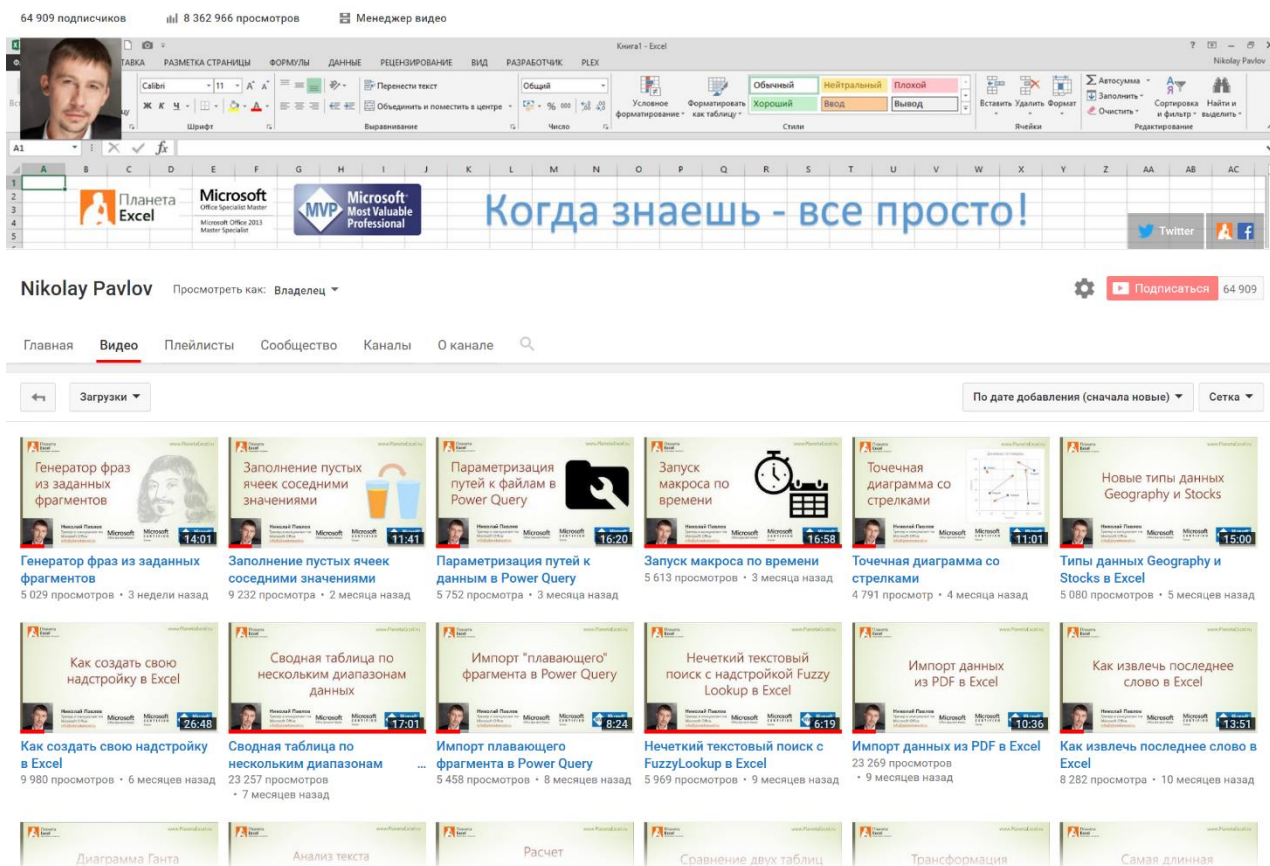
Если у вас есть возможность и желание, приходите на наши занятия, буду рад познакомиться с вами лично и помочь в решении ваших задач. Подробности можно найти по адресу <https://www.planetaexcel.ru/treningi/>.

Видеоуроки и канал на YouTube

Если вы из тех, кому лучше один раз увидеть, то рекомендую заглянуть на мой сайт в раздел **«Видеоуроки»** www.PlanetaExcel.ru/video, где я регулярно выкладываю обучающие видеоуроки по различным техникам и аспектам применения Microsoft Excel и Power Query в реальных бизнес-задачах:



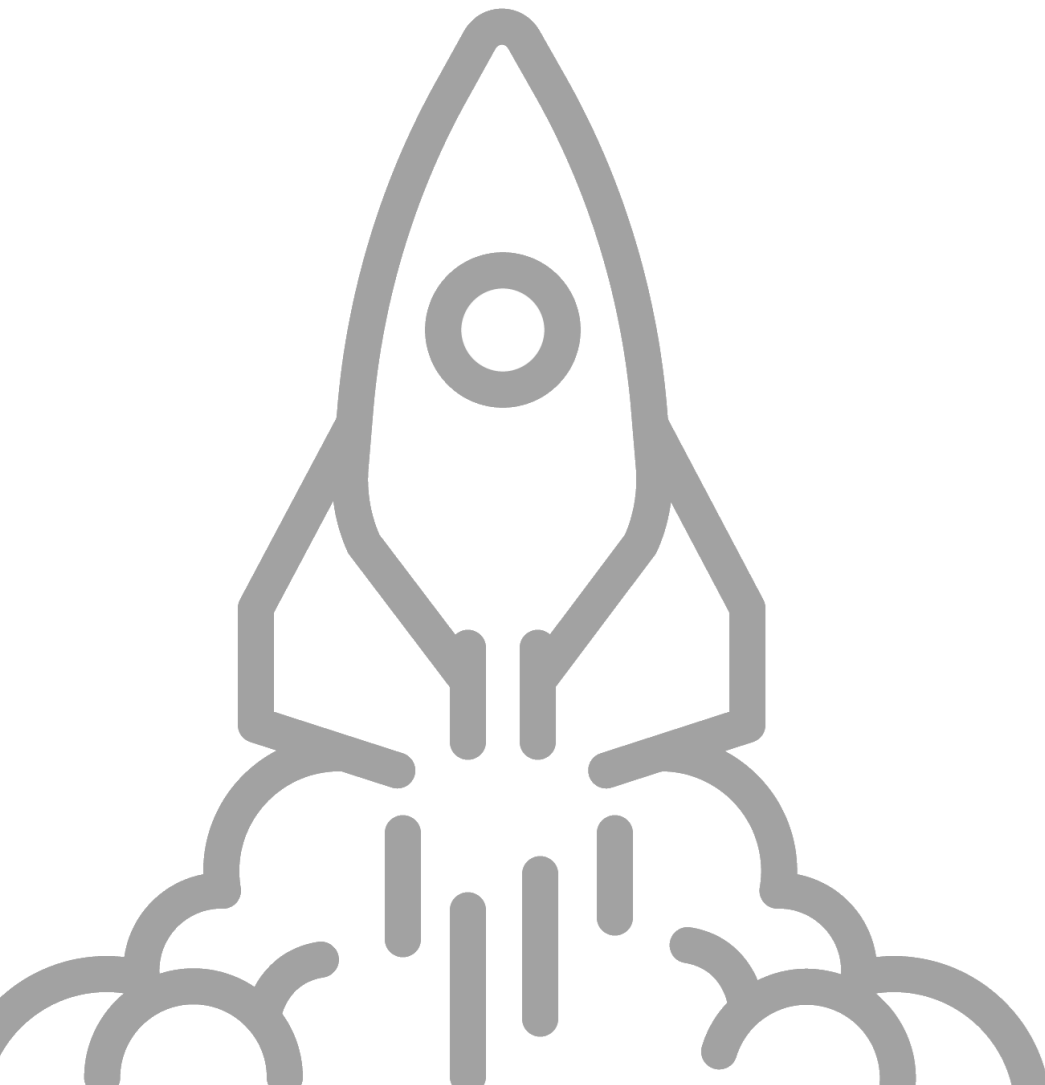
Эти же видео (порой с весьма интересными комментариями и обсуждениями к ним) можно всегда найти и на моём YouTube-канале <https://www.youtube.com/user/planetaexcel/videos>.



Начало работы с Power Query

В этой главе мы:

- разберемся с тем, **что же такое Power Query** и как он появился;
- какие **версии** Power Query бывают и откуда их скачать;
- как Power Query взаимодействует с другими приложениями и настройками из BI-семейства: Power Pivot, Power Map, Power BI и т. д.;
- уточним, **что может и чего не может** Power Query, в чём он хорош, а где его лучше не использовать;
- разберемся с тем, **откуда** можно загружать данные в Power Query;
- пошагово и подробно пройдем весь **процесс загрузки и обработки данных** на примере CSV-файла.

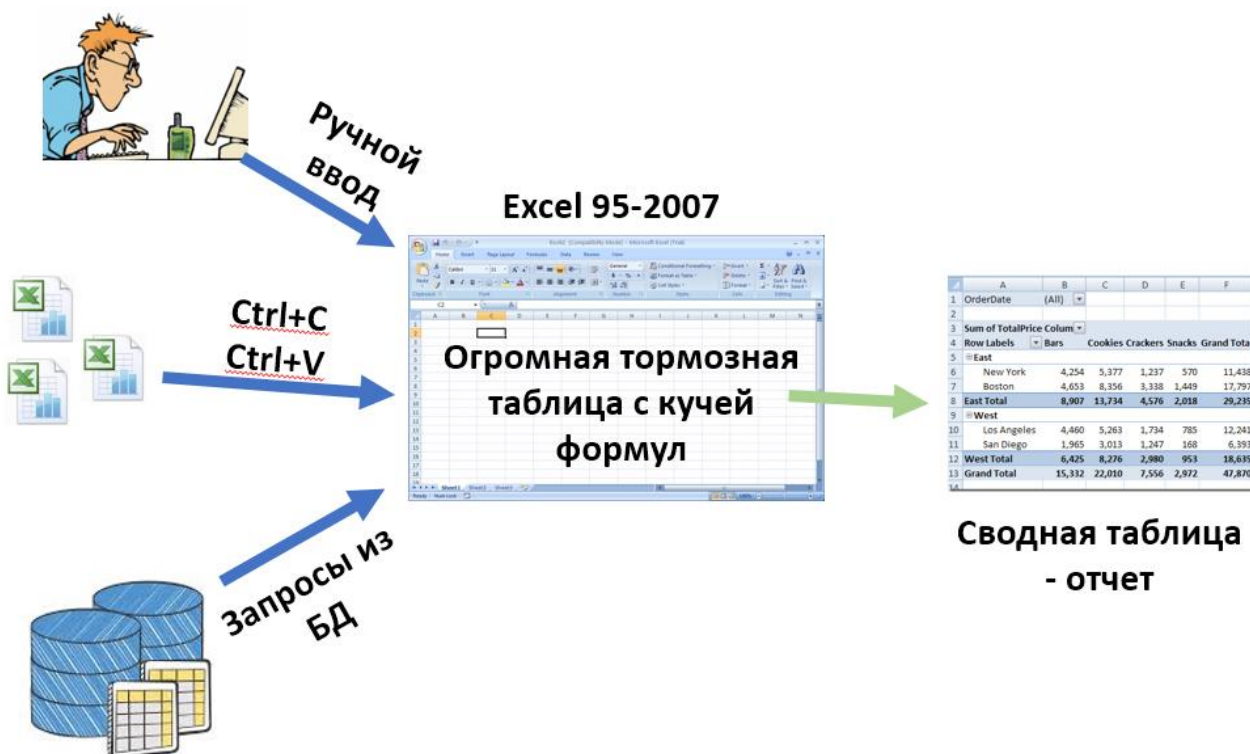


Тихая революция

*Мы так часто пилим, что совсем забываем заточить пилу.
(Стивен Кови)*

*Знаешь, зарплата не очень, зато работа сложная
(кф. «Книжный магазин Блэка»)*

Как обычно выглядит типовой алгоритм работы большинства пользователей Excel, сталкивающихся с анализом данных? Схематично его можно изобразить так:



Давайте я поясню.

Сначала данные для анализа надо как-то собрать, правильно? Тут на выбор есть:

- **ручное копирование** необходимых данных из исходных файлов. В смысле, тупой копипастинг. Долго, скучно и, как следствие, с большой вероятностью человеческих ошибок;
- сбор данных с помощью создания **в формулах прямых ссылок на исходные файлы**. Уже лучше, но тут велика вероятность последующих проблем со связями между книгами (медленная загрузка, ошибки #ССЫЛКА при обновлении, сбившиеся формулы при переименовании исходных файлов и т. п.);
- **макросы для сбора данных**. Хороший способ, но только если вы умеете программировать на Visual Basic на соответствующем уровне. Ну, или у вас есть знакомый программист;
- Прямое **подключение к базам данных**. Какую-то часть необходимой для отчета информации придется, возможно, выгрузить из корпоративных программ учета (баз данных, 1С, ERP-систем и т. п.). Тут все очень зависит от конкретной программы. Иногда их интерфейс бывает не очень дружелюбным, вплоть до необходимости написания запросов на языке баз данных SQL, что не каждому под силу.

Когда данные собраны, их необходимо обработать и довести до состояния, пригодного для последующего анализа. Обычно это включает:

- **зачистку данных.** Такие мелочи, как удаление лишних пробелов, пустых строк, объединенных ячеек, исправление регистра, преобразования «чисел-как-текст» в нормальные и т. п. требуют на самом деле приличного количества времени;
- **удаление мусора.** Иногда исходные данные содержат лишнюю информацию. Особенно часто это встречается при выгрузках из баз данных и ERP-систем. Эти лишние строки и столбцы нужно убрать;
- **обогащение данных.** Термином «обогащение» в теории БД принято обозначать подгрузку в наши данные подробностей из других таблиц-справочников. Например, цен из прайс-листа по артикулу товара или адреса клиента по его номеру договора. В большинстве случаев для этого используют формулы и функции – известные многим **ВПР (VLOOKUP)**, **ИНДЕКС (INDEX)**, **ПОИСКПОЗ (MATCH)** и т. д. Часто на этом этапе от большого количества таблиц и сложных формул Excel уже начинает под тормаживать, а пользователь – лезть на стену.

И когда вся информация собрана и приведена в божеский вид, то можно (наконец-то!) приступать непосредственно к анализу – строить сводные таблицы, писать формулы и рисовать красивые диаграммы для презентации руководству.

Знакомо?

Задумайтесь: на сбор данных и приведение их в порядок уходит в разы больше времени, чем на сам анализ. А через месяц/квартал/неделю, когда приходит время очередного отчета, вся эта адская карусель раскручивается заново, сжигая рабочие часы и нервные клетки.

На самом деле все может и должно быть иначе. Выход есть.

Еще в 2013 году специально созданная группа разработчиков внутри Microsoft выпустила для Excel специальную надстройку – **Power Query** (другие названия – **Data Explorer**, **Get & Transform**). Она предназначена для загрузки данных в Microsoft Excel из любых источников и всяческих трансформаций загруженных данных для последующего анализа.

С ее помощью большинство из перечисленных выше задач по сбору и «причёсыванию» данных решаются за считанные минуты, легко и изящно, порой даже без касания клавиатуры! Мощь и польза от Power Query были настолько велики, что начиная с версии Excel 2016, Microsoft уже внедрила эту надстройку в Excel по умолчанию, добавив все её возможности к стандартному функционалу своего редактора электронных таблиц.

Так что будущее уже здесь: тихая революция свершилась. А если вы держите эту книгу в руках, то вы её участник.



Что может и чего не может Power Query

*Если кто-то не оправдал ваши ожидания, то в этом нет его вины. Ведь это ваши ожидания.
(Народная мудрость)*

Прежде чем продолжать и переходить к деталям, давайте разберемся с тем, что может и чего не может Power Query. Чтобы понимать, в каких задачах ее можно использовать, а где она не поможет в принципе.

Итак, **Power Query МОЖЕТ**:





- **загружать** в Microsoft Excel для последующей обработки данные из почти любых источников (файлов, баз данных, интернета, эл. почты и т. д. – подробно об этом в следующей главе);
- **собирать данные** из нескольких листов, файлов или папок (включая подпапки);
- **трансформировать** загруженные данные (фильтровать, сортировать, группировать, сворачивать или разворачивать в столбцы по заданному признаку и т. д.);
- **связывать и объединять** таблицы между собой (без функций типа **ВПР**, **ИНДЕКС**, **ПОИСКПОЗ** и т. п.);
- работать с **датами и текстом** (зачистка, исправления регистра, лишних пробелов и т. д.);
- выполнять **простые вычисления** с данными, включая логику (аналог функции **ЕСЛИ**).

При этом **Power Query НЕ УМЕЕТ**:

- **редактировать загруженные данные напрямую**, т. е. нельзя, как на листе Excel, щелкнуть мышью в ячейку и исправить любое значение, если вдруг захочется. Тут есть своя логика: хочешь менять данные – меняй их в источнике, откуда они загружаются;
- выполнять **сложные статистические и математические расчеты**. Сумму, среднее и даже медиану посчитать Power Query вполне может, но сложные функции типа финансовых или статистических здесь отсутствуют. Тем не менее вполне можно добавить их потом, дописав нужные формулы рядом с выгруженными из Power Query на лист таблицами;
- **визуализировать** и наглядно представлять числовые данные. Power Query – это не про визуализацию, а про данные. Диаграммы, условное форматирование, спарклайны и прочая красота здесь отсутствуют. Опять же, если они нужны, то никто не запрещает использовать их потом, применив к выгруженным из Power Query на лист значениям, или перейдя на использование Microsoft Power BI – отдельного приложения для бизнес-аналитики с шикарными визуальными возможностями (см. <https://powerbi.microsoft.com>), включающего, кстати, в себя весь функционал Power Query;
- мгновенно и на лету **пересчитывать результаты**, как это происходит с обычными формулами в Excel. Здесь при изменении исходных данных придется запускать обновление запросов вручную;
- **разделять данные** на несколько таблиц, листов или файлов. Собрать – это запросто, а вот разборку пока в Excel возможно реализовать только вручную или с помощью макросов.

Откуда можно загружать данные в Power Query

Детальный разбор наиболее часто встречающихся вариантов импорта мы сделаем в следующих двух главах. Сейчас же хотелось в двух словах описать всю картину целиком. В качестве источника данных для Power Query может выступить практически всё.

Базы данных	Интернет	ERP, CRM	Файлы
      	    	   	      

Базы данных

Это наиболее легкий и удобный путь получения информации, т. к. база данных сама по себе является их упорядоченным хранилищем с таблицами, связями, данными определенного типа и т. д. Power Query умеет подключаться почти ко всем существующим на сегодняшний день более-менее известным базам данных, а с остальными может общаться через ODBC (Open Database Connectivity) – специальный программный интерфейс доступа к базам данных.

Интернет и облачные хранилища

Power Query умеет выхватывать нужные данные с веб-страниц с последующим их обновлением. Такая возможность очень пригодится, если вам нужно, например, собирать прайс-листы конкурентов с их сайтов для дальнейшего анализа и сравнения. Причем путем небольших ухищрений путь к нужной странице можно задавать как переменную (параметр). Это позволит легко получать, допустим, обменный курс с сайта банка или расписание с веб-страницы авиакомпании на заданную дату.

Также Power Query «дружит» с Outlook и умеет загружать данные из почтовых ящиков Microsoft Exchange: контакты, список встреч из вашего календаря или писем из почтовых папок. Поддерживается импорт из облачных сервисов Microsoft Azure. Можно даже загрузить ваши посты с Facebook, если хотите.

Корпоративные программы

По умолчанию Power Query умеет соединяться и запрашивать данные из нескольких самых известных в мире программ для управления предприятиями (SAP, Microsoft Dynamics) и CRM-системы Salesforce.

Некоторые сайты и программы (например, популярная в России и соседних странах 1С) поддерживают обмен данными по протоколу OData (Open Data Protocol) – специально разработанному открытому набору правил для запросов и обновления данных между компьютерами по сети. Power Query тоже его понимает.

Файлы и папки

Само собой, Power Query умеет отлично импортировать любые данные из файлов Microsoft Excel любых типов (XLS, XLSX, XLSB, XLSM), лишь бы они не были защищены паролем.

Также легко загрузить данные из текстовых файлов любого формата: обычного TXT, или разделенного запятыми или точкой с запятой CSV.

Текстовые и экселевские форматы замечательно импортируются оптом: достаточно указать папку, и все найденные в ней и вложенных подпапках файлы будут автоматически загружены в Power Query.

Поддерживается импорт из общепринятых веб-форматов: HTML, XML, JSON.

И путем некоторых «танцев с бубном» возможен даже импорт из PDF (с промежуточным этапом в виде распознавания PDF в Microsoft Word).

Другие источники

Что делать, если в перечисленном выше списке нет нужного вам варианта? Если вашу корпоративную базу данных писали 100 лет назад бородатые программисты на Fortran'е под MS-DOS? Или, наоборот, она суперновая и уникальная? Вариантов тут три.

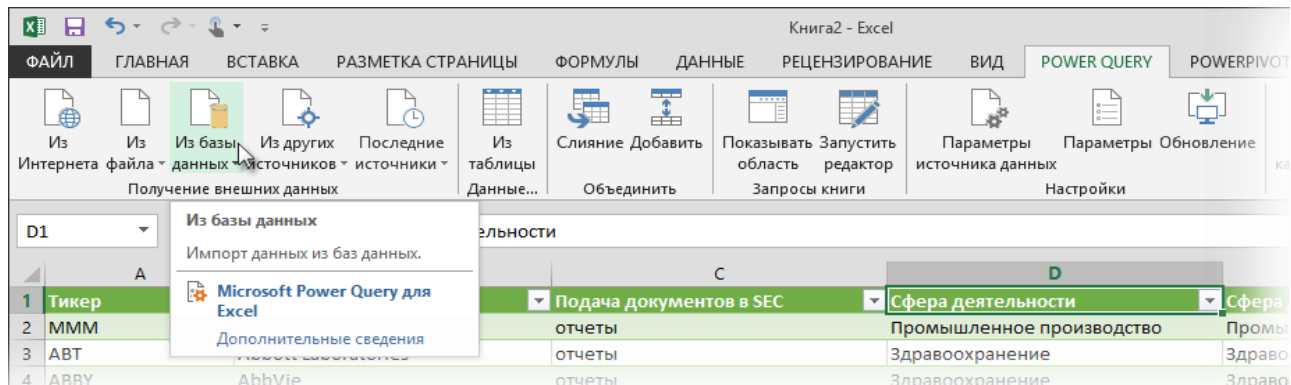
- **Поставить другую версию Office.** Существует несколько вариантов поставки Microsoft Office (Home & Student, Enterprise, Business и т.д.) и наборы коннекторов в них отличаются. Самый полный набор, насколько я могу судить по своему опыту, имеется в версии Office 365 Pro Plus.
- **Ждать и надеяться.** Возможно, Microsoft добавит нужный вам источник с очередным обновлением Power Query (а они выходят несколько раз в год). Так было, например, с Facebook и SAP HANA: возможность подключения к ним появилась после одного из недавних обновлений.
- **Поискать в интернете.** Многочисленные энтузиасты и программисты давно разрабатывают для Power Query свои коннекторы для загрузки данных из нестандартных программ или форматов файлов. Как насчет импорта статистики своих пробежек из Strava или загрузки xls, но в zip-архиве? Без проблем, всё это уже существует. Рекомендую для начала заглянуть в репозиторий GitHub с готовыми примерами и документацией по пользовательским коннекторам <https://github.com/Microsoft/DataConnectors> или искать в интернете по фразе «*Power Query custom connector*».

Версии и обновления

На момент написания этой книги (март 2019 года) Power Query существует в нескольких вариантах.

Отдельная надстройка для Excel 2010–2013

Для старых версий Excel 2010–2013 Power Query подключается как надстройка, которую можно бесплатно скачать с сайта Microsoft (<https://www.microsoft.com/ru-RU/download/details.aspx?id=39379>) и установить на ваш компьютер. После установки в интерфейсе Microsoft Excel добавится соответствующая вкладка:



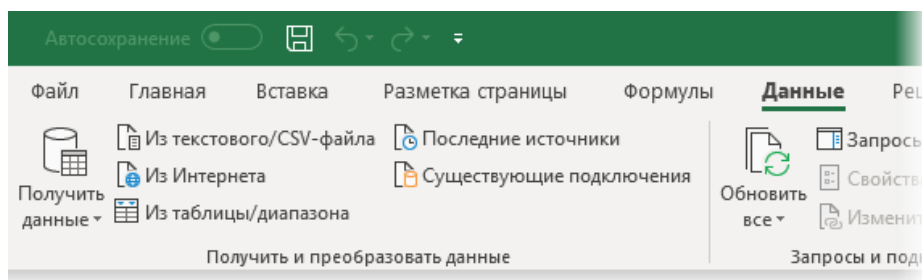
Обратите внимание, что при скачивании надстройка Power Query предлагается в 32- или 64-битной версиях. Нужно выбирать ту же разрядность, что и у вашего Excel.

Минимальные тех.требования: Windows 7, Office 2010 и Internet Explorer 9 или новее.

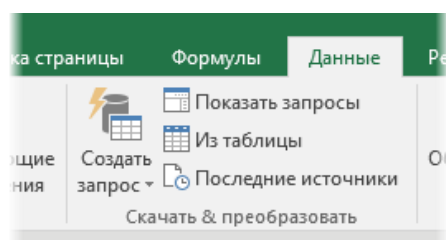
Встроенный функционал в Excel 2016–2019

Если у вас Excel 2016–2019, то надстройка Power Query уже встроена в него по умолчанию, и все её инструменты находятся на вкладке **Данные (Data)**. В зависимости от того, какая у вас версия Excel (по подписке Office 365 или автономная, с установленными последними обновлениями или нет), выглядеть это может немного по-разному.

В случае Office 365 это будет группа **Получить и преобразовать данные (Get & Transform Data)**:



В случае автономной версии Microsoft Excel это будет группа **Скачать & преобразовать (Get & Transform)**, расположенная там же на вкладке **Данные**, но чуть правее:



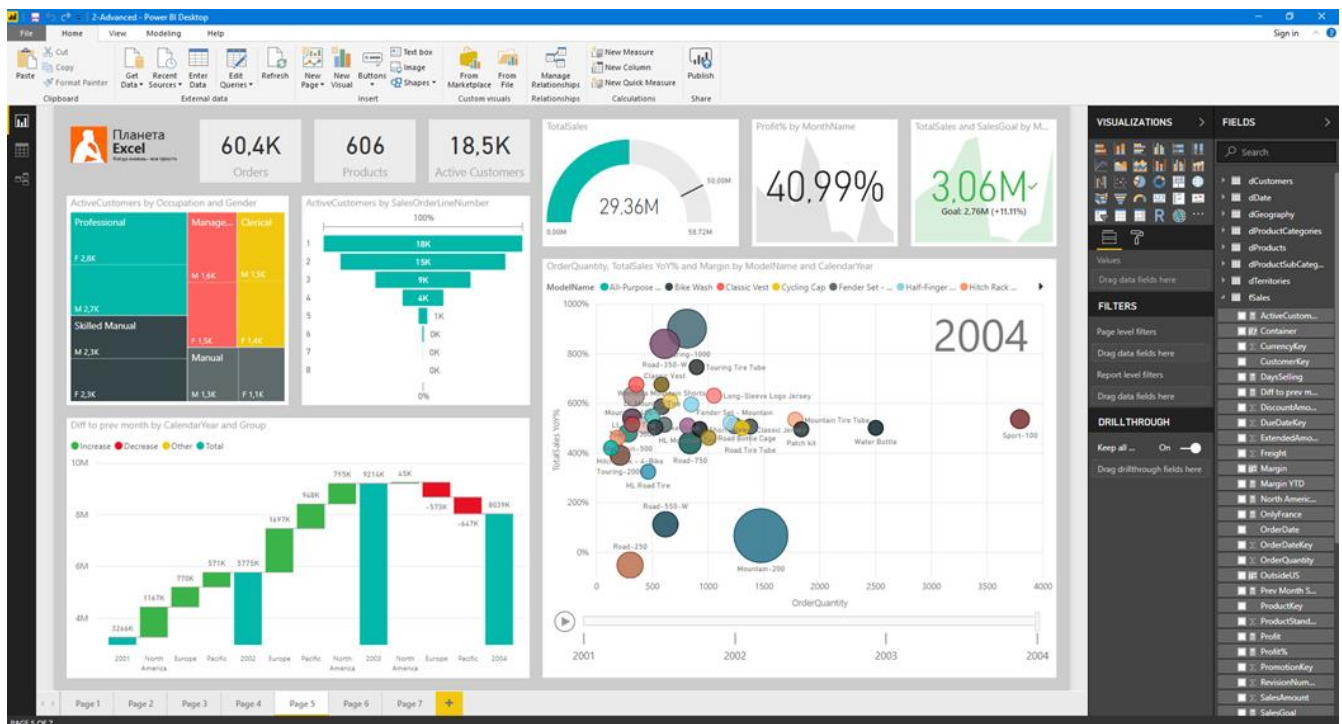
Обратите внимание, что в этой версии кнопка **Получить данные (Get Data)** будет называться уже **Создать запрос (New Query)**, но выполняет при этом абсолютно те же функции.

Часть Microsoft Power BI Desktop

Хотя в этой книге речь будет идти в основном об использовании Power Query в рамках Microsoft Excel, но на самом деле всё описанное на 100% применимо и к **Power BI Desktop** – мощнейшему средству сбора, анализа и визуализации любых данных. Дело в том, что эта программа включает в себя, по сути, связку из трех компонентов:

- весь функционал Power Query для сбора и приведения в порядок исходных данных;
- весь функционал надстройки Power Pivot для Excel и язык DAX (Data Analysis eXpressions) для анализа больших объемов данных и сложных вычислений с ними;
- большое количество красивейших анимированных диаграмм и визуализаций для данных, KPI, таблиц, географических карт и т. п.

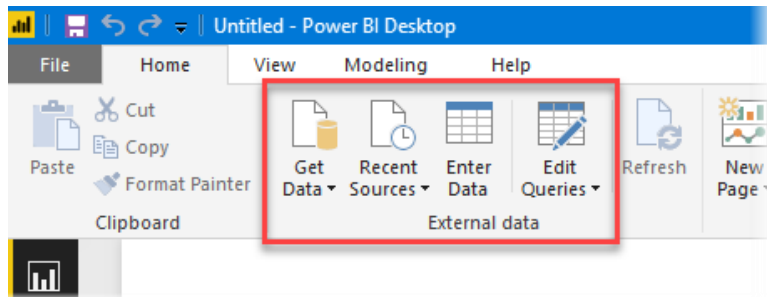
Причем созданный в Power BI Desktop отчет можно в считанные секунды опубликовать на сервере, и тогда он станет доступен для просмотра и при необходимости редактирования вашим коллегам, руководителям, клиентам и т. д. Выглядит всё это примерно так:



И в любой момент доступно практически на любом устройстве (смартфоне, планшете, компьютере) с сохранением всей интерактивности (!):



Весь инструментарий надстройки Power Query в приложении Power BI Desktop находится на главной вкладке в группе **Внешние данные** (Home → External data):



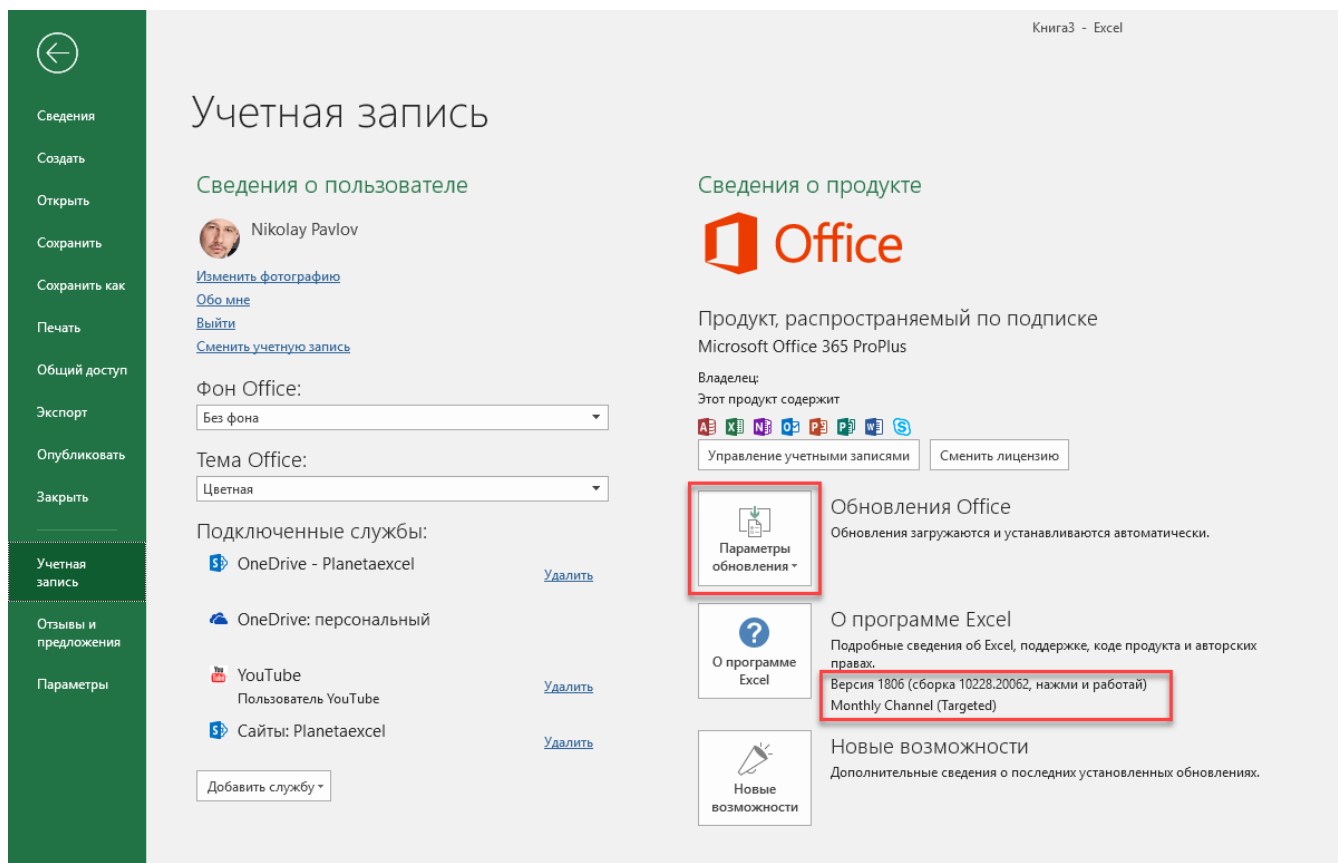
Функциональность всех перечисленных версий при этом почти одинаковая, за исключением, пожалуй, что только Power BI, который традиционно по возможностям и новым фичам опережает классический Office. Но если говорить про Excel, то совершенно непринципиально, какая именно версия Power Query у вас там будет. Я использую Office 365, так что все скриншоты в этой книге будут оттуда.

Что действительно важно, так это **своевременно устанавливать все обновления**, которые Microsoft выпускает для Microsoft Office и входящих в его состав Excel и Power Query. Если раньше обновления для Office выходили относительно редко и большими пакетами (SP = Service Pack), то сейчас команда разработчиков перешла на более мелкие, но частые (помесячные обычно) апдейты, каждый из которых оперативно исправляет обнаруженные ошибки и, самое главное, добавляет новые функции и инструменты.

Интересно, что Microsoft с некоторых пор очень внимательно прислушивается к пожеланиям и критике пользователей. Существует даже специальный сайт <https://excel.uservoice.com/>, где любой человек может опубликовать свою идею по улучшению Excel или проголосовать за приглянувшуюся ему чужую. Набранные больше всего голосов предложения реализуются Microsoft в ближайших обновлениях.

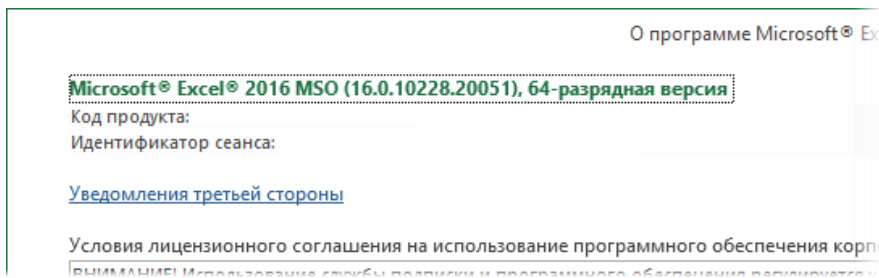
Так что если иллюстрации из этой книги отличаются от картинок на вашем мониторе, то, скорее всего, вам пора обновиться (или мне пора выпустить следующее издание этой книги).

Узнать вашу текущую версию и проверить обновления можно, выбрав команды **Файл → Учетная запись** (File → Account):



Под кнопкой **Параметры обновления** скрывается и нужная нам команда **Обновить (Update)**. Если вы её не видите у себя в Excel, значит, за загрузку и установку обновлений у вас отвечает кто-то другой (скорее всего, системный администратор), и обращаться надо к нему.

Нажав на кнопку **О программе Excel (About)**, также можно увидеть в появившемся окне текущую версию и разрядность вашего Excel:



Для больших и тяжелых задач очень рекомендуется установить 64-разрядную версию Office, для небольших хватит и 32 бит.

И последнее.

Опять же, на момент написания этой книги (июнь 2018 г.), Power Query **не поддерживался**:

- на версиях Excel 2007 (и всех предшествующих);
- любых версиях Excel для Mac;
- версиях Excel для iOS и Android;
- онлайн-версии Excel.

Sad but true, как говорится. Но, возможно, мы увидим подвижки в этом вопросе, и Power Query еще придет на другие платформы в будущем.

Основные принципы работы в Power Query на примере загрузки ТХТ-файла

*Сделай шаг, и дорога появится сама собой.
(Стив Джобс)*

Общая логика работы с любыми данными в Power Query укладывается в простую схему:



Сначала мы загружаем данные в Power Query, потом их приводим в нужный нам вид, затем выгружаем обратно в Excel. В будущем можно автоматически повторить всю цепочку описанных действий, просто обновив запрос.

Давайте пошагово и подробно рассмотрим весь процесс на примере импорта и последующей обработки текстового файла.

Постановка задачи

Предположим, что каждый день мы выгружаем из какой-нибудь корпоративной программы или базы данных текстовый файл с отчётом вот такого вида:

```

Отчет - Notepad
File Edit Format View Help
Отчет по продажам за янв-мар 2017 г
Дата создания: 12052017
Пользователь: Пупкин

Код заказа;Дата;Город;Бренд;Кол-во;Стоимость
ST5;20-01-2017;Нефтеюганск;Peugeot;6;1,849
SW6;30-01-2017;г.Великий Новгород;Mitsubishi;1;15,542
KJ8;25-01-2017;Братск;Renault;8;88,846
0ZS;04-01-2017;Петропавловск-Камчатский;Mitsubishi;8;97,127
U0I;31-01-2017;Новомосковск;Toyota;9;42,834
8ZE;05-01-2017;Черкесск;Subaru;6;21,575
UR7;09-01-2017;Улан-Удэ;Peugeot;8;24,843
ST5;20-01-2017;г.Нефтеюганск;Peugeot;6;1,849
ИТОГО=294,465

-----

LNR;11-02-2017;Первоуральск;Saab;5;69,134
KLH;27-02-2017;Грозный;Fiat;2;58,933
J6X;17-02-2017;г.Саранск;Honda;8;9,342
H0P;03-02-2017;Томск;Skoda;2;56,566
E71;09-02-2017;Великий Новгород;Mini;1;71,661
A0S;19-02-2017;Курган;Toyota;6;82,200
0BM;16-02-2017;Старый Оскол;Fiat;4;17,882
0BM;16-02-2017;Старый Оскол;Fiat;4;17,882
0BM;16-02-2017;Старый Оскол;Fiat;4;17,882
P20;11-02-2017;Орск;Hyundai;9;80,457
3G8;30-02-2017;г.Новороссийск;Suzuki;5;30,952
ИТОГО=512,891

-----

CFJ;07-03-2017;Якутск;Ford;3;17,325
L51;12-03-2017;Бердск;Saab;3;88,439
C6K;13-03-2017;Владикавказ;Saab;6;61,377
C6K;13-03-2017;Владикавказ;Saab;6;61,377
  
```

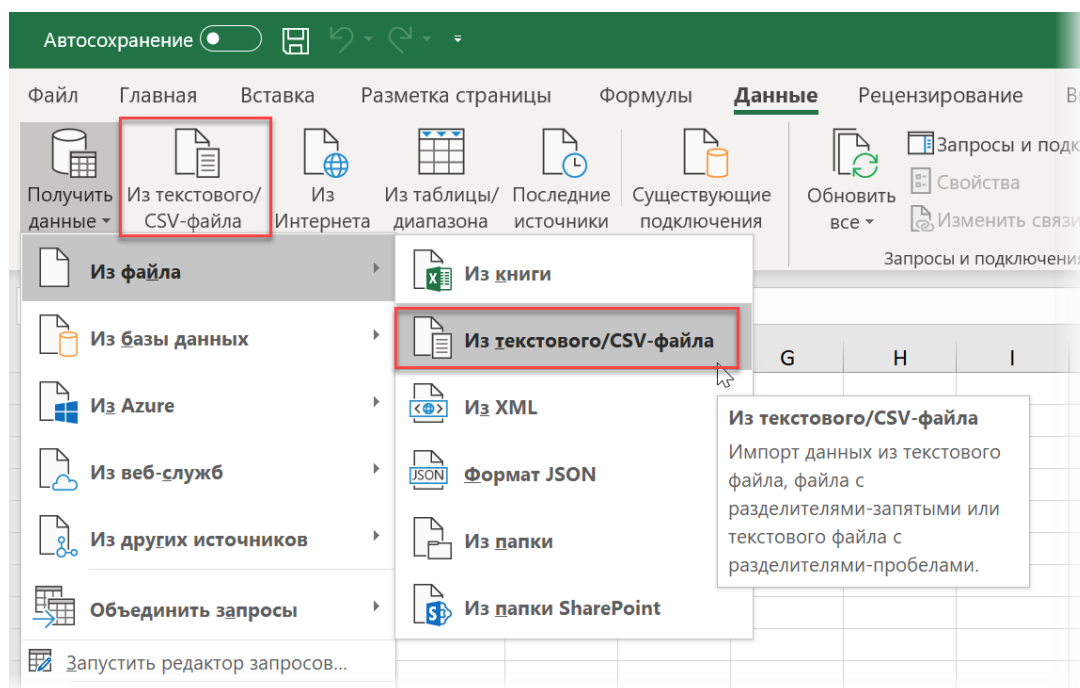
Обратите внимание на следующие моменты:

- В верхней части есть ненужная шапка с техническими данными по выгрузке, автору и т. д.
- Блоки данных разделены «для красоты» строкой из дефисов (причем их количество не везде одинаковое).
- В конце каждого блока есть итоговая строка с суммой (которая нам не нужна на самом деле, т. к. в будущем мы будем строить сводную таблицу по этим данным и считать все сами).
- Даты в формате, который не понимает Excel (через дефис).
- Город записан по-разному (с буквой «г» и без).
- В стоимостях используются нероссийские разделители (запятая вместо пробела как тысячный разделитель).
- В данных есть повторы.

Содержимое этого текстового файла нужно загрузить в Power Query, исправить все недочеты, а потом выгрузить обратно на лист и построить по «причесанным» данным сводную таблицу.

Загружаем файл

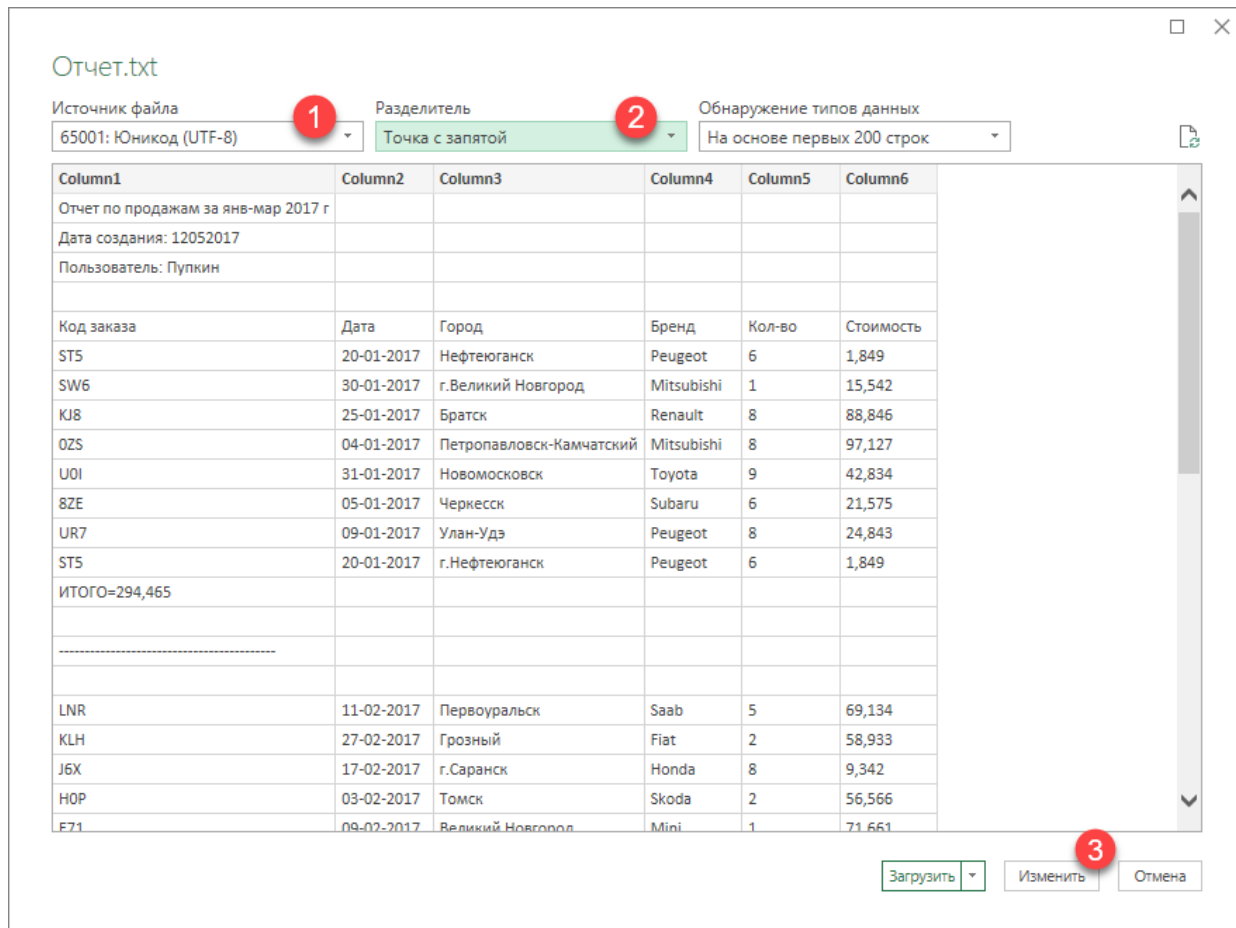
Чтобы загрузить содержимое этого файла, выберем на вкладке **Данные (Data)** или на вкладке **Power Query** (если она установлена как отдельная надстройка) команды **Получить данные** или **Создать запрос → Из файла → Из текстового / CSV-файла** (Get Data → From File → From text / CSV-file) или воспользуемся одноимённой кнопкой:



В появившемся окне нужно будет выбрать файл, а затем появится окно предварительного просмотра. Здесь нужно сделать следующее:

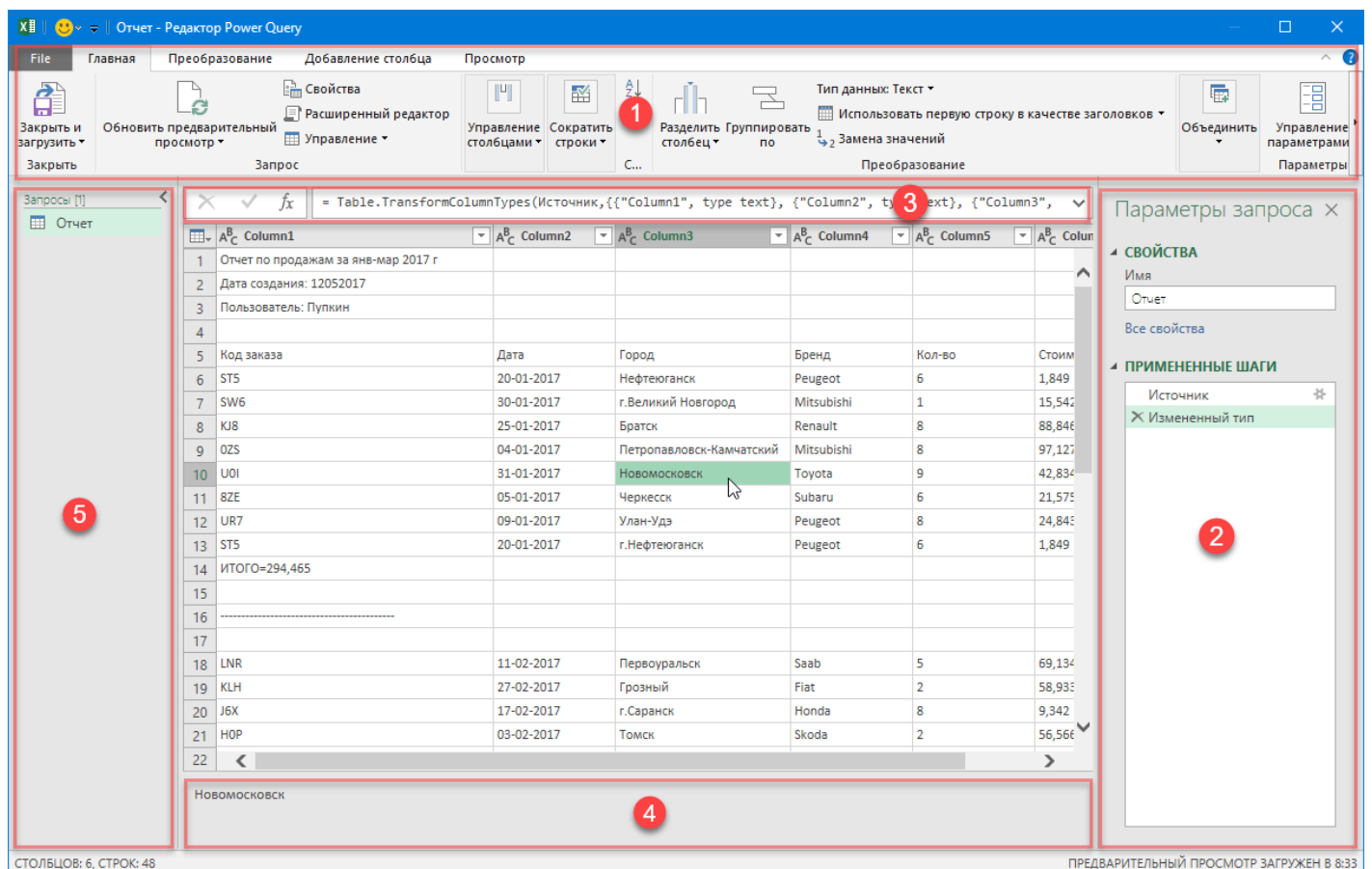
1. Проверить правильность выбранной кодировки в выпадающем списке **Источник файла**. Обычно это Юникод, но при импорте, например, файлов из старых систем, созданных еще под MS-DOS, кодировка может отличаться. Нужный вариант находится перебором, главное, чтобы русский текст не превращался в «крякозяблы».
2. Подобрать правильный разделитель столбцов (в нашем случае точка с запятой).
3. Нажать кнопку **Изменить¹** (Edit) для того, чтобы загрузить данные в Power Query для дальнейшей обработки. Соседняя кнопка **Загрузить** (Load) нам не нужна, т. к. поместит данные сразу на лист как есть, а нам их нужно сначала привести в порядок.

¹ В некоторых версиях Power Query эта кнопка может называться **Преобразовать данные** (Transform Data) или **Очистить данные** (Clean Data).



Окно редактора запросов

После нажатия на кнопку **Изменить** мы увидим главное окно Power Query – редактор запросов:



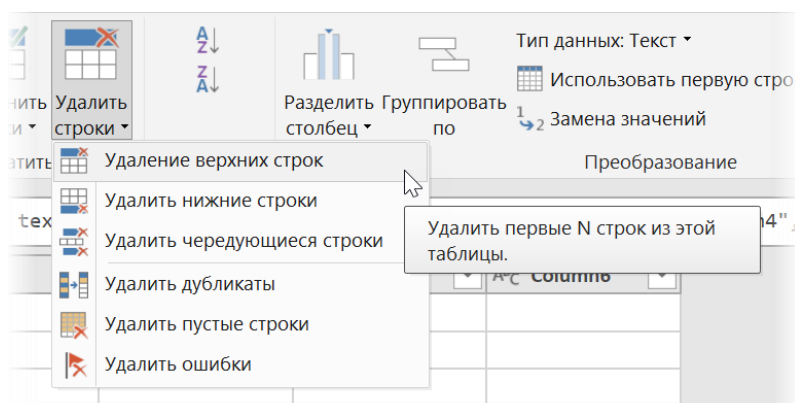
Давайте пробежимся по его элементам:

1. **Лента.** В верхней части окна виден привычный интерфейс – лента и вкладки **Главная (Home)**, **Преобразование (Transform)**, **Добавление столбца (Add column)** и **Просмотр (View)**. На этих вкладках находится большинство нужных нам команд для работы с данными.
2. **Параметры запроса.** В правой части окна расположена панель **Параметры запроса**, где можно дать имя нашему запросу, а чуть ниже виден список всех действий (шагов), которые мы уже применили к данным. Первым шагом **Источник (Source)** был запрос к файлу, а второй **Измененный тип (Changed Type)** Power Query автоматически добавил сам, пытаясь распознать тип данных (числа, текст, даты и т. п.) в каждом столбце. Любой шаг можно подкорректировать, нажав на знак шестеренки справа от шага, или удалить, щелкнув по кресту слева. Если вдруг этой панели не видно, то включить ее можно на вкладке **Вид → Параметры запроса (View → Query Settings)**.
3. **Строка формул.** Все выполненные действия в Power Query записываются в виде последовательности команд на специальном встроенном языке M. В строке формул отображается команда, соответствующая текущему выделенному шагу. Если строки формул не видно, то включить её можно на вкладке **Вид → Строка формул (View → Formula Bar)**. Общий вид запроса, т. е. все его команды на языке M, можно увидеть, если выбрать на той же вкладке **Просмотр** команду **Расширенный редактор (View → Advanced Editor)**.
4. **Панель просмотра.** Содержимое любой ячейки можно просмотреть в нижней панели, если щелкнуть мышью в белый фон (не текст!) ячейки. В Power Query, в отличие от Excel, в ячейках могут храниться не только одиночные числа или текст, но целые списки или даже таблицы.
5. **Панель запросов.** В одной книге Excel можно создать и использовать несколько запросов. Например, один запрос может брать данные с листа, другой – из интернета, а третий – соединять их между собой и выгружать затем результаты сборки на лист в виде сводной таблицы. Быстро переключаться между запросами можно как раз с помощью этой панели. По умолчанию эта панель обычно свернута (кнопка со стрелкой в правом верхнем углу панели).

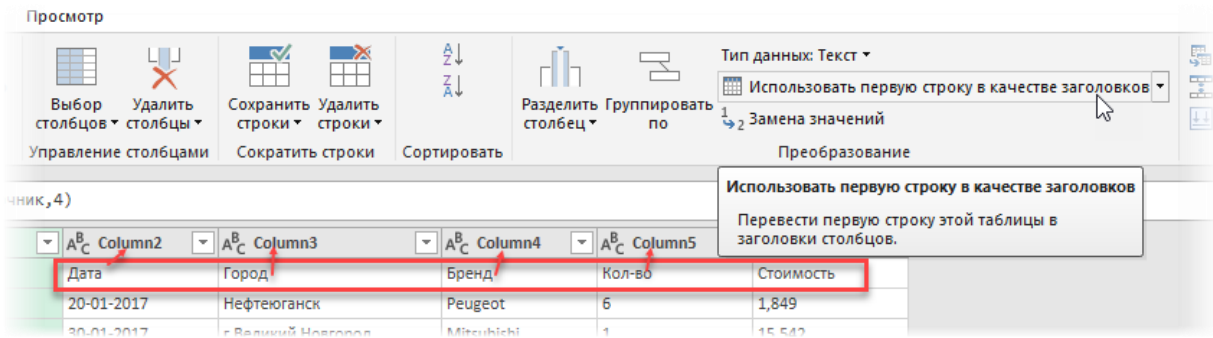
Наводим порядок в данных

Теперь давайте сделаем все необходимые действия для приведения наших данных в нормальный вид.

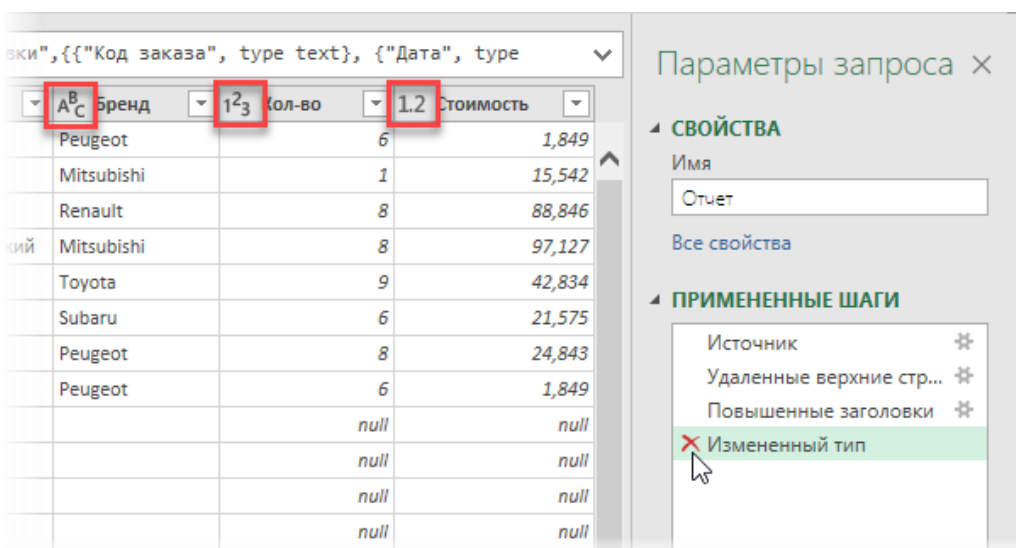
1. Удалите шаг **Измененный тип (Changed Type)** в панели **Параметров запроса**, т. к. пока еще рано назначать столбцам типы данных. Мы сделаем это позже.
2. Убрать четыре верхние ненужные строки можно на вкладке **Главная → группа Сократить строки → Удалить строки → Удаление верхних строк (Home → Remove rows)**:



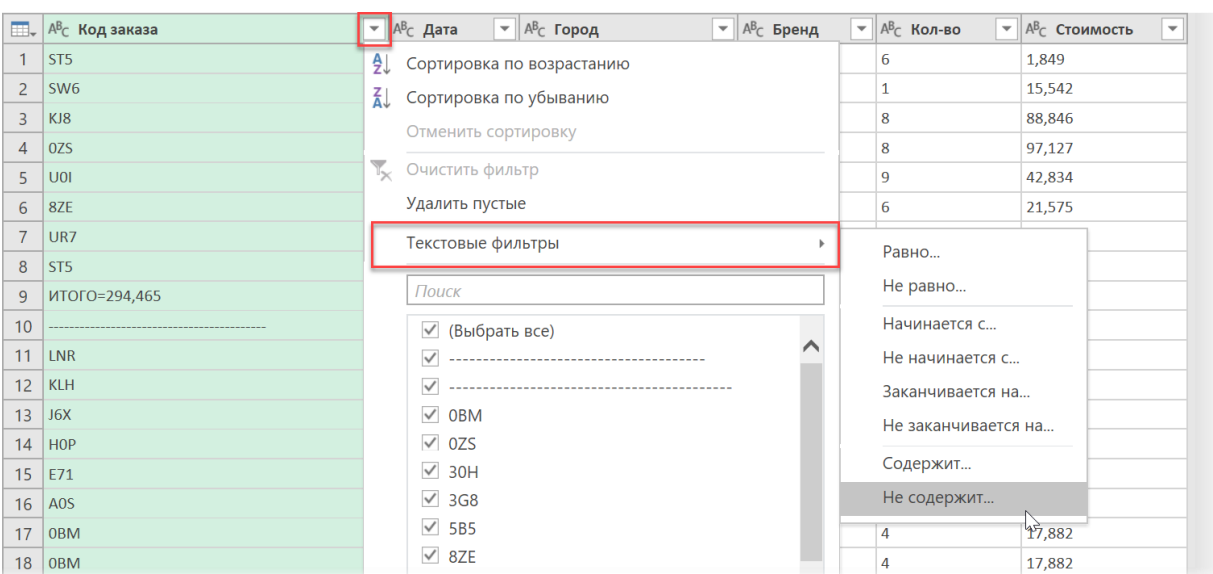
3. После удаления верхних строк окажется, что первая строка в наших данных содержит, по сути, заголовки столбцов. Чтобы поднять её в шапку таблицы, используем на **Главной** вкладке команду **Использовать первую строку в качестве заголовков (Home → Use First Row as Headers)**:



4. Обратите внимание, что после предыдущего шага Power Query опять услужливо попытался распознать типы данных в столбцах (см. значки в шапке) и самовольно добавил шаг **Измененный тип (Changed Type)**. Причем тип не везде определился правильно: в последнем столбце **Стоимость** запятую он понял как разделитель целой и дробной части, а она у нас на самом деле разделитель тысяч. Так что лучше опять удалить этот шаг, мы настроим типы самостоятельно чуть позже:



5. Избавимся от пустых строк. Для этого на вкладке **Главная** выберем **Удалить строки** → **Удалить пустые строки** (Home → Remove rows → Remove empty rows):
6. Чуть сложнее можно убрать строки-разделители с дефисами и итоги. Для этого в Power Query можно использовать фильтр. В обычном Microsoft Excel фильтр всего лишь скрывает ненужные данные. В Power Query же **фильтрация приводит к удалению ненужных строк**. Для это выберем в фильтре по столбцу **Код заказа** опции **Текстовые фильтры** → **Не содержит** (Text filters → Does not contains):



В появившемся окне можно ввести одно или несколько условий отбора:

Фильтрация строк

Базовый Подробнее

Сохранять строки, в которых "Код заказа"

не содержит ---

И Или

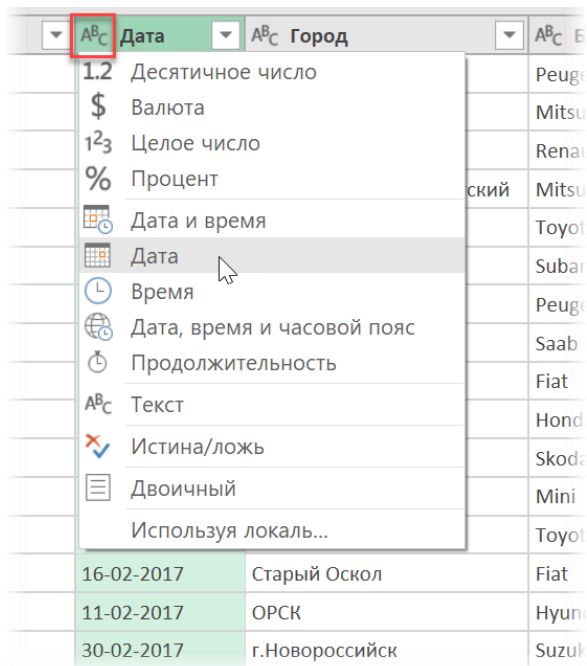
не содержит ИТОГО

Обратите особое внимание на то, как введено слово *ИТОГО* (заглавными буквами). В отличие от Excel **Power Query различает строчные и прописные буквы**. Невнимательность к регистру, по моему опыту, является одной из самых распространенных ошибок среди новичков в Power Query.

- Заметьте, что в первом столбце у нас попадаются дубликаты, давайте от них избавимся. Для этого можно использовать контекстное меню, которое в Power Query умеет очень многое и крайне удобно. Щёлкните правой кнопкой мыши по заголовку столбца **Код заказа**, выберите соответствующую команду – и дело сделано:

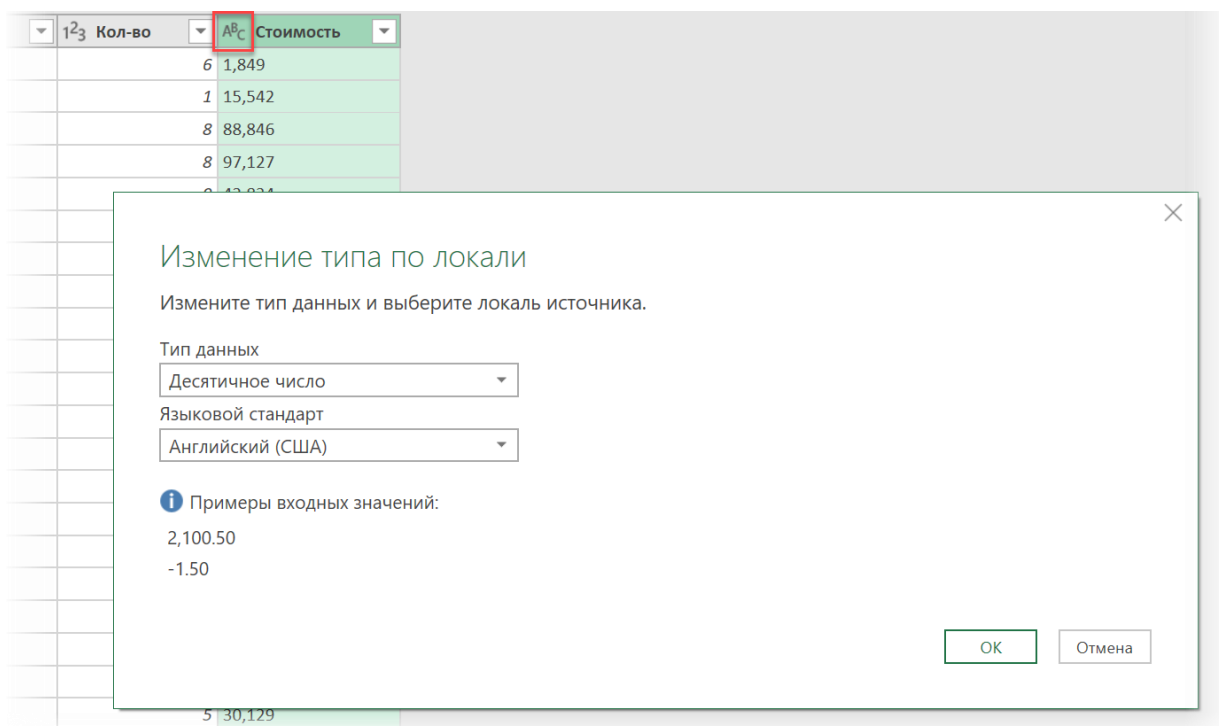
	ABС Код заказа	ABС Бренд	ABС Город
1	ST5	Peuge	
2	SW6	Mitsu	
3	KJ8	Renau	
4	OZS	Mitsu	
5	U0I	Toyot	
6	8ZE	Subar	
7	UR7	Peuge	
8	ST5	Peuge	
9	LNR	Saab	
10	KLH	Fiat	
11	J6X	Hond	
12	H0P	Skoda	
13	E71	Mini	
14	A0S	Toyot	
15	0BM	Fiat	
16	0BM	Fiat	
17	0BM	Fiat	
18	P2O	Hyunc	
19	3G8	Suzuk	
20	CFJ	Ford	
21	L51	Saab	Бердск
22	ССК	Saab	12-03-2017

- Теперь, когда все основные трансформации выполнены, можно, наконец, назначить каждому столбцу соответствующие типы данных. Проще всего это сделать щелчком по значку в левом верхнем углу каждого столбца, рядом с треугольником фильтра. Для столбцов **Код заказа**, **Город** и **Бренд** можно оставить текстовый, для столбца **Кол-во** установить **Целое число (Whole number)**, для столбца с датой – формат **Дата (Date)**:



Если у вас не видно этой иконки, то, скорее всего, у вас устаревшая версия Power Query и нужно обновиться. А пока тип данных можно задать, щёлкнув по заголовку столбца правой кнопкой мыши и выбрав команду **Тип изменения (Change Type)**.

Интереснее всего будет ситуация с последней колонкой **Стоимость**, где у нас числа в нестандартном (для «русского» Excel) формате. Для их корректного распознавания нужно выбрать не опцию **Десятичное число (Decimal)**, а последнюю в контекстном меню команду **Используя локаль (Use local)**. Под *локалью* в данном случае понимается набор региональных (локальных) настроек записи чисел, дат и денежных сумм для разных стран. Открывшееся окно позволит выбрать любой нужный тип данных и стандарт (в нашем случае **Английский (США)**), и после нажатия на **ОК** числа будут правильно интерпретированы:



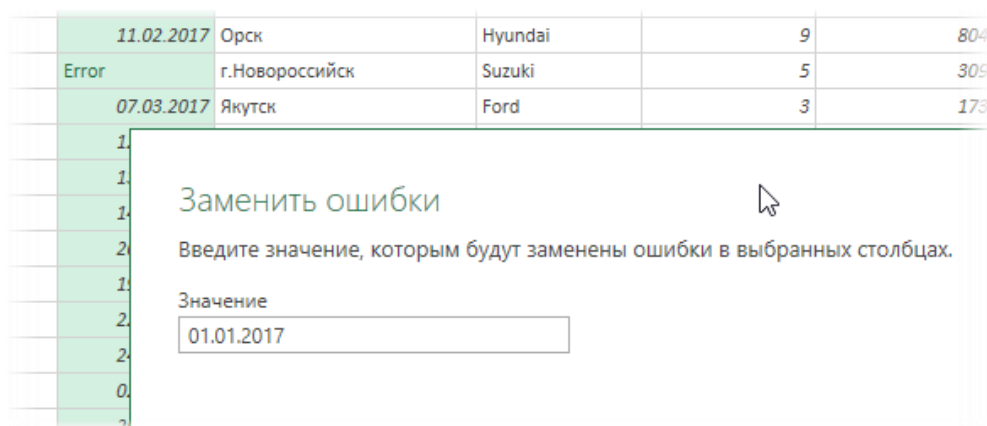
- Обратите внимание, что после преобразования в дату на одной из ячеек появилась ошибка. Чтобы понять ее причину, можно последовательно пощелкать мышью по шагам в панели **Параметры запроса**, отловив момент ее возникновения:

15	P2O	11-02-2017	Орск	→	15	P2O	11.02.2017	Орск
16	3G8	30-02-2017	г.Новорос		16	3G8	Error	г.Новорос
17	CFJ	07-03-2017	Якутск		17	CFJ	07.03.2017	Якутск

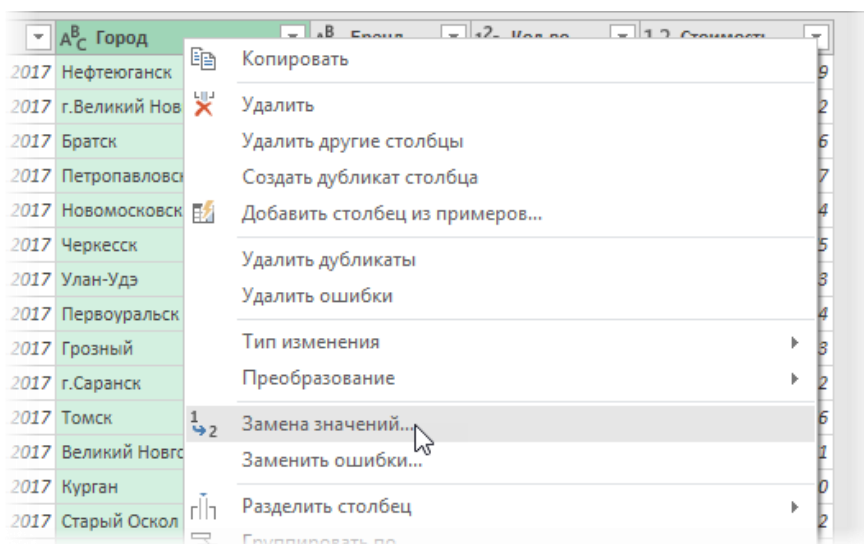
Также можно щёлкнуть мышью в фон ячейки с ошибкой (но не в слово Error) и увидеть описание ошибки в нижней части окна.

Причина банальна: человеческий фактор, как всегда. Вопрос: что делать с получившейся ошибкой? Вариантов три.

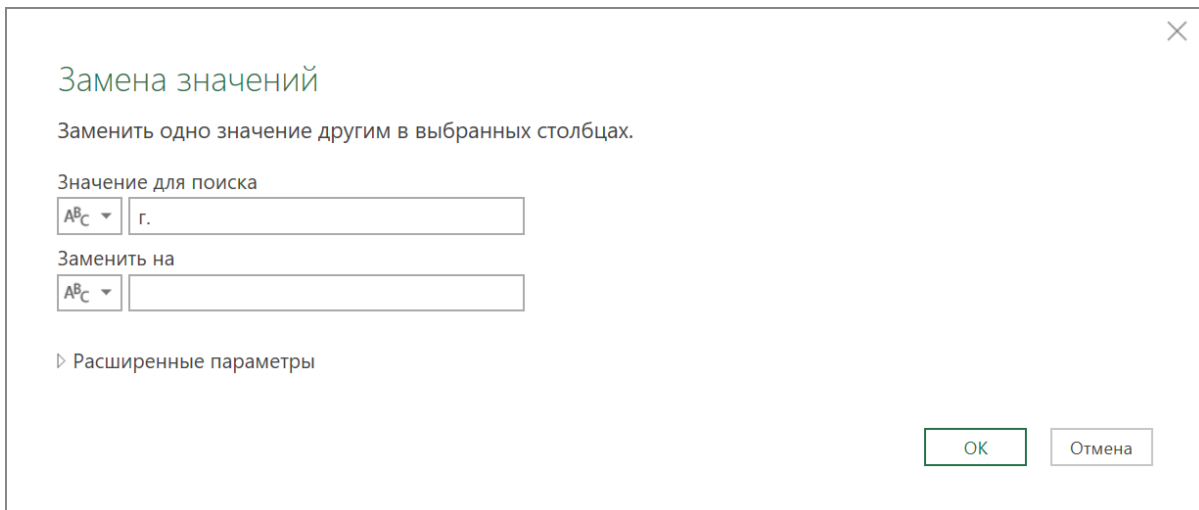
- **Оставить все как есть.** При выгрузке на лист ячейка с **Error** будет пустой, но в процессе обновления мы получим сообщение о найденных ошибках и подробности по ним.
- **Удалить строки с ошибками**, выделив столбец с датой и выбрав на вкладке **Главная** → **Удалить строки** → **Удалить ошибки** (Home → Remove rows → Remove errors).
- **Заменить ошибки** на какое-либо значение, например на 1 января 2017 года или любую другую подходящую дату. Для этого можно щелкнуть правой кнопкой мыши по заголовку столбца даты и выбрать команду **Заменить ошибки** (Replace errors) из контекстного меню:



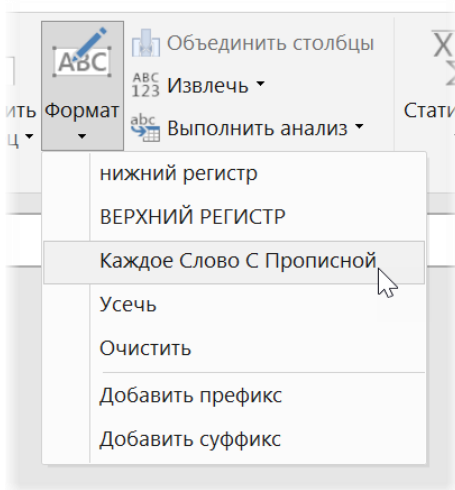
10. Похожим образом можно избавиться, например, и от букв «г» в столбце с городами. Щелкните правой кнопкой мыши по заголовку столбца **Город** и в контекстном меню будут все возможные действия для этой колонки, в том числе и нужная нам команда **Замена значений** (Replace values):



Далее можно ввести значения для поиска и замены и быстро зачистить ненужные символы:



11. И, наконец, можно дополнительно исправить регистр в столбце с городами, выбрав на вкладке **Преобразование** → **Формат** → **Каждое слово с прописной** (Transform → Format → Capitalize Each Word):



На этом этапе приведения в порядок будем считать выполненным и переходим к следующему шагу – выгрузке результатов.

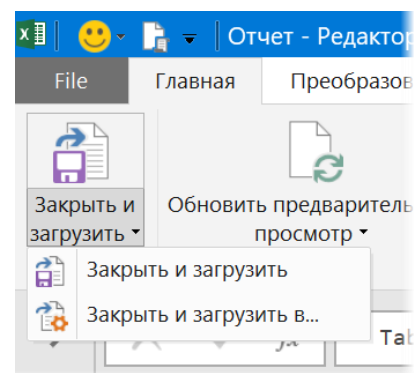
Выгрузка результатов из Power Query обратно в Excel

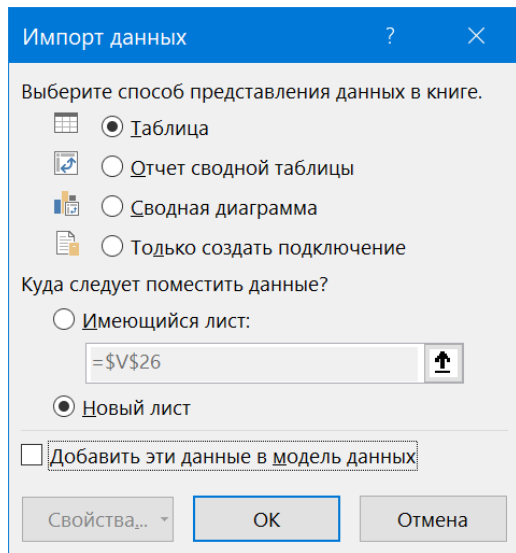
Теперь, когда все операции по «причёсыванию» данных выполнены, полученные результаты надо выгрузить из Power Query обратно в Excel. Для этого служит команда **Закреть и загрузить** на вкладке **Главная** (Home → Close & Load).

Обратите внимание, что в раскрывающемся списке присутствует две команды с похожими названиями, но разным смыслом:

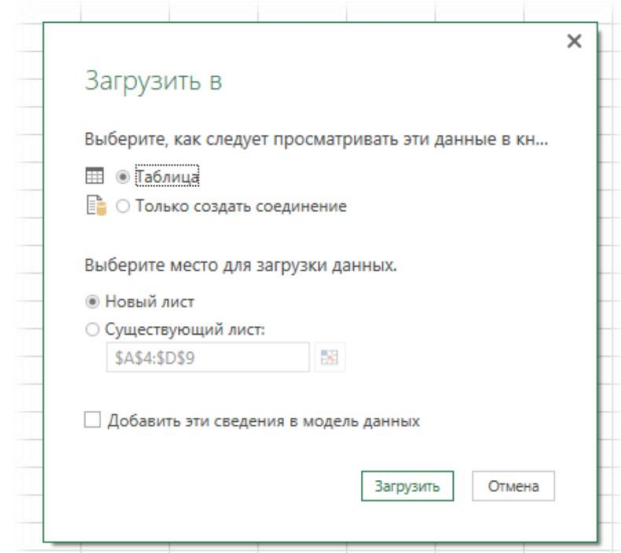
- **Закреть и загрузить** (Close & Load) – закрывает окно Power Query и выгружает все результаты в виде «умной» таблицы на добавленный в текущую книгу новый лист.
- **Закреть и загрузить в...** (Close & Load to...) – выводит диалоговое окно, где можно выбрать, куда и в каком виде мы хотим выгрузить результаты.

Это окно может выглядеть немного по-разному в зависимости от используемой версии Excel.





или так



- **Таблица (Table)** – данные выгружаются в виде «умной» таблицы (на отдельный новый или один из имеющихся листов).
- **Только создать подключение (Create Only Connection)** – данные остаются в памяти и нигде явно не выгружаются. По ним, однако, можно позднее построить сводную таблицу или воспользоваться ими в другом запросе. Этот вариант экономит место на листе и размер файла.
- **Отчет сводной таблицы (Pivot Table)** – то же самое, что и предыдущий вариант, но с построением сводной таблицы по созданному подключению.
- **Сводная диаграмма (Pivot Chart)** – то же самое, что и предыдущий вариант, но рядом со сводной таблицей будет построена еще и сводная диаграмма.
- Флажок **Добавить эти данные в модель данных (Add to Data Model)** загружает результаты в надстройку Power Pivot для дальнейшего анализа и использования. Подробнее с этим вариантом мы разберемся позже в отдельной главе.

Выберем для начала простую выгрузку в виде таблицы на новый лист. После нажатия на **OK** на пустом добавленном листе появится наша «умная» таблица с результатами обработки из Power Query:

	A	B	C	D	E	F	G	H
1	Код заказа	Дата	Город	Бренд	Кол-во	Стоимость		
2	ST5	20.01.2017	Нефтеюганск	Peugeot	6	1849		
3	SW6	30.01.2017	Великий Новгород	Mitsubishi	1	15542		
4	KJ8	25.01.2017	Братск	Renault	8	88846		
5	OZS	04.01.2017	Петропавловск-Камчатский	Mitsubishi	8	97127		
6	U0I	31.01.2017	Новомосковск	Toyota	9	42834		
7	8ZE	05.01.2017	Черкесск	Subaru	6	21575		
8	UR7	09.01.2017	Улан-Удэ	Peugeot	8	24843		
9	LNR	11.02.2017	Первоуральск	Saab	5	69134		
10	KLH	27.02.2017	Грозный	Fiat	2	58933		
11	J6X	17.02.2017	Саранск	Honda	8	9342		
12	H0P	03.02.2017	Томск	Skoda	2	56566		
13	E71	09.02.2017	Великий Новгород	Mini	1	71661		
14	A0S	19.02.2017	Курган	Toyota	6	82200		
15	OBM	16.02.2017	Старый Оскол	Fiat	4	17882		
16	P2O	11.02.2017	Орск	Hyundai	9	80457		
17	3G8	01.01.2017	Новороссийск	Suzuki	5	30952		
18	CFJ	07.03.2017	Якутск	Ford	3	17325		
19	L51	12.03.2017	Бердск	Saab	3	88439		
20	C6K	13.03.2017	Владикавказ	Saab	6	61377		
21	P0O	14.03.2017	Ногинск	Rover	9	59120		
22	5B5	26.03.2017	Копейск	Renault	5	30129		
23	KIU	19.03.2017	Северск	Mitsubishi	7	18253		
24	QID	22.03.2017	Балаково	Renault	9	12067		
25	RUT	24.03.2017	Самара	Nissan	6	43202		
26	30H	02.03.2017	Тюмень	Mazda	9	87039		
27	ORU	26.03.2017	Химки	Mini	8	84127		
28	SGZ	26.03.2017	Евпатория	Saab	3	92263		
29	KVE	29.03.2017	Первоуральск	Peugeot	8	86905		

Запросы и подключения

Запросы | Подключения

1 запрос

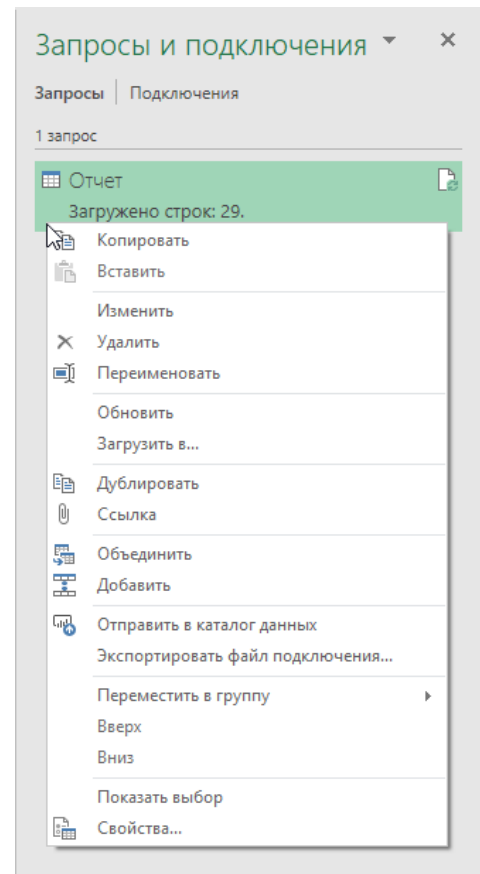
Отчет

Загружено строк: 29.

В правой части окна должна появиться панель **Запросы и подключения** (Queries & Connections), где должен отобразиться наш запрос и количество загруженных строк. Если щелкнуть правой кнопкой мыши по нашему запросу *Отчет* в этой панели, то можно увидеть набор всех доступных нам команд:

Работу с запросами мы еще рассмотрим подробно в следующих главах, а сейчас отметим только самые важные команды из этого списка, а именно:

- **Обновить** (Refresh) – повторяет весь процесс обращения к файлу и последующих трансформаций заново. Если завтра вы получите новый файл с другими данными, то достаточно будет подменить его в исходной папке (с сохранением имени файла, само-собой) и обновить запрос.
- **Загрузить в...** (Load to...) – позволяет еще раз вывести диалоговое окно с вариантами выбора места и типа выгрузки (таблица, сводная, только подключение и т.д.)
- **Переименовать** (Rename) – если вы забыли дать запросу понятное имя в редакторе, то это можно легко сделать тут.
- **Удалить** (Delete) – при удалении запроса удаляются все записанные шаги, но выгруженные на лист результаты останутся на месте (только их нельзя будет больше обновить).



Построение сводной таблицы по результатам запроса

Если на выходе запроса нам нужно получить сводную таблицу, то у нас есть два варианта.

Можно просто выгрузить результаты запроса на лист в виде «умной» таблицы, как мы это делали в предыдущем пункте, а потом вручную построить по ней сводную стандартным образом:

1. Выделить любую ячейку результирующей таблицы запроса.
2. Пойти на вкладку **Вставка** (Insert).
3. Нажать на кнопку **Сводная таблица** (Pivot Table).
4. Проверить, чтобы имя таблицы было в поле источника данных.
5. Нажать на кнопку **OK** и перейти к конструированию сводной, т. е. переносу мышкой полей таблицы в области строк, столбцов, значений и фильтра.

The screenshot shows the Excel interface with the 'Вставка' (Insert) ribbon active. A red circle '1' points to a cell in the data table. Another red circle '2' points to the 'Вставка' ribbon. A third red circle '3' points to the 'Сводная таблица' (PivotTable) button. A dialog box titled 'Создание сводной таблицы' is open, with a red circle '4' pointing to the 'Отчет' field in the 'Выборите данные для анализа' section. A final red circle '5' points to the 'OK' button in the dialog.

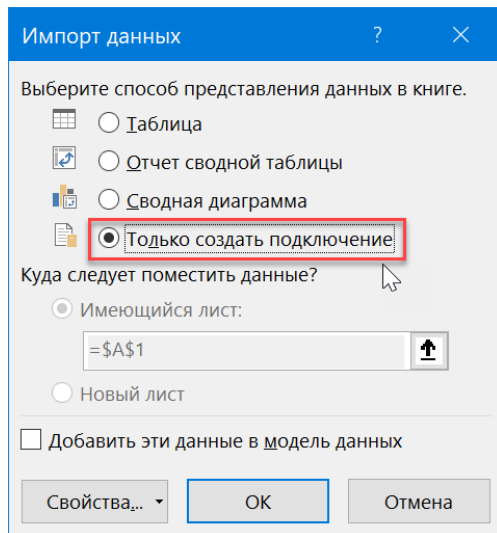
Код заказа	Дата	Город	Бренд	Кол-во	Стоимость
ST5	20.01.2017	Нефтеюганск	Peugeot	6	1849
SW6	30.01.2017	Великий Новгород	Mitsubishi	1	15542
KJ8	25.01.2017	Братск	Renault	8	88846
OZS	04.01.2017	Петрозаводск	Mitsubishi	8	87127
U0I	31.01.2017	Новосибирск	Peugeot	1	15542
8ZE	05.01.2017	Челябинск	Peugeot	1	15542
UR7	09.01.2017	Ульяновск	Peugeot	1	15542
LNR	11.02.2017	Пермь	Peugeot	1	15542
KLH	27.02.2017	Грозный	Peugeot	1	15542
U6X	17.02.2017	Саратов	Peugeot	1	15542
HOI	03.02.2017	Тольятти	Peugeot	1	15542
E71	09.02.2017	Великий Новгород	Peugeot	1	15542
AOS	19.02.2017	Курск	Peugeot	1	15542
OBM	16.02.2017	Ставрополь	Peugeot	1	15542
P2O	11.02.2017	Орел	Peugeot	1	15542
3G8	01.01.2017	Новосибирск	Peugeot	1	15542
CFJ	07.03.2017	Якутск	Peugeot	1	15542
L51	12.03.2017	Белгород	Peugeot	1	15542
S6K	13.03.2017	Владимир	Peugeot	1	15542
P00	14.03.2017	Новосибирск	Peugeot	1	15542
5B5	26.03.2017	Колыбель	Peugeot	1	15542
KIU	19.03.2017	Северск	Mitsubishi	7	18253
QID	22.03.2017	Балаково	Renault	9	12067

Этот классический и очевидный способ, однако, имеет пару недостатков.

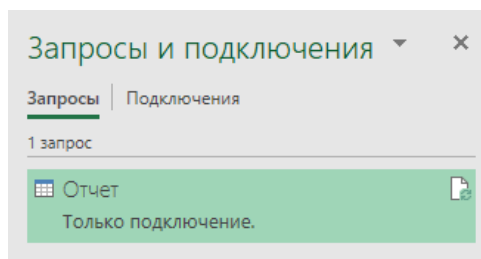
Если в результате запроса получается больше 1 048 576 строк, то результирующая таблица просто **не влезет на лист Excel** и будет обрезана, потеряв часть данных.

Результирующая таблица (даже если она умещается на лист) может быть очень большой, тогда она **заметно утяжелит и замедлит ваш файл**. Причем видеть сами исходные данные – все эти сотни тысяч строк – иногда даже и не требуется! Главное – это построить по ним сводную, которая в конечном счёте и нужна.

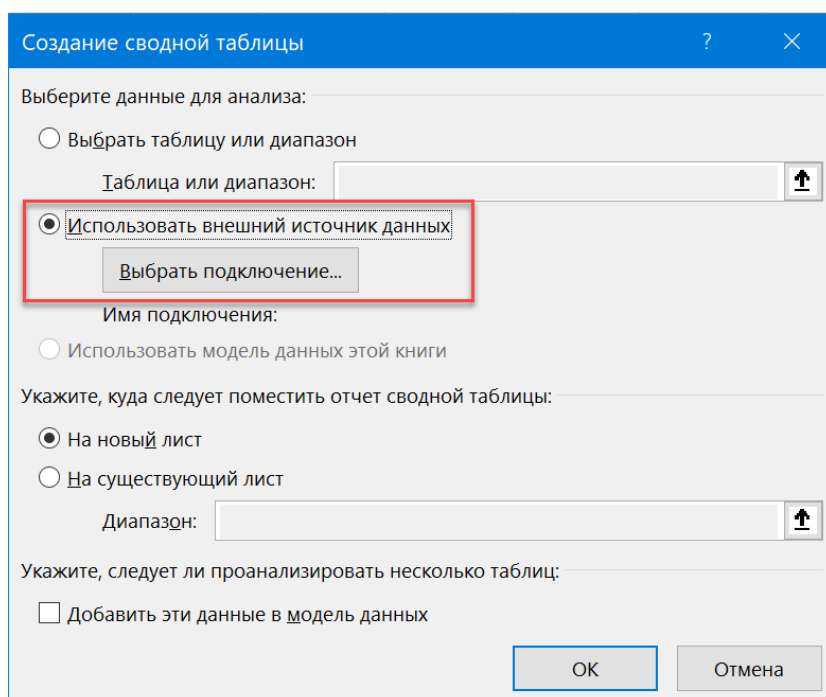
В этом случае разумнее будет сохранить результаты запроса как подключение, выбрав в диалоговом окне после команды **Закреть и загрузить** → **Закреть и загрузить в...** опцию **Только создать подключение** (Create Only Connection):



Тогда данные на лист не выгружаются, а остаются в памяти. На панели запросов такой запрос будет помечен соответствующим образом:



Теперь можно создать новую пустую сводную таблицу через **Вставка** → **Сводная таблица**, а затем в следующем окне выбрать опцию **Использовать внешний источник данных** (Use external data source) и нажать кнопку **Выбрать подключение** (Choose connection):

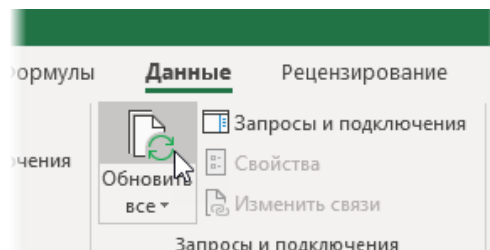


Тогда в следующем окне нужно будет выбрать созданное ранее подключение и построить потом сводную таблицу стандартным образом, но по данным, находящимся в памяти компьютера, а не явно на листе. Такой способ заметно облегчает и ускоряет файл, особенно при большом количестве данных.

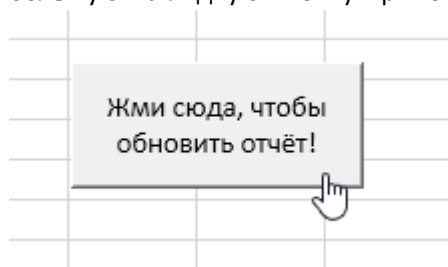
Обновление запросов

В отличие от обычных формул Microsoft Excel запросы Power Query не обновляются автоматически, нужно делать это вручную. Для этого есть несколько способов.

- Щелкнуть правой кнопкой мыши по запросу в правой панели **Запросы и подключения** и выбрать команду **Обновить (Refresh)**.
- Та же команда **Обновить** будет доступна, если щелкнуть правой кнопкой мыши по самой таблице, полученной в результате запроса (но не по сводной, построенной по результатам!).
- Если у вас в книге работает несколько запросов (возможно, связанных между собой) и сводных таблиц, построенных по их результатам, то часто удобнее обновить сразу всё оптом. Для этого можно на вкладке **Данные** нажать кнопку **Обновить всё (Refresh All)** или сочетание клавиш **Ctrl+Alt+F5**.



В некоторых ситуациях бывает удобнее производить обновление макросом, назначив его, например, на большую наглядную кнопку прямо на листе, которую уже никак не пропустит начинающий пользователь:

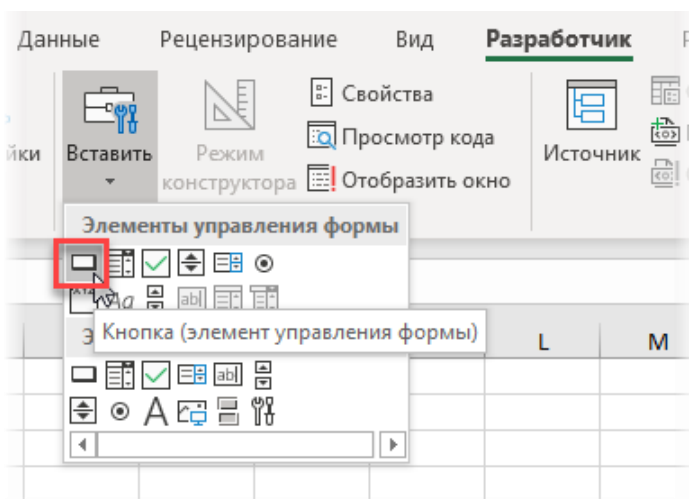


Чтобы реализовать такое, нужно:

1. Пересохранить книгу в формате с поддержкой макросов (xlsm).
2. Нажать сочетание клавиш **Alt+F11**, чтобы открыть редактор Visual Basic.
3. Добавить в редакторе новый пустой модуль для кода через меню **Insert → Module**.
4. Набрать с клавиатуры в модуль код макроса из трех строчек, обновляющий все запросы:

```
Sub Update_All()
    ActiveWorkbook.RefreshAll
End Sub
```

5. Вернуться в Microsoft Excel.
6. Добавить на лист кнопку с помощью команды **Вставка – Кнопка (Insert – Button)**:



7. Назначить на кнопку наш макрос **Update_All** в появившемся диалоговом окне или щелкнув правой кнопкой мыши по нарисованной кнопке и выбрав команду **Назначить макрос** (**Assign Macro**).

Для профессиональной работы со сложными моделями данных также можно использовать отдельную программу **Power Update** (<http://poweronbi.com/power-update/>). Она умеет автоматически обновлять запросы Power Query и Power Pivot, делать обновления по расписанию, параллельную загрузку данных из разных источников и многое другое, но не бесплатна, к сожалению (хотя есть временная бесплатная trial-версия).

Исходный код запроса на языке M

У нея внутри неонка!

(Аркадий и Борис Стругацкие, «Сказка о Тройке»)

Важно понимать, что все действия (шаги), которые мы воспроизводим в окне редактора запросов Power Query, на самом деле записываются внутри него в виде команд на специальном встроенном языке, который лаконично называется M («эм»). Заглянуть «под капот» и увидеть исходный код запроса можно, нажав в окне Power Query на вкладке **Вид** кнопку **Расширенный редактор** (View → Advanced Editor).

Вот так, например, выглядит код нашего запроса, который мы создали для импорта данных из текстового файла **Отчет.txt** из прошлой главы:

```

let
    Источник = Csv.Document(File.Contents("C:\Users\pavlo\OneDrive\Проекты\Книга СД\Примеры\Отчет.txt"),[Delimiter=";", Columns=6, Encodi
    #"Удаленные верхние строки" = Table.Skip(Источник,4),
    #"Повышенные заголовки" = Table.PromoteHeaders(#"Удаленные верхние строки", [PromoteAllScalars=true]),
    #"Удалены пустые строки" = Table.SelectRows(#"Повышенные заголовки", each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValue
    #"Строки с примененным фильтром" = Table.SelectRows(#"Удалены пустые строки", each not Text.Contains([Код заказа], "---") and not Tex
    #"Удаленные дубликаты" = Table.Distinct(#"Строки с примененным фильтром", {"Код заказа"}),
    #"Измененный тип" = Table.TransformColumnTypes(#"Удаленные дубликаты",{"Дата", type date}, {"Кол-во", Int64.Type}),
    #"Измененный тип с языком" = Table.TransformColumnTypes(#"Измененный тип", {"Стоимость", type number}, {"en-US"},
    #"Замененные ошибки" = Table.ReplaceErrorValues(#"Измененный тип с языком", {"Дата", #date(2017, 1, 1)}),
    #"Замененное значение" = Table.ReplaceValue(#"Замененные ошибки", "г.", "", Replacer.ReplaceText, {"Город"}),
    #"Выполнена капитализация каждого слова" = Table.TransformColumns(#"Замененное значение",{"Город", Text.Proper, type text})
in
    #"Выполнена капитализация каждого слова"
  
```

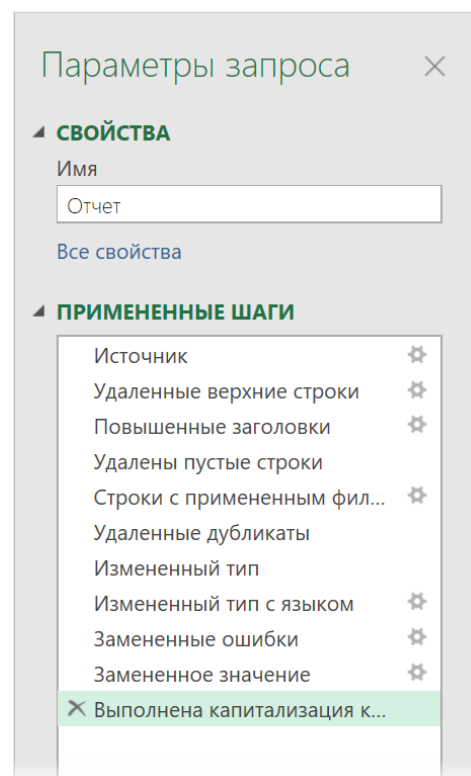
✓ Синтаксические ошибки не обнаружены.

Готово Отмена

С непривычки выглядит жутковато, да. Однако, если присмотреться, то вы заметите, что каждая строка в этом коде – это соответствующий шаг в правой панели редактора запросов **Примененные шаги** (Applied Steps).

Причем каждая строка – это, по сути, функция, которая берет результат предыдущего шага и как-то его трансформирует, передавая, в свою очередь, выходные данные на следующий шаг.

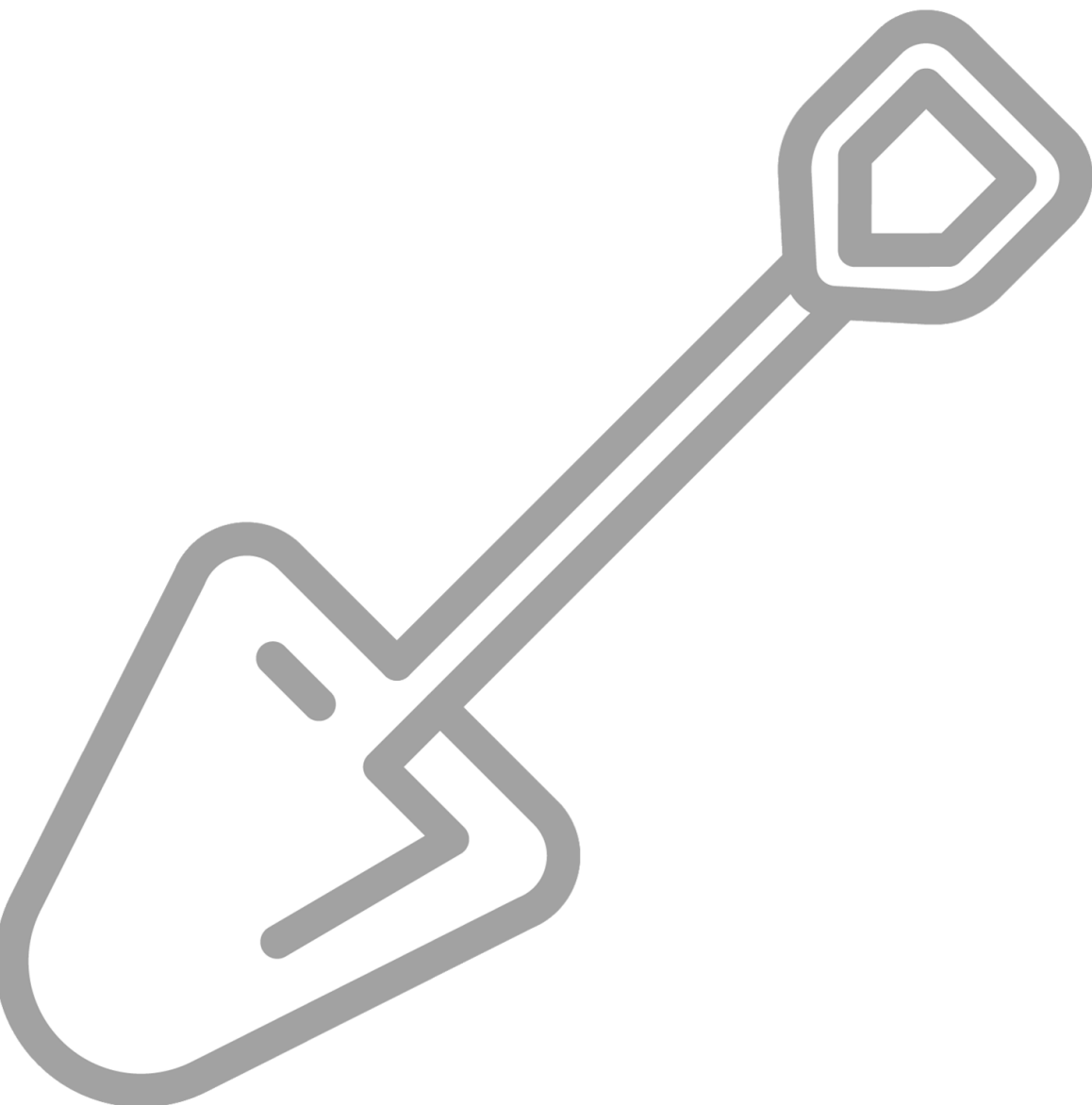
На начальном этапе освоения Power Query заходить в это окно придется нечасто, но чем дальше вы будете продвигаться, тем чаще будете оказываться в ситуациях, когда небольшая ручная правка кода будет выручать вас в сложных ситуациях. Ближе к концу этой книги мы посвятим работе с кодом несколько глав, но сейчас будет вполне достаточно общего понимания, чтобы не перегрузить мозг лишними деталями уже на старте.



Загрузка данных в Power Query

В этой главе мы рассмотрим все основные способы загрузки исходных данных в Power Query, а именно:

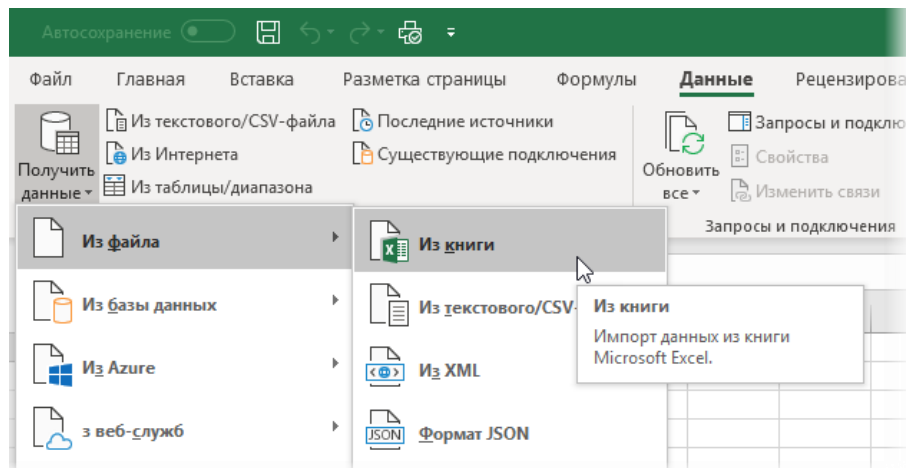
- какие объекты Power Query может импортировать из **книг Excel** и как это сделать;
- как загружать данные с **веб-страниц** из интернета;
- как подключать Power Query к **базам данных** и вытягивать информацию оттуда;
- разберём загрузку из общепринятых **форматов обмена данными XML и JSON**;
- попробуем импортировать таблицу даже из **PDF-файла** (через Word).



Загрузка данных из внешней книги Excel

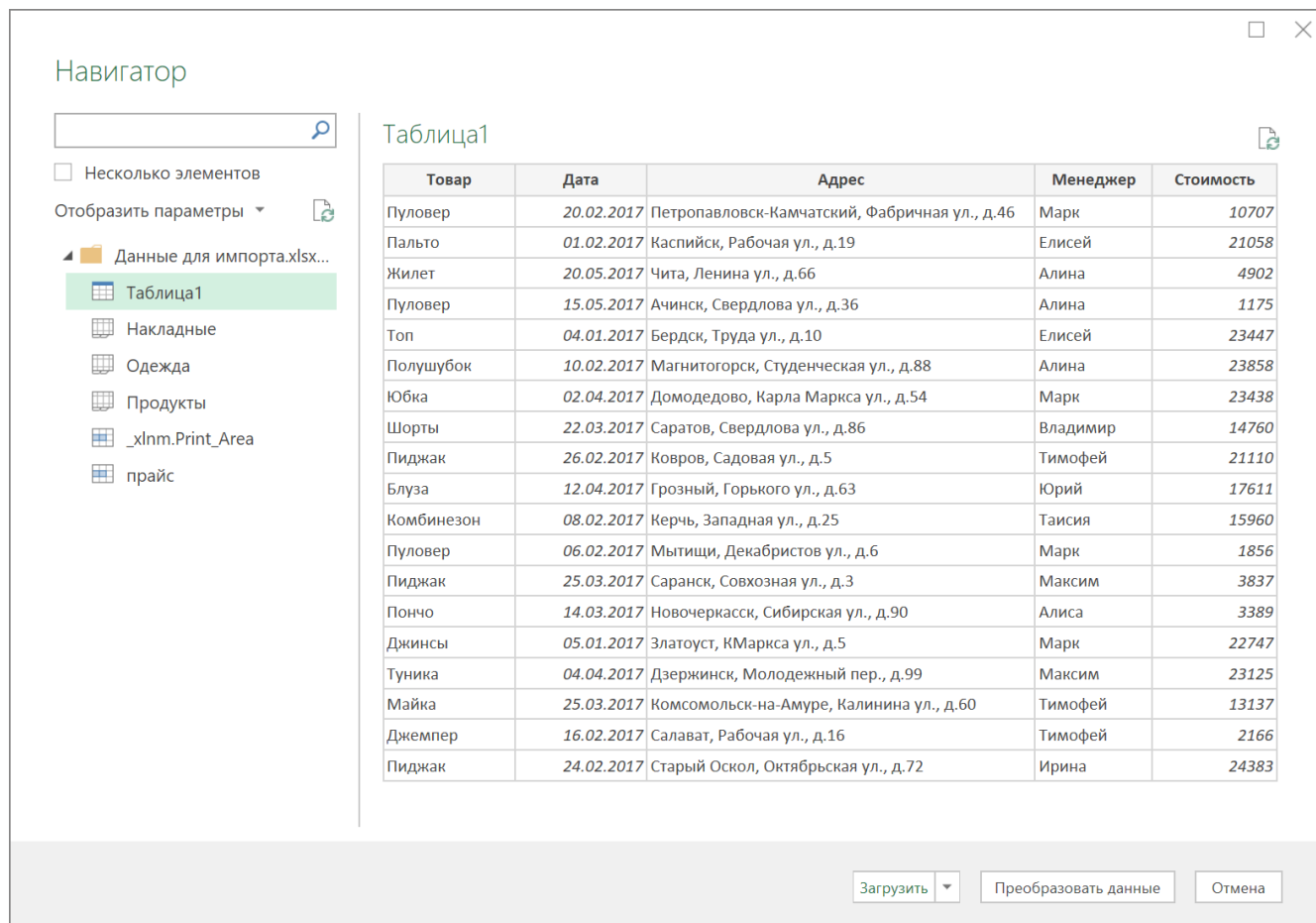
Это один из самых распространенных сценариев: вам надо загрузить в текущую книгу какие-то данные из другого файла Excel, расположенного где-то вовне (на вашем компьютере или в корпоративной сети).

Для выполнения такой задачи выберем на вкладке **Данные** команду **Получить данные** → **Из файла** → **Из книги** (Get Data → From file → Excel):






Помните о том, что в некоторых версиях Excel кнопка **Получить данные** может называться **Создать запрос** (New Query).

Затем нужно будет выбрать файл (я использую файл **Данные для импорта.xlsx** в папке с примерами к этой книге), и потом появится окно **Навигатора** (Navigator):



Обратите внимание на левую половину этого окна: там хорошо видно, какие именно объекты Power Query может в принципе импортировать из других книг Excel. Для каждого объекта используется свой значок, и у каждого есть свои особенности. Давайте пройдемся по ним поподробнее:

Листы

-  Накладные
-  Одежда
-  Продукты

Тут всё просто: загружается всё содержимое листа целиком. Если на листе есть какие-то дополнительные ненужные данные, то их придется потом отфильтровать/удалить.

«Умные» таблицы

-  Таблица1


Этим неофициальным, но уже устоявшимся термином обозначаются диапазоны, конвертированные в *Таблицы* с помощью команд **Главная → Форматировать как таблицу** (Home → Format as Table) или **Вставка → Таблица** (Insert → Table) или с помощью сочетания клавиш **Ctrl+T**.

Пожалуй, главная выгода от такого преобразования – это автоматическая подстройка размеров таблицы при добавлении-удалении данных в будущем. Кроме этого, «умные» таблицы имеют много других плюсов:

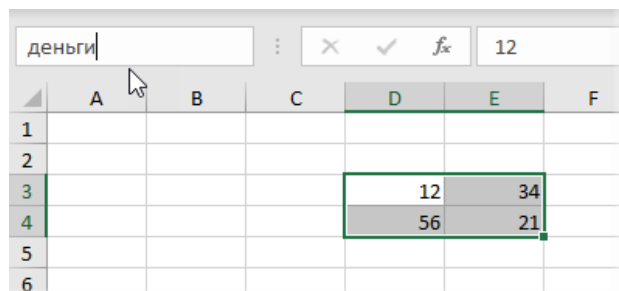
- возможность быстрого форматирования;
- мгновенное «протягивание» формул на весь столбец;
- автоматическое закрепление строки заголовка;
- быстрое добавление строки итогов;
- удобная фильтрация с помощью срезов и т. д.

Более подробно о таких таблицах можно почитать в моей предыдущей книге «Microsoft Excel: Готовые решения – бери и пользуйся» или на сайте на странице <https://www.planetaexcel.ru/techniques/2/136/>.

Именованные диапазоны

-  прайс

Любой ячейке или диапазону ячеек в Excel можно дать собственное имя, или, как ещё говорят, создать *именованный диапазон*. Самый простой способ – это сделать (выделить) требуемую ячейку(и), ввести имя в поле адреса в строке формул (там, где обычно отображается адрес текущей ячейки) и нажать клавишу **Enter**:



	A	B	C	D	E	F
1						
2						
3				12	34	
4				56	21	
5						
6						

Другой способ – это использовать **Диспетчер имен** с вкладки **Формулы** (Formulas → Name Manager). С его помощью имена можно не только создавать, но и редактировать, удалять и т. д.

В отличие от «умных» таблиц именованные диапазоны не растягиваются автоматически при дописывании к ним данных (если только вы не вставляете строки или столбцы в середину диапазона).

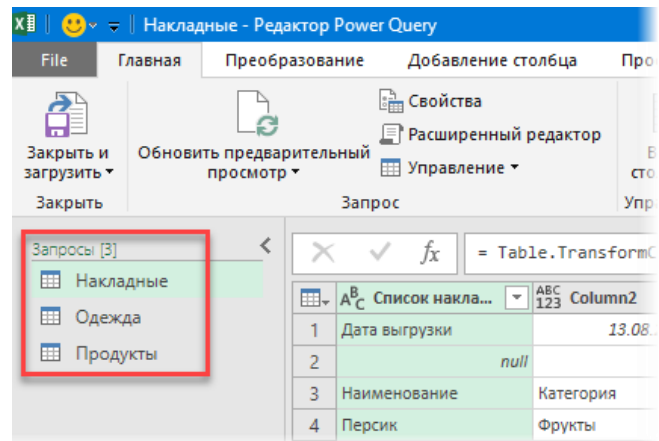
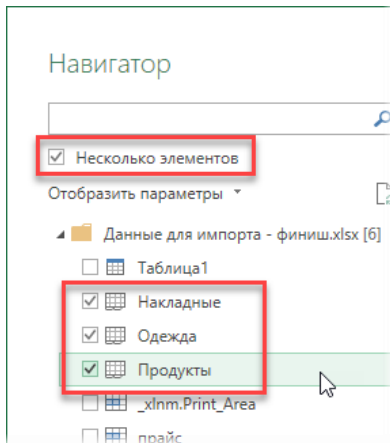
Области печати

-  _xlnm.Print_Area

Кроме всего вышеперечисленного, Power Query видит также и области печати, задаваемые на вкладке **Разметка страницы → Область печати → Задать** (Page Setup → Print Area → Set). В некоторых случаях такой вариант оказывается удобнее, чем остальные.

Кроме всего прочего, в окне **Навигатора** есть флажок **Несколько элементов** (Multiple items), включение которого позволяет выбрать для импорта более чем один объект. Например, можно выделить сразу несколько

листов. Тогда для каждого из них будет сделан отдельный запрос, и все они отобразятся в левой панели редактора запросов Power Query, где между ними можно будет удобно переключаться:



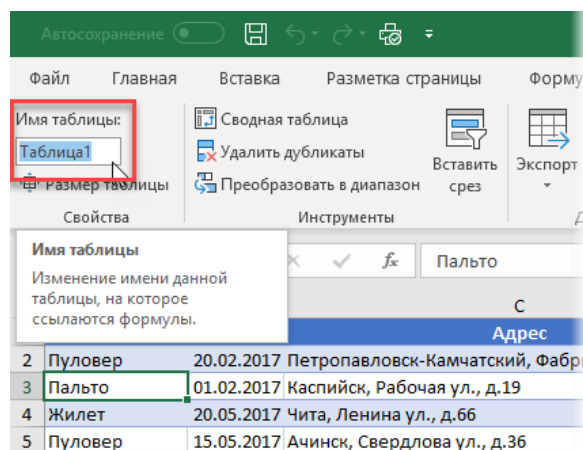
Загрузка данных из текущей книги Excel

Рассмотрим еще один очень частый сценарий: нужные нам данные расположены не где-то снаружи (в другом файле, базе данных, интернете), а прямо тут – в текущей книге Microsoft Excel. И эти данные мы хотим загрузить в Power Query и как-то обработать, почистить, трансформировать и т. д. В такой ситуации у нас есть несколько способов.

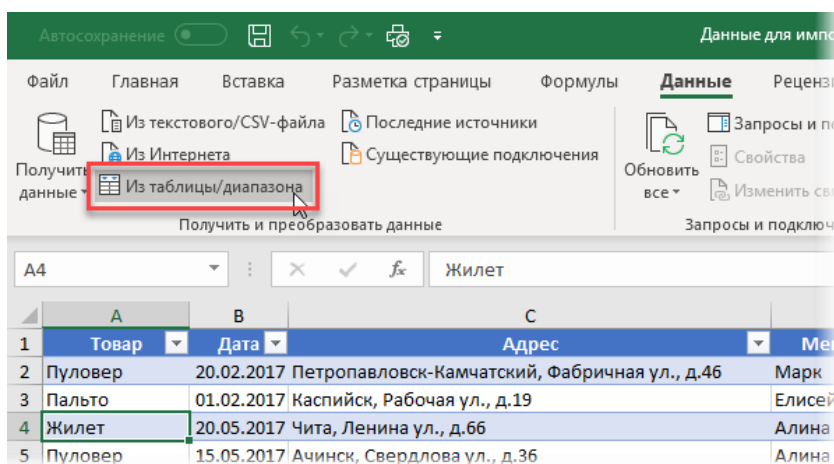
«Умная» таблица

Это один из самых распространенных и удобных вариантов загрузки данных в Power Query:

1. Выделяем диапазон с нужными данными и жмём сочетание клавиш **Ctrl+T** или кнопку **Форматировать как таблицу** на вкладке **Главная** (Home → Format as Table). Диапазон конвертируется в так называемую «умную» таблицу, которой можно (очень желательно!) дать понятное имя на появившейся вкладке **Конструктор** (Design).



2. Затем созданную таблицу можно мгновенно загрузить в Power Query, используя кнопку **Из таблицы/диапазона** (From Table / Range) на вкладке **Данные** (Data):



В принципе, первый шаг в этом алгоритме не обязателен: если вы не преобразуете выделенный диапазон в «умную» таблицу заранее, то это произойдет автоматически. Но тогда и имя таблице будет присвоено тоже стандартное (вида *Таблица1,2,3...*), что создает путаницу даже в несложных моделях. Причем попытка переименовать таблицу уже после загрузки в Power Query приведет к ошибке, т. к. в запросе будет использоваться первоначальное имя, и это придется корректировать вручную дополнительно. Так что лучше сначала превратить таблицу в «умную», потом дать ей имя, и только потом загружать в Power Query – так надежнее.

Большой плюс данного способа состоит в том, что «умные» таблицы имеют свойство автоматически растягиваться и сжиматься, корректируя свои размеры под реальное количество данных. То есть если в будущем дописать к нашей таблице несколько новых строк и/или столбцов, то достаточно будет просто обновить наш запрос, чтобы эти изменения попали в Power Query.

Именованный диапазон

В некоторых случаях превращение диапазона в «умную» таблицу может быть нежелательно. Например, у таких таблиц должна быть строго однострочная правильная шапка и не должно быть объединенных ячеек, что не всегда имеет место. Или может мешать тот факт, что формулы в «умных» таблицах автоматически распространяются сразу на весь столбец и т. д. Тогда для загрузки данных в Power Query можно использовать другой подход:

1. Выделяем нужные ячейки и даём им имя, превращая их в *именованный диапазон*. Это можно сделать с помощью **Диспетчера имен** на вкладке **Формулы (Formulas → Name manager)** или просто вписав имя диапазона в поле адреса в левой части строки заголовка и нажав на **Enter**:

The screenshot shows an Excel spreadsheet with a table of employee data. The table has columns for employee names and months (January, February, March). The named range 'отгрузки' is applied to the data cells. The formula bar shows the name 'Сотрудник'.

	A	B	C	D	E	F
1						
2		Сотрудник	январь	февраль	март	
3		Татьяна	61	88	63	
4		Ольга	67	83	98	
5		Варвара	32	82	95	
6		Иван	15	46	70	
7		Владислав	68	10	48	
8		Максим	33	4	55	
9						

Имя должно начинаться с буквы (не с цифры), не должно содержать пробелов (можно их заменить на подчеркивание) и не должно быть похоже на адрес ячейки (т. е. нельзя дать имя «A1», например).

2. Затем жмём на ту же кнопку **Из таблицы / диапазона (From Table / Range)** с вкладки **Данные**, которую использовали в предыдущем пункте. Именованный диапазон будет моментально загружен в окно редактора запросов Power Query.

Принципиальная разница с предыдущим способом здесь в том, что именованный диапазон не умеет автоматически расширяться при дописывании к нему новых строк и столбцов. Единственный вариант – это вставлять их в середину, раздвигая уже имеющиеся строчки или столбцы.

Универсальный способ с функцией Excel.CurrentWorkbook

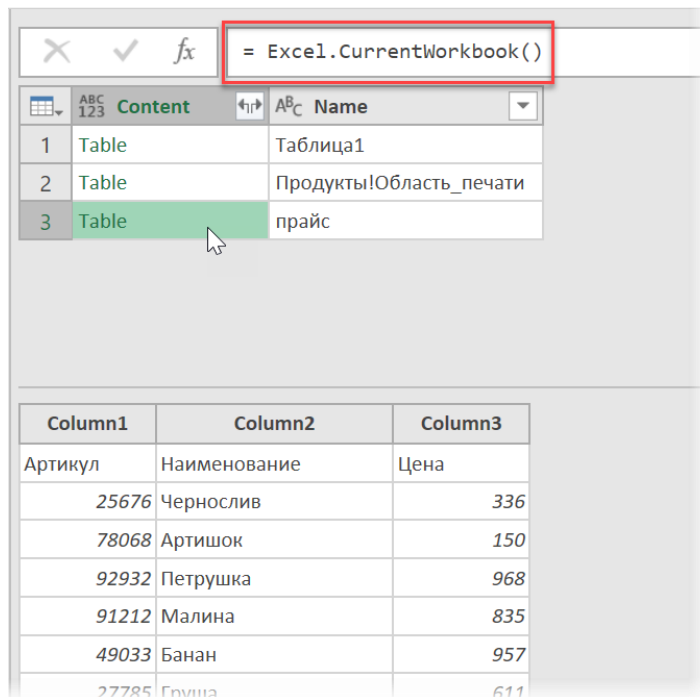
Ещё один способ загрузить данные из текущей книги Excel в Power Query – это использовать специальную встроенную функцию языка M, которая открывает доступ ко всему содержимому текущей книги.

Для этого выберите на вкладке **Данные → Получить данные → Из других источников → Пустой запрос (Data → Get Data → From other sources → Blank query)** и в строку формул в открывшемся редакторе введите следующую функцию:

```
=Excel.CurrentWorkbook()
```

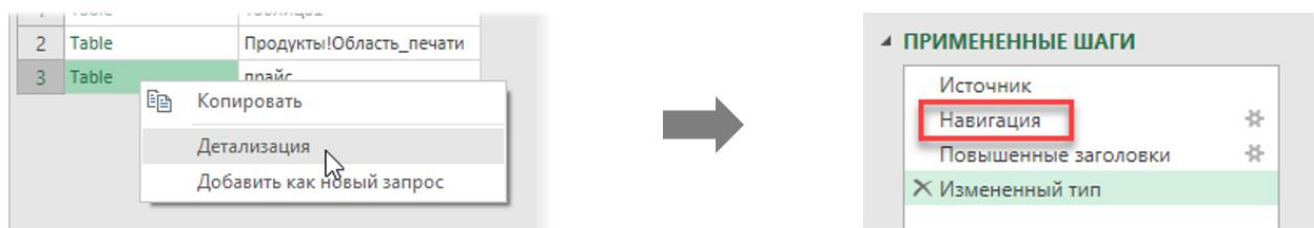
Хочу еще раз предупредить, что вводить любые формулы и функции нужно строго с соблюдением прописных и строчных букв, т. к. язык M является регистрочувствительным. Если строки формул не видно, то её можно включить на вкладке **Вид → Строка формул (View → Formula Bar)**.

После ввода и нажатия на **Enter** мы должны увидеть таблицу с содержимым текущей книги. В ней отображаются все «умные» таблицы (*Таблица1*), области печати и все именованные диапазоны (*прайс*, *отгрузки*), но, к сожалению, нет листов. Просмотреть содержимое любого из перечисленных объектов можно, если щелкнуть мышью в белый фон соответствующей ячейки в столбце **Content**:



Если же щелкнуть мышью не в фон ячейки, а прямо по слову **Table**, то мы, как иногда говорят, «провалимся» в выбранную таблицу и доберёмся, таким образом, до нужных нам данных. Это же действие можно выполнить, если щелкнуть правой кнопкой мыши по ячейке и выбрать команду **Детализация** (Drill Down).

Что интересно, этот нырок в данные тоже будет зафиксирован в правой панели Power Query как шаг **Навигация** (Navigation) и впоследствии может быть легко отменен (удален), если мы передумаем:



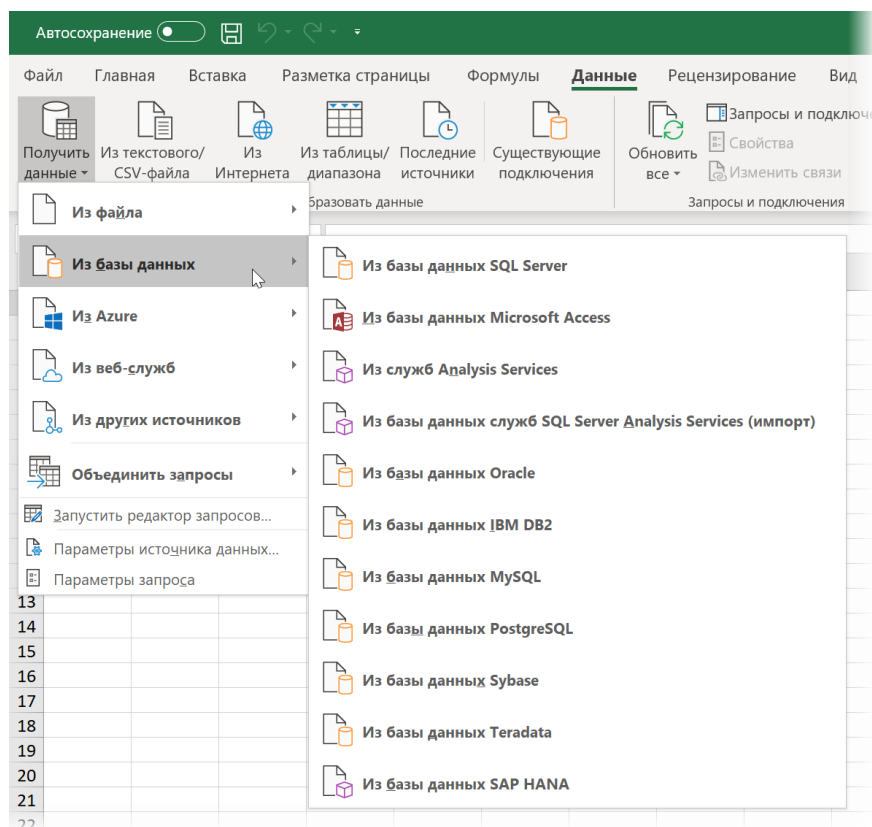
Как вы уже, наверное, понимаете, это универсальный способ, хотя он и требует небольшого погружения в М-код.

Кроме того, у этого варианта есть еще один нюанс: функция `Excel.CurrentWorkbook` выводит не только статические именованные диапазоны (т. е. те, которые мы создали напрямую, дав имена ячейкам), но и динамические¹ (т. е. созданные с помощью формул). Такие диапазоны будут автоматически растягиваться, как «умные» таблицы при дописывании к ним новых данных. Так что если «умные» таблицы вы по каким-либо причинам использовать не хотите, но автоподстройка размеров вам нужна, то это хороший вариант.

¹ Способы создания таких диапазонов я подробно разобрал в своей прошлой книге «Мастер Формул» и выкладывал на сайте (вместе с видеоуроком) на странице <https://www.planetaexcel.ru/techniques/2/219/>.

Подключение к базам данных

При необходимости Power Query легко подключить почти к любой существующей базе данных или корпоративной ERP-системе на её основе. За это на вкладке **Данные** отвечает команда **Получить данные → Из базы данных (Data → Get Data → From Database)**:

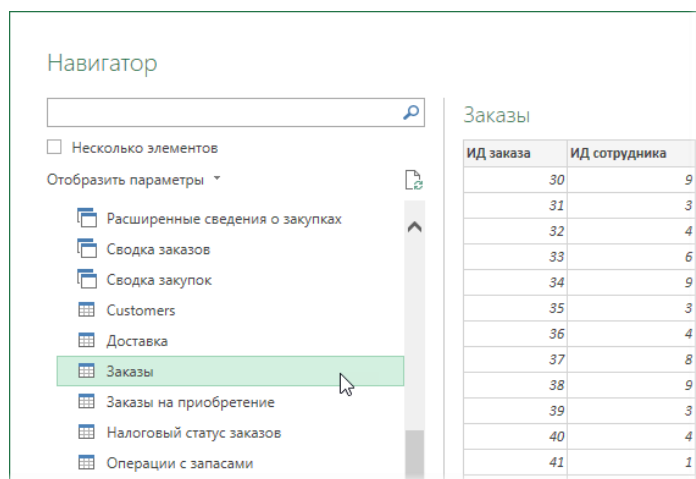


Выбор здесь предоставляется, как видите, большой: есть коннекторы ко всем современным БД, и не так давно (в июле 2017 года) был добавлен даже коннектор к известной ERP-системе SAP. Вне зависимости от типа конкретной БД, которая будет у вас, процесс подключения выглядит примерно одинаково. Давайте рассмотрим его на примере подключения к БД Access (файл **BoreyDatabase.accdb** из папки с примерами к этой книге).

После выбора опции **Из базы данных Microsoft Access (From Access Database)** в приведенном выше списке появится окно выбора файла и затем уже знакомый нам **Навигатор**, где будет отображено всё содержимое выбранной БД.

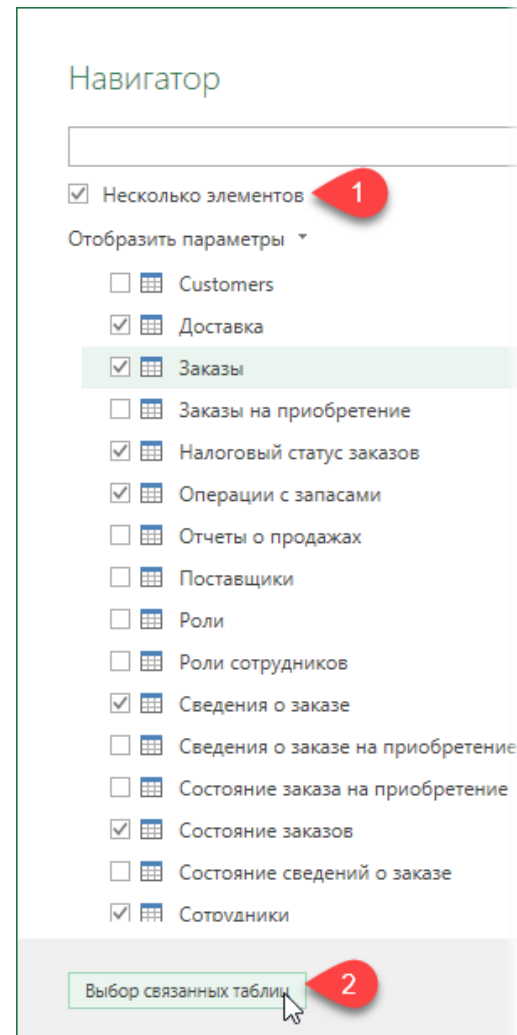
Обратите внимание, что в левой части в списке видны два типа объектов – таблицы и запросы.

- **Таблицы** обозначаются характерным значком и представляют собой исходные таблицы БД с содержащимися в них данными.
- **Запросы** обозначаются значком двойного окошка с синим верхом и представляют собой, по сути, компиляцию данных из нескольких таблиц с определенной фильтрацией и сортировкой.



Хотя технически это два очень разных типа объектов, Power Query может работать с ними совершенно спокойно, воспринимая их оба как простые таблицы. Так что нам с вами останется только выбрать нужные и нажать на кнопку **Изменить (Edit)** или **Преобразовать данные (Transform Data)** в правом нижнем углу окна, чтобы загрузить их в Power Query.

Обычно в любой БД таблицы связаны между собой по значениям ключевых полей (столбцов). Например, в нашей БД таблица **Заказы** имеет связи с таблицами **Доставка**, **Сотрудники**, **Операции с запасами** и др. Чтобы загрузить для анализа сразу все связанные таблицы, нужно выделить исходную таблицу (Заказы), потом включить флажок **Несколько элементов (Multiple Items)** и нажать на кнопку **Выбор связанных таблиц (Select related tables)**. Power Query запросит у БД связи, автоматически пометит флажками все таблицы, связанные с текущей, и создаст отдельные запросы для каждой из них.



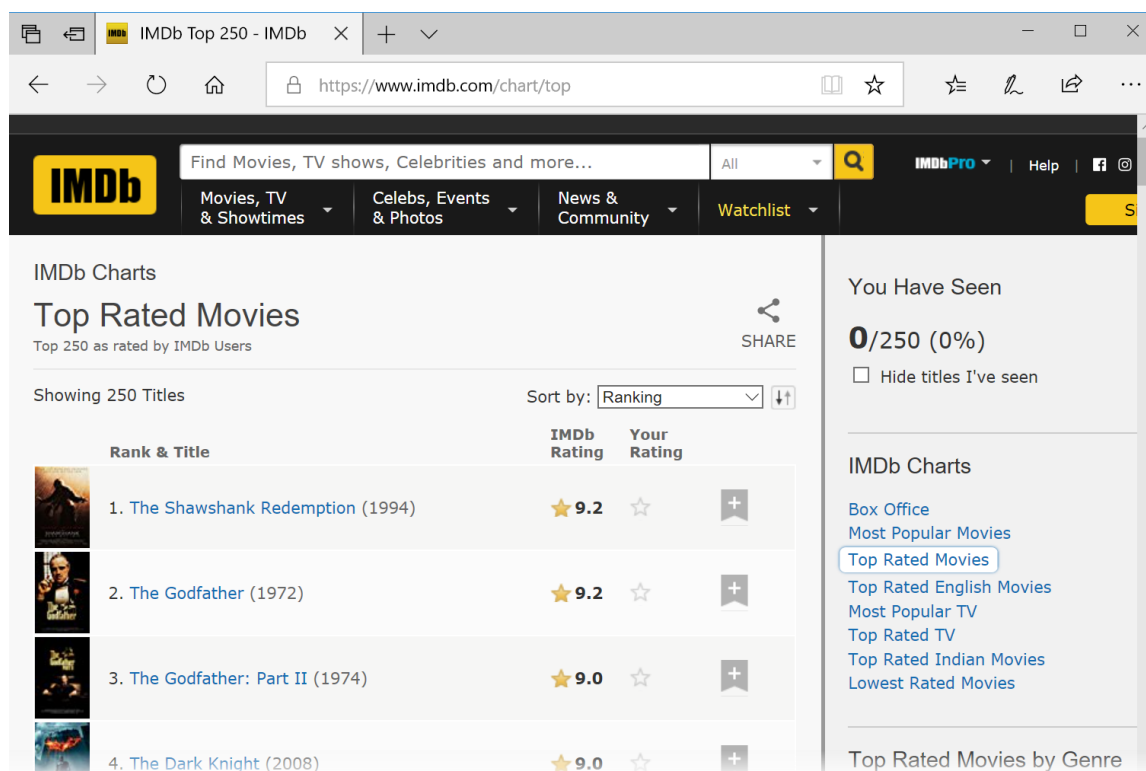
Загрузка данных из интернета

*Я попытался читать Диккенса после Твиттера и, кажется, потянул мышцу в мозгу.
(Хью Лори)*

Импорт данных с веб-страниц

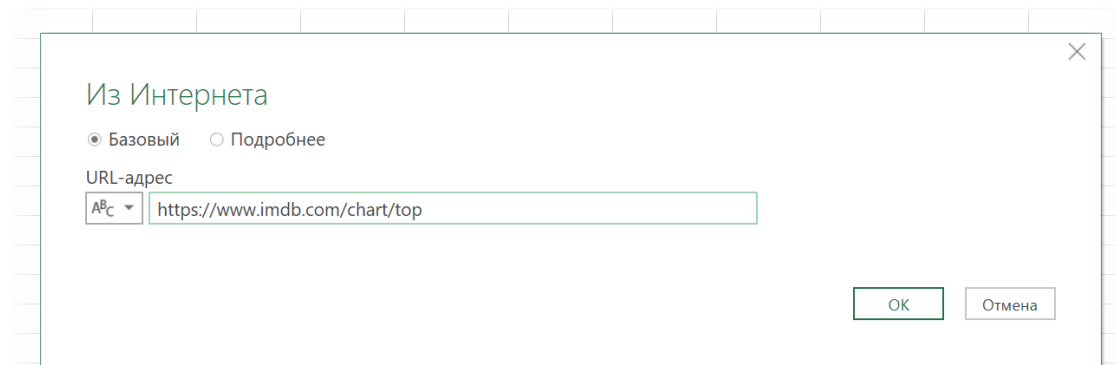
Microsoft Excel поддерживает импорт данных с веб-страниц сайтов уже достаточно давно¹. Однако эта встроенная функциональность была весьма скромной и требовала определенных «танцев с бубном». Power Query справляется с этой задачей гораздо легче.

Допустим, мы хотим загрузить в Excel таблицу с Топ-250 фильмов с сайта www.imdb.com. Соответствующая страница на сайте имеет адрес <https://www.imdb.com/chart/top> и выглядит следующим образом:



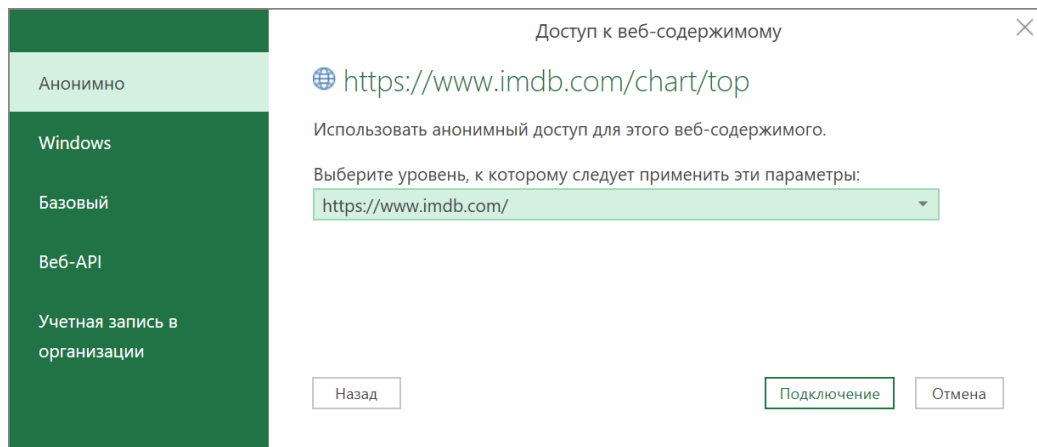
Для импорта нажмем в Microsoft Excel на вкладке **Данные** на кнопку **Из Интернета** (Data → From Internet) или выберем там же **Данные** → **Получить данные** → **Из других источников** → **Из интернета** (Data → Get Data → From Other sources → From web)

Затем введем (скопируем и вставим) адрес нужной веб-страницы в появившееся окно:

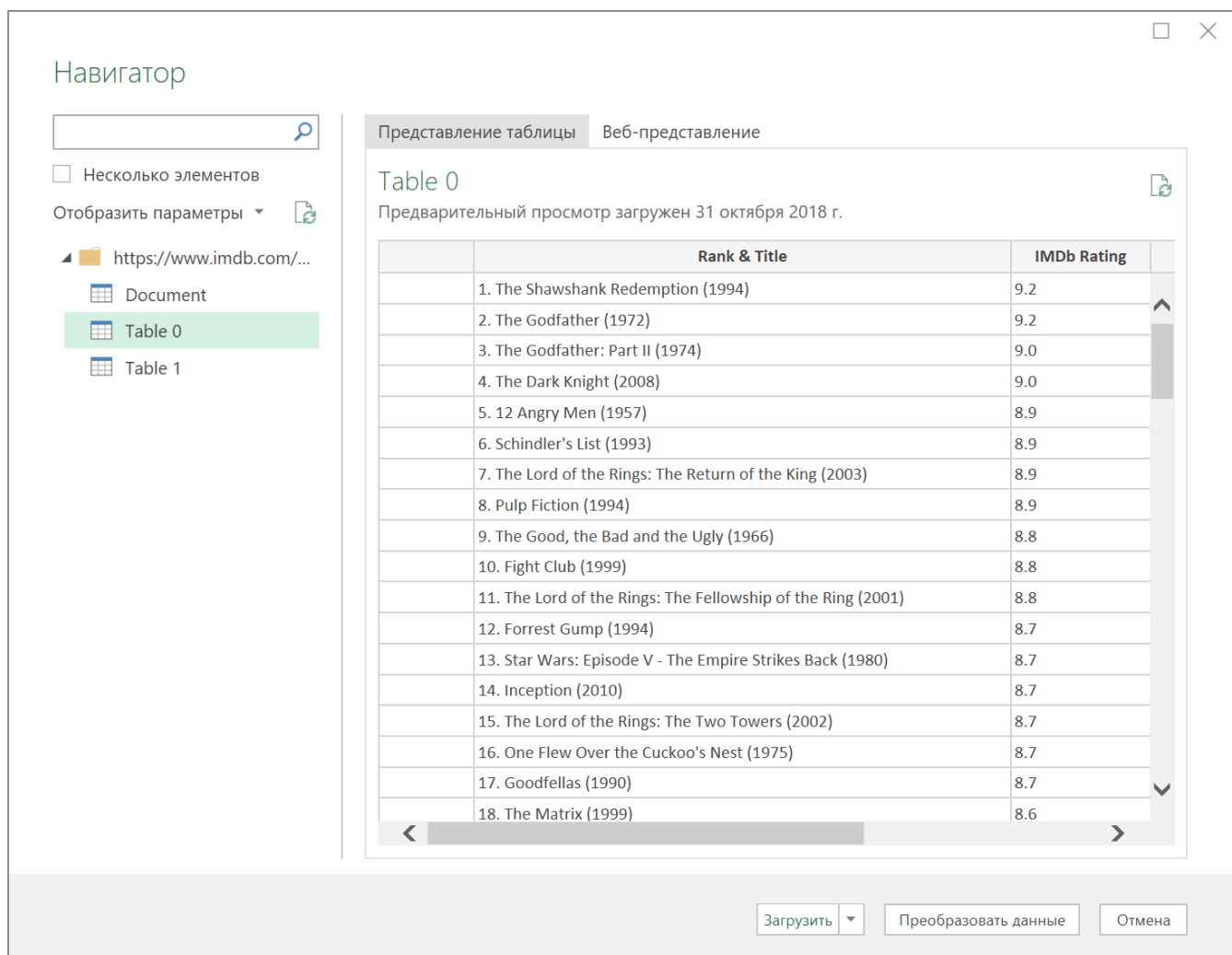


¹ Подробнее об этом в моей книге «Microsoft Excel: Готовые решения – бери и пользуйся!» в главе «Импорт курса валют из интернета» и на моём сайте на странице <https://www.planetaexcel.ru/techniques/13/129/>.

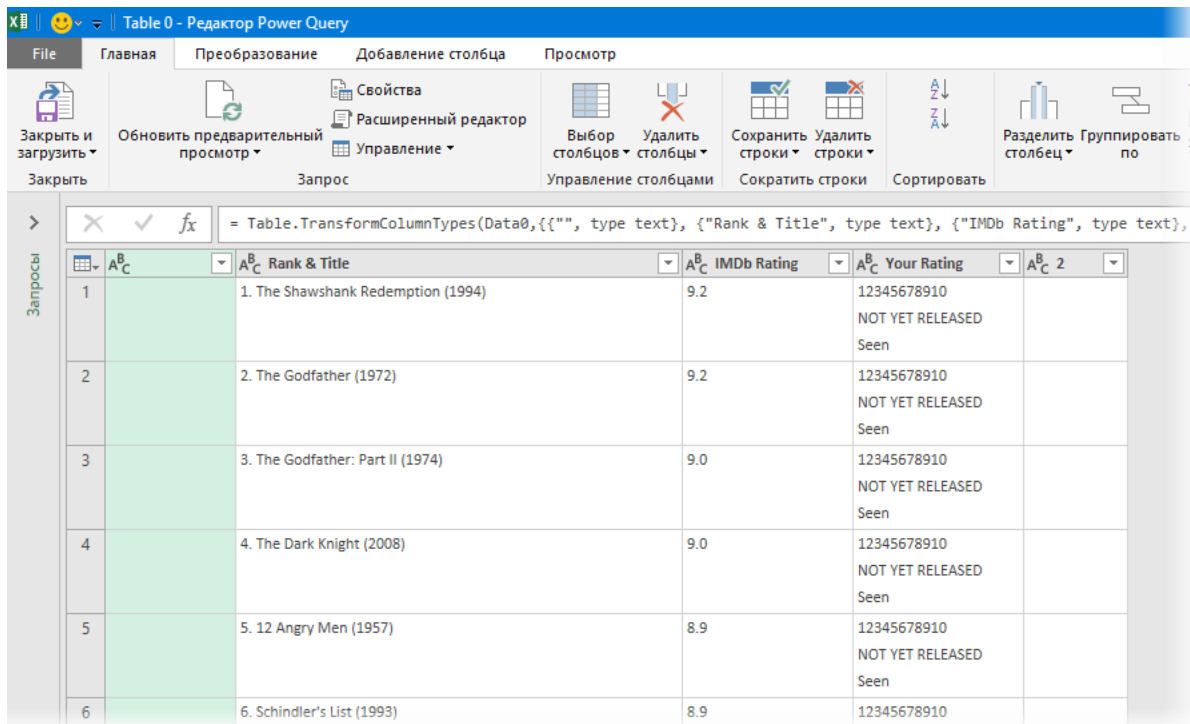
Для доступа к некоторым сайтам (особенно корпоративным) часто требуется пароль, поэтому в следующем окне Power Query предложит нам выбрать вариант доступа: анонимный, использовать пароль Windows, использовать отдельный пароль или учётную запись в организации и т. д.:



Для доступа к сайту IMDb пароль не нужен, поэтому выберем **Анонимно (Anonymously)** и нажмем кнопку **Подключение**. Спустя некоторое время на экране должно появиться окно **Навигатора (Navigator)**, где в левой половине будут перечислены все таблицы, которые Power Query нашел на указанной веб-странице.

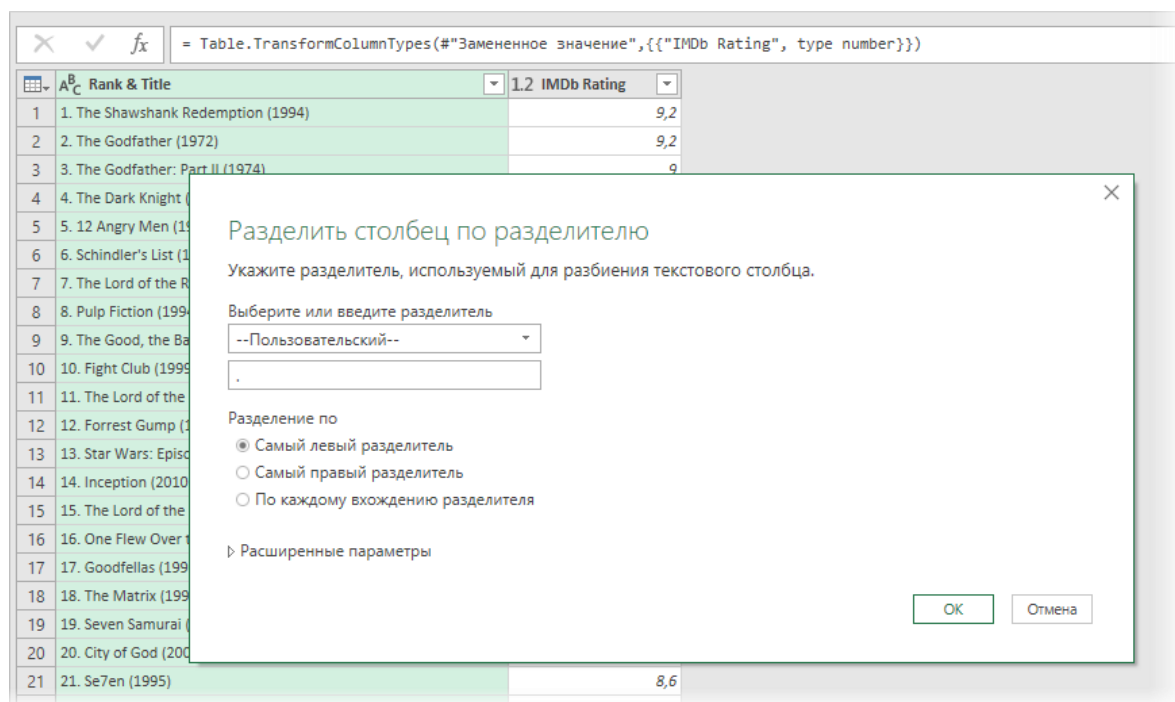


Дальше «методом тыка» находим нужную нам таблицу (в нашем случае это **Table 0**) и жмем кнопку **Преобразовать данные (Transform Data)** или **Изменить (Edit)**. Выбранная таблица загрузится в редактор запросов:



Останется привести её в порядок, а именно:

1. Удалить ненужные столбцы щелчком правой кнопки мыши по заголовку столбца с последующим выбором команды **Удалить столбцы (Remove columns)**.
2. Заменить в столбце рейтинга точку на запятую (если нужно), щелкнув правой кнопкой мыши по заголовку столбца **IMDb Rating** → **Замена значений (Replace Values)**. А затем поменять формат данных в столбце на **Десятичное число (Decimal)**, используя кнопку выбора форматов в шапке столбца.
3. Разделить номер и название фильма в столбце **Rank & Title**, для чего выделить столбец и выбрать **Главная → Разделить столбец → По разделителю (Home → Split column → By delimiter)**:



4. Переименовать столбцы, сделав по их названиям двойной щелчок мышью.

То, что получилось, останется выгрузить обратно в Excel с помощью кнопки **Закреть и загрузить** на вкладке **Главная (Home → Close & Load)**:

	A	B	C	D
1	Позиция	Название фильма	IMDb Rating	
2	1	The Shawshank Redemption (1994)	9,2	
3	2	The Godfather (1972)	9,2	
4	3	The Godfather: Part II (1974)	9	
5	4	The Dark Knight (2008)	9	
6	5	12 Angry Men (1957)	8,9	
7	6	Schindler's List (1993)	8,9	
8	7	The Lord of the Rings: The Return of the King (2003)	8,9	
9	8	Pulp Fiction (1994)	8,9	
10	9	The Good, the Bad and the Ugly (1966)	8,8	
11	10	Fight Club (1999)	8,8	
12	11	The Lord of the Rings: The Fellowship of the Ring (2001)	8,8	

Запросы и подключения

Запросы | Подключения





1 запрос

Топ фильмов с IMDb
Загружено строк: 250.

Прямая загрузка Excel-файлов с веб-страниц

На некоторых сайтах нужная нам информация может быть не просто размещена на веб-странице в виде таблицы или текста, а выложена прямо в виде готовых файлов-книг Excel. Это еще больше облегчает задачу и позволяет моментально загрузить эти данные в Power Query.

Например, на сайте Центрального Банка России на странице с экономическими исследованиями http://cbr.ru/ec_research/ есть ссылки на размещенные там файлы Excel:

-  [итоги десятилетия 2008-2017 / годов в российском банковском секторе: тенденции и факторы](#)
-  [Статистические данные к докладу «Итоги десятилетия 2008-2017 годов в российском банковском секторе: тенденции и факторы»](#)
-  [Анализ долговой нагрузки в отраслях российской экономики](#)
-  [Финансы и кредитование в России](#)

Чтобы загрузить содержимое такого файла, нужно знать прямую ссылку на него. Для этого можно щёлкнуть правой кнопкой мыши по ссылке и выбрать в контекстном меню команду **Копировать ссылку (Copy Link)**. Затем скопированный адрес нужно вставить в окно импорта в Excel, которое появляется после нажатия на кнопку **Из Интернета** на вкладке **Данные (Data → From Internet)**:

Из Интернета

Базовый Подробнее

URL-адрес

Весь дальнейший процесс абсолютно не отличается от загрузки данных из внешнего Excel-файла, который мы уже разбирали.

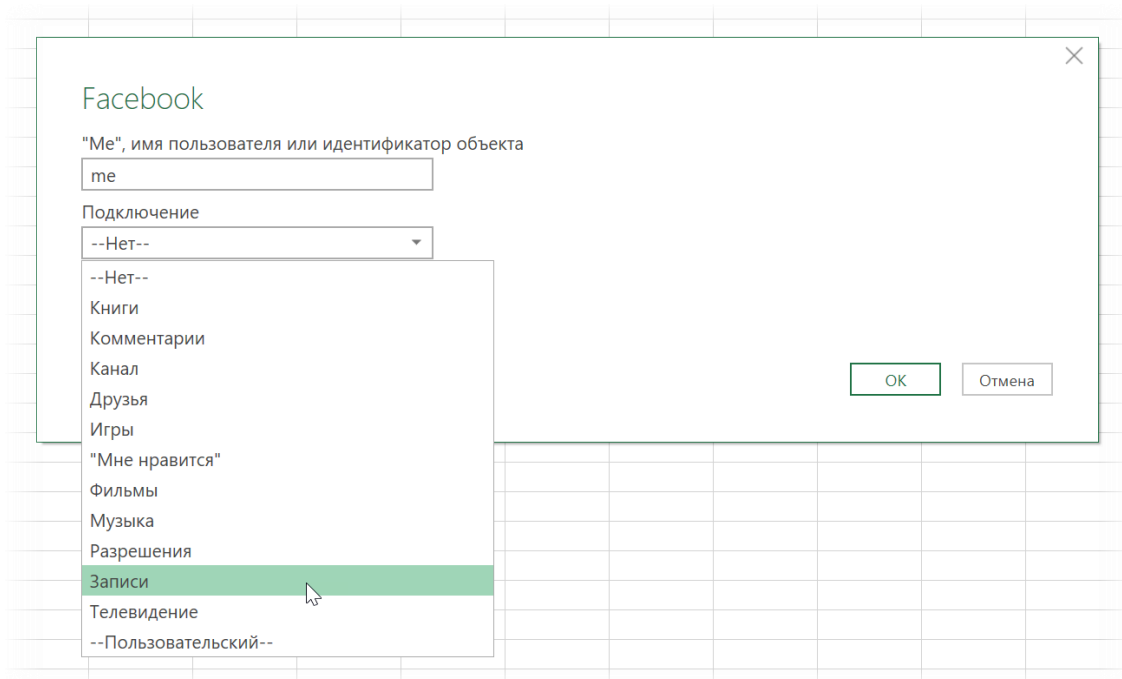
Загрузка данных из Facebook

*Разве я похож на человека, которому нужны деньги?
(Марк Цукерберг)*

В качестве любопытного варианта внешнего коннектора в Power Query встроена возможность импорта данных из Facebook с последующим анализом всех ваших активностей в этой социальной сети, лайков, друзей и прочей статистики. Такая возможность может быть интересна не только топовым блогерам, но и специалистам, курирующим публичные страницы компаний и брендов или продавцам, продвигающим свои товары через соц.сети (SMM).

Для импорта из Facebook необходимо выбрать на вкладке **Данные** → **Получить данные** → **Из веб-служб** → **Из Facebook** (Data → Get Data → From web-services → From Facebook).

После этого нам будет предложено ввести свой логин и пароль для доступа к нашей странице и решить, что именно мы хотим с неё загрузить:



Если выбрать, например, **Записи (Posts)**, то они отобразятся в следующем окне предварительного просмотра:

message	created_time	id	object_link	story
Приехал на тренинг по Power Query в Маххium, а тут...	2019-03-17T10:04:27+0000	1141025795971228_2606033869470406	Record	null
Вернулся из гостеприимной Тюмени с тренинга для...	2019-03-06T17:34:08+0000	1141025795971228_2585434121530381	Record	null
Суперкнига «Tools of Titans» Тима Ферриса @timferri...	2019-02-24T10:27:32+0000	1141025795971228_2563777960362664	Record	null
Закончили продвинутой тренинг по #Excel для сотру...	2019-02-19T18:09:24+0000	1141025795971228_2553321454741648	Record	null
Провёл тренинги для двух весьма продвинутых груп...	2019-02-15T15:56:40+0000	1141025795971228_2543873409019786	Record	null
Закончили тренинги по #Excel для двух команд из R...	2019-02-03T15:55:53+0000	1141025795971228_2517466704993790	Record	null
Сын наедине с морем. Про папу с мамой забыл аж...	2019-01-14T06:03:01+0000	1141025795971228_2474172815989846	Record	null
#таххгоyalbelek #безфильтров	2019-01-09T05:12:54+0000	1141025795971228_2463906193683175	Record	null
Намахались тут сегодня «мотыгами» на тренировке...	2019-01-05T15:02:36+0000	1141025795971228_2456282484445546	Record	null
Зимнее море #безфильтров в #таххгоyalbelek	2019-01-05T09:39:42+0000	1141025795971228_2455789817828146	Record	null
Зимняя Турция мне все больше нравится. #таххгоya...	2019-01-04T11:55:54+0000	1141025795971228_2453960081344453	Record	null
Наш Новый год на этот раз выглядит как-то так. Неп...	2018-12-31T10:26:44+0000	1141025795971228_2445165078890620	Record	null
Хо-хо! Закончили финальный в этом году тренинг - Р...	2018-12-26T15:08:14+0000	1141025795971228_2435433103197151	Record	null
Отточили мастерство создания презентаций на связ...	2018-12-24T18:33:17+0000	1141025795971228_2431287983611663	Record	null
Сегодня в Nestle у меня был редкий, но при этом жу...	2018-12-13T16:18:10+0000	1141025795971228_2407409702666158	Record	null
Приехал в красивый московский офис #bayar - буде...	2018-12-12T07:00:35+0000	1141025795971228_2404471822959946	Record	null
Начинается пора новогоднего обмена подарками с...	2018-12-09T12:43:38+0000	1141025795971228_2398405113566617	Record	null
Продолжаю тихую революцию в обработке данных :...	2018-11-30T05:14:10+0000	1141025795971228_2379284828811979	Record	null
Из серии «придумайте подпись» :)	2018-11-28T19:59:15+0000	1141025795971228_2376666649073797	Record	null
Подоспели фотки с последних семинаров в Астане, к...	2018-11-26T05:31:43+0000	1141025795971228_2371199022953893	Record	null

В предварительном просмотре данные были усечены из-за ограничения размера.

Дальше останется только нажать на кнопку **Преобразовать данные (Transform Data)** или **Изменить (Edit)** в правом нижнем углу окна и загрузить данные в Power Query для дальнейшей обработки и анализа.

Загрузка информации через Open Data Protocol (OData)

Еще одним универсальным способом подключения Power Query ко множеству корпоративных программ и баз данных через интернет может служить загрузка данных через Open Data Protocol (OData).



Open Data Protocol (OData) – это открытый протокол для запросов и обмена данными. С его помощью можно запрашивать у баз данных нужную нам информацию и получать ответы в формате XML или JSON, которые потом загружать в Power Query для дальнейшей обработки и трансформации.

На сегодняшний день этот протокол поддерживает большинство современных корпоративных ERP- и CRM-систем и баз данных, например:

- 1С: Предприятие;
- Microsoft Dynamics;
- SAP;
- SQL Server;
- SharePoint и др.

Также множество некоммерческих организаций и государственных органов «отдают» в этом формате открытые данные о своей работе, статистику различных показателей и т. д. Более подробную информацию об этом протоколе, его версиях и всех программах и сервисах, которые его поддерживают, можно узнать на официальном портале www.odata.org.

Технически использование протокола OData в Power Query очень несложно и напоминает подключение к обычной базе данных, которое мы уже рассматривали на примере Access. Всё, что вам нужно знать для осуществления подключения, – это адрес канала (его должны знать ваши айтишники).

В качестве эксперимента попробуем загрузить в Excel данные с демонстрационного сервера портала www.odata.org, где расположена тестовая база данных **Northwind**, используемая обычно в качестве «подопытного кролика» в подобных ситуациях.

Выберем на вкладке **Данные** → **Получить данные** → **Из других источников** → **Из канала OData** (Data → Get Data → From other sources → OData Feed) и введем в появившееся окно адрес сервера:

A screenshot of the 'Канал OData' (OData Channel) dialog box in Excel. The dialog has a title bar with a close button (X). Below the title, there are two radio buttons: 'Базовый' (Basic) which is selected, and 'Подробнее' (Advanced). Below that is a label 'URL-адрес' (URL address) and a text input field containing 'http://services.odata.org/Northwind/Northwind.svc'. To the left of the input field is a small dropdown menu with 'ABC' and a downward arrow. At the bottom right, there are two buttons: 'ОК' (OK) and 'Отмена' (Cancel). A mouse cursor is pointing at the 'ОК' button.

После нажатия на **ОК** появится уже знакомое нам окно **Навигатора**, где можно стандартным образом выбрать нужную нам таблицу:

Навигатор

Несколько элементов

Отобразить параметры

http://services.odata.org/Northwind/Northwind.svc...

- Alphabetical_list_of_products
- Categories
- Category_Sales_for_1997
- Current_Product_Lists
- Customer_and_Suppliers_by_Cities
- CustomerDemographics
- Customers
- Employees**
- Invoices
- Order_Details

Employees

EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate
1	Davolio	Nancy	Sales Representative	Ms.	08.12.1948 0:00
2	Fuller	Andrew	Vice President, Sales	Dr.	19.02.1952 0:00
3	Leverling	Janet	Sales Representative	Ms.	30.08.1963 0:00
4	Peacock	Margaret	Sales Representative	Mrs.	19.09.1937 0:00
5	Buchanan	Steven	Sales Manager	Mr.	04.03.1955 0:00
6	Suyama	Michael	Sales Representative	Mr.	02.07.1963 0:00
7	King	Robert	Sales Representative	Mr.	29.05.1960 0:00
8	Callahan	Laura	Inside Sales Coordinator	Ms.	09.01.1958 0:00
9	Dodsworth	Anne	Sales Representative	Ms.	27.01.1966 0:00

После нажатия на кнопку **Преобразовать данные** (Transform Data) или **Изменить** (Edit) в правом нижнем углу отмеченная таблица будет загружена в редактор запросов Power Query, откуда её потом можно (после желаемой доработки) загрузить, например, на лист Excel:

	A	B	C	D	E	F	G	
1	EmployeeID	LastName	FirstName	Title	TitleOfCourtesy	BirthDate	HireDate	Address
2	1	Davolio	Nancy	Sales Representative	Ms.	08.12.1948 0:00	01.05.1992 0:00	507 - 20th A
3	2	Fuller	Andrew	Vice President, Sales	Dr.	19.02.1952 0:00	14.08.1992 0:00	908 W. Cap
4	3	Leverling	Janet	Sales Representative	Ms.	30.08.1963 0:00	01.04.1992 0:00	722 Moss B
5	4	Peacock	Margaret	Sales Representative	Mrs.	19.09.1937 0:00	03.05.1993 0:00	4110 Old Re
6	5	Buchanan	Steven	Sales Manager	Mr.	04.03.1955 0:00	17.10.1993 0:00	14 Garrett H
7	6	Suyama	Michael	Sales Representative	Mr.	02.07.1963 0:00	17.10.1993 0:00	Coventry H
8	7	King	Robert	Sales Representative	Mr.	29.05.1960 0:00	02.01.1994 0:00	Edgeham H
9	8	Callahan	Laura	Inside Sales Coordinator	Ms.	09.01.1958 0:00	05.03.1994 0:00	4726 - 11th
10	9	Dodsworth	Anne	Sales Representative	Ms.	27.01.1966 0:00	15.11.1994 0:00	7 Houndsto

Если же говорить о нашей с вами рабочей реальности, то на данный момент доступ по протоколу OData является одним из самых удобных способов загрузки в Excel данных из 1С, которая поддерживает этот вариант обмена данными, начиная с версии 1С:Предприятие 8.3.

Для активации такой возможности необходимо в 1С в режиме конфигуратора выбрать в меню команды **Администрирование** → **Публикация на веб-сервере** и включить затем в появившемся окне флажок **Публиковать стандартный интерфейс OData**:

Публикация на веб-сервере

Основные | OpenID | Прочие

Имя: Demo

Веб-сервер: Internet Information Services

Каталог: C:\inetpub\wwwroot\Demo

Публиковать тонкий клиент и веб-клиент

Публиковать стандартный интерфейс OData

Web-сервисы | HTTP сервисы

Публиковать HTTP сервисы по умолчанию

Имя	Корневой URL
<input checked="" type="checkbox"/> AppMigration	AppMigration
<input checked="" type="checkbox"/> Передача данных	

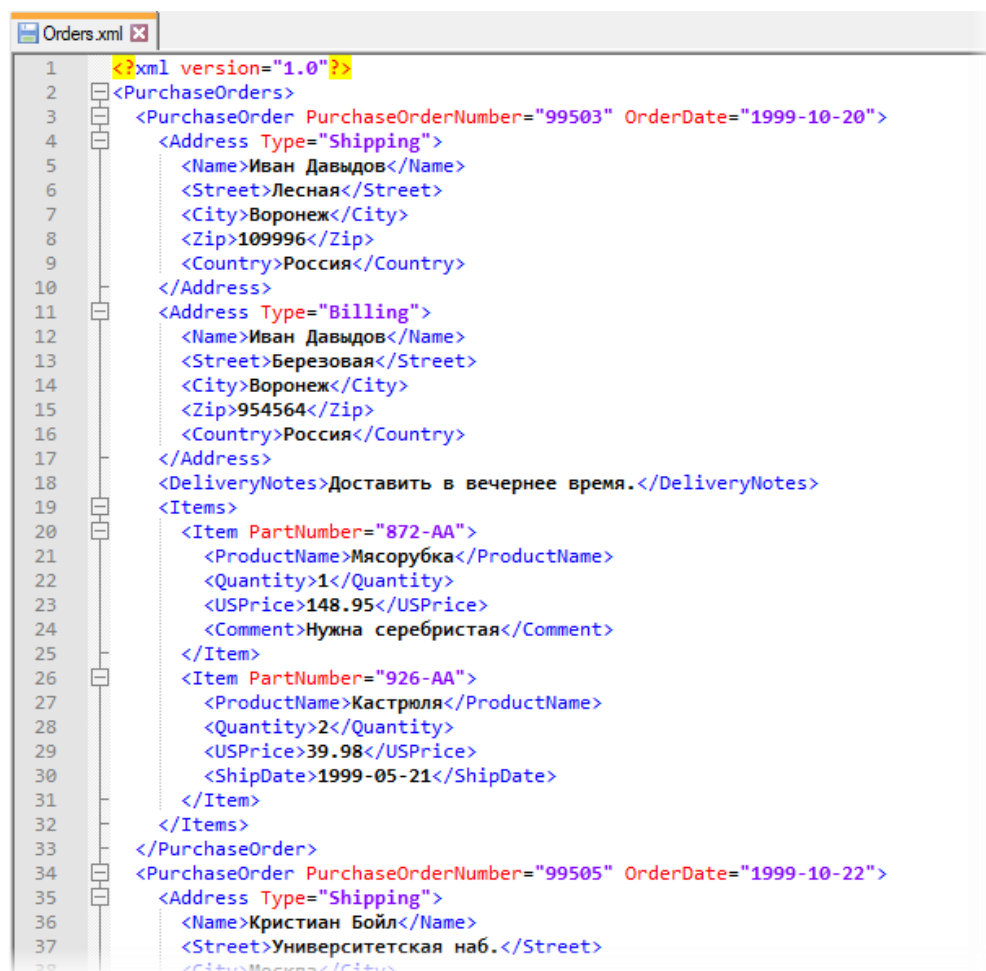
Опубликовать | Отключить | Сохранить | Загрузить | Закрывать | Справка

После нажатия на кнопку **Опубликовать** и последующей перезагрузки сервера для запуска соответствующей службы можно будет обратиться к 1С из Power Query, задав в качестве адресной строки имя веб-сервера и имя базы из публикации, добавив к ним путь `/odata/standard.odata/`. У нас должны запросить пароль, и затем мы увидим все таблицы 1С, которые нам разрешено импортировать.

Загрузка данных из файлов XML

XML (eXtensible Markup Language) – это специальный универсальный язык разметки, очень широко используемый в обработке и хранении данных. Множество программ, баз данных и сервисов могут выгружать во внешний мир данные в этом формате.

По сути, XML-файл – это обычный текстовый файл, хранящий внутри себя информацию, размеченную тегами. Вот так, например, выглядит XML-файл, содержащий список заказов (файл Orders.xml из папки с примерами к этой книге). Для удобства и наглядности я открыл его в бесплатном редакторе **Notepad++** с подсветкой синтаксиса:



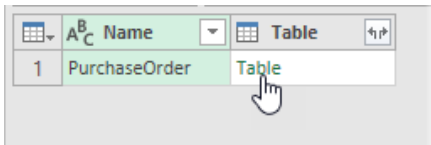
```
1 <?xml version="1.0"?>
2 <PurchaseOrders>
3   <PurchaseOrder PurchaseOrderNumber="99503" OrderDate="1999-10-20">
4     <Address Type="Shipping">
5       <Name>Иван Давыдов</Name>
6       <Street>Лесная</Street>
7       <City>Воронеж</City>
8       <Zip>109996</Zip>
9       <Country>Россия</Country>
10    </Address>
11    <Address Type="Billing">
12      <Name>Иван Давыдов</Name>
13      <Street>Березовая</Street>
14      <City>Воронеж</City>
15      <Zip>954564</Zip>
16      <Country>Россия</Country>
17    </Address>
18    <DeliveryNotes>Доставить в вечернее время.</DeliveryNotes>
19    <Items>
20      <Item PartNumber="872-AA">
21        <ProductName>Мясорубка</ProductName>
22        <Quantity>1</Quantity>
23        <USPrice>148.95</USPrice>
24        <Comment>Нужна серебристая</Comment>
25      </Item>
26      <Item PartNumber="926-AA">
27        <ProductName>Кастрюля</ProductName>
28        <Quantity>2</Quantity>
29        <USPrice>39.98</USPrice>
30        <ShipDate>1999-05-21</ShipDate>
31      </Item>
32    </Items>
33  </PurchaseOrder>
34  <PurchaseOrder PurchaseOrderNumber="99505" OrderDate="1999-10-22">
35    <Address Type="Shipping">
36      <Name>Кристиан Бойл</Name>
37      <Street>Университетская наб.</Street>
38      <City>Москва</City>
```

Обратите внимание на следующее:

- каждый элемент данных обрамляется тегами: имя клиента между **<Name>** и **</Name>**, улица между **<Street>** и **</Street>** и т. д.;
- некоторые блоки вложены друг в друга: например, имя, улица, город и страна вложены в блок **<Address>** ... **</Address>**, а он, в свою очередь, вложен в блок всего заказа **<PurchaseOrder>** ... **</PurchaseOrder>**;
- у некоторых тегов бывают атрибуты, как, например, атрибут **Type** у тегов **Address**, определяющий тип адреса – адрес доставки или адрес для выставления счёта.

Разумеется, Power Query тоже понимает этот язык и легко может импортировать данные в таком виде.

Выберем на вкладке **Данные** команду **Получить данные → Из файла → Из XML (Data → Get Data → From XML)** и укажем местоположение исходного файла. На следующем шаге на экране появится **Навигатор**, где нужно будет выбрать опять же наш файл и нажать кнопку **Преобразовать данные (Transform Data)** или **Изменить (Edit)**, чтобы загрузить всю начинку файла в редактор запросов Power Query. Сначала мы увидим содержимое верхнего уровня – таблицу заказов:



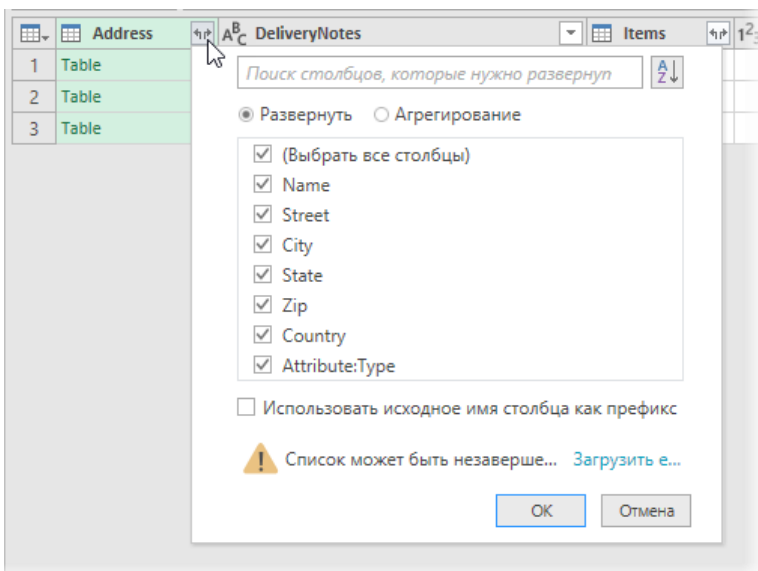
Щелчком по слову **Table** мы «провалимся» на уровень глубже, в нашу таблицу, и увидим список из трех заказов, которые там хранятся:

	Address	DeliveryNotes	Items	Attribute:PurchaseOrderNumber	Attribute:OrderDate
1	Table	Доставить в вечернее время.	Table	99503	20.10.1999
2	Table	Позвонить за час до доставки.	Table	99505	22.10.1999
3	Table		Table	99504	22.10.1999

Обратите внимание, что в ячейках столбцов **Address** и **Items** расположены вложенные таблицы, содержащие, в свою очередь, подробности по каждому адресу и позиции в каждом заказе. Увидеть содержимое этих таблиц можно, если щелкнуть левой кнопкой мыши в белый фон ячейки (но не в слово **Table**):

Name	Street	City	Zip	Country	Attribute:Type
Иван Давыдов	Лесная	Воронеж	109996	Россия	Shipping
Иван Давыдов	Березовая	Воронеж	954564	Россия	Billing

В большинстве случаев для удобства анализа нам лучше отображать все данные в одной таблице. Для этого можно развернуть содержимое всех вложенных таблиц, щелкнув мышью по значку со сдвоенными стрелками в шапке соответствующего столбца:



В открывшемся окне можно выбрать столбцы из вложенной таблицы, которые нам интересны. Также в данном случае можно убрать галочку **Использовать исходное имя столбца как префикс** (Use original column name as prefix), чтобы на выходе не получить названия вида **Address.Name**, **Address.Street**, **Address.City** и т. д. После нажатия на **OK** содержимое вложенных таблиц появится на экране вместо столбца **Address**:

	ABC 123	Name	ABC 123	Street	ABC 123	City	ABC 123	Zip	ABC 123	Country	ABC 123	Attribute:Type	A ^B _C	DeliveryNotes
1		Иван Давыдов		Лесная		Воронеж		109996		Россия		Shipping		Доставить в вечернее время.
2		Иван Давыдов		Березовая		Воронеж		954564		Россия		Billing		Доставить в вечернее время.
3		Кристиан Бойл		Университетская наб.		Москва		198112		Россия		Shipping		Позвонить за час до доставки.
4		Ерофей Барнаулов		Тенистая		Минск		398112		Беларусь		Shipping		

Аналогичным образом, само собой, можно развернуть и таблицы с содержимым заказов в столбце **Items**.

Загрузка и визуализация геоданных из файлов JSON

Ещё одним, хоть и относительно новым, но уже весьма широко распространенным текстовым форматом обмена данными является формат JSON, основанный на языке JavaScript.



Импорт данных в этом формате в Power Query очень похож на загрузку XML-файлов, но имеет некоторые специфические особенности, которые стоит рассмотреть. В качестве примера возьмем JSON-файл с данными о станциях велопроката в Москве, который я загрузил с сайта «Портал открытых данных Российской Федерации», расположенного по адресу <https://data.gov.ru/opendata/7704786030-bikerentalstations>.

В исходном виде он выглядит так (файл **bikes.json** из папки с примерами к книге):

```

1 [
2   {
3     "global_id": 2757596,
4     "Photo": "ef5d82cb-0544-4326-9cb6-a9ed9b347033",
5     "ID": 69,
6     "Name": "Пункт проката велосипедов № 359",
7     "StationCapacity": 18,
8     "BikeParkingSlotsAmount": 18,
9     "Longitude_WGS84": "37.5341",
10    "Latitude_WGS84": "55.6946",
11    "AdmArea": "Западный административный округ",
12    "District": "район Раменки",
13    "Location": "Ломоносовский проспект, дом 66",
14    "DepartmentalAffiliation": "Департамент транспорта и развития дорожно
    -транспортной инфраструктуры города Москвы",
15    "OperOrgName": "ЗАО «СитиБайк»",
16    "OperOrgWebsite": "www.velobike.ru",
17    "Photo_en": "ef5d82cb-0544-4326-9cb6-a9ed9b347033",
18    "ID_en": 69,
19    "Name_en": "Bike rental № 359",
20    "StationCapacity_en": 18,
21    "BikeParkingSlotsAmount_en": 18,
22    "Longitude_WGS84_en": "37.5341",
23    "Latitude_WGS84_en": "55.6946",
24    "AdmArea_en": "Zapadny'j administrativny'j okrug",
25    "District_en": "rajon Ramenki",
26    "Location_en": "Lomonosovskiy prospekt, dom 66",
27    "DepartmentalAffiliation_en": "Department of transportation and development of
    road transport infrastructure of Moscow city",
28    "OperOrgName_en": "ZAO «SitiBayk»",
29    "OperOrgWebsite_en": "www.velobike.ru"
30  },
31  {
32    "global_id": 19646433,
33    "Photo": "a6b08527-1580-4bd1-bfc5-0b555bf1081c",
34    "ID": 323,

```

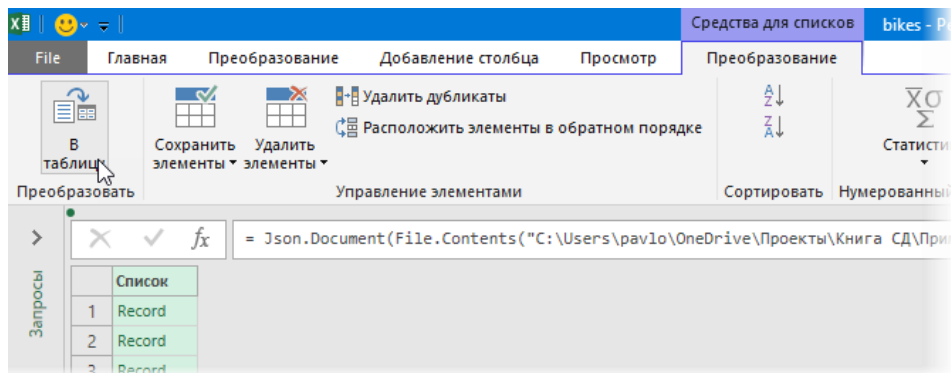
Обратите внимание, что в отличие от XML, где все данные размечаются открывающими и закрывающими тегами, в JSON информация записана в формате «ключ» : «значение», что позволяет его легко читать без использования специальных программ.

Для загрузки выберем в Excel на вкладке **Данные** → **Получить данные** → **Из файла** → **Формат JSON** (Data → Get Data → From file → JSON). После указания местоположения исходного файла **bikes.json** на экране появится окно редактора запросов Power Query, где данные из файла отобразятся в виде списка-столбца с записями (Record) в каждой ячейке. Если щелкнуть мышью в белый фон любой ячейки, можно увидеть содержимое каждой записи – информацию о соответствующей велостанции.

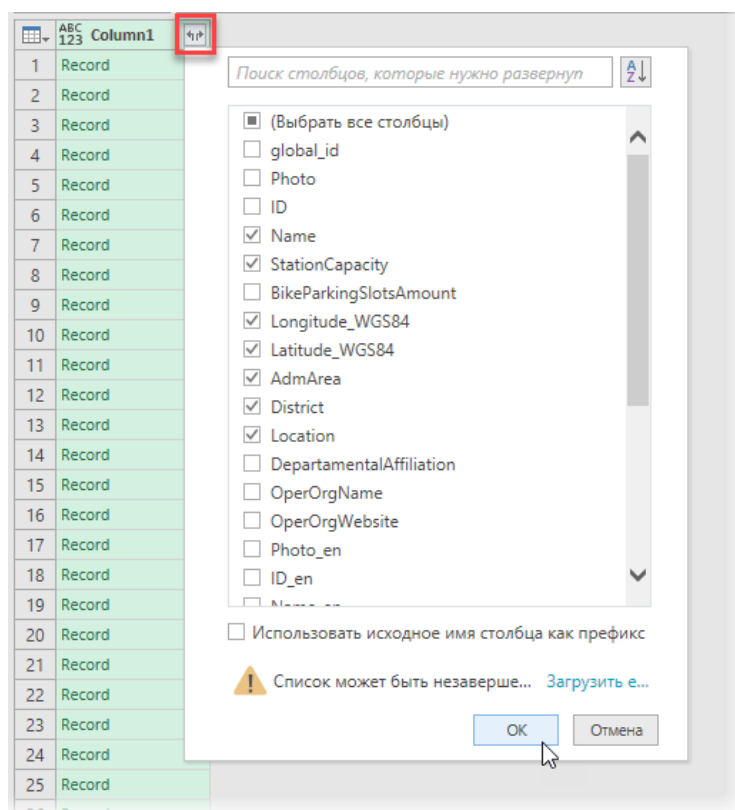
Список	
1	Record
2	Record
3	Record
4	Record
5	Record
6	Record
7	Record

global_id	19646433
Photo	a6b08527-1580-4bd1-bfc5-0b555bf1081c
ID	323
Name	Пункт проката велосипедов № 27
StationCapacity	12
BikeParkingSlotsAmount	12

Чтобы превратить всё это в более привычную и удобную для работы таблицу, выберем на вкладке **Средства для списков - Преобразование** команду **В таблицу** (List Tools → Transform → To table):



После этого список преобразуется в таблицу и в шапке таблицы появится уже знакомая нам кнопка с двойными стрелками для разворачивания содержимого каждой записи. После нажатия на неё можно выбрать нужные столбцы и нажать **ОК**:



Наш столбец развернётся в таблицу:

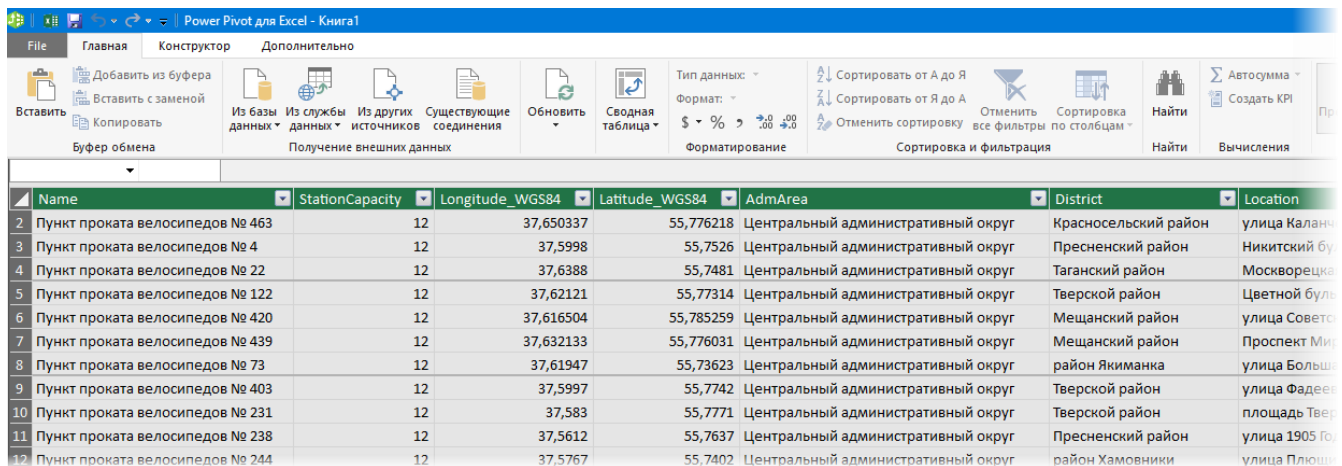
	ABC 123	Name	ABC 123	StationCapacity	ABC 123	Longitude_WGS84	ABC 123	Latitude_WGS84	ABC 123	AdmArea	ABC 123	District
1		Пункт проката велосипедов № 359		18		37.5341		55.6946		Западный административный округ		район Раменки
2		Пункт проката велосипедов № 27		12		37.604770		55.756744		Центральный административный округ		Пресненский район
3		Пункт проката велосипедов № 463		12		37.650337		55.776218		Центральный административный округ		Красносельский райо
4		Пункт проката велосипедов № 386		12		37.519916		55.634468		Юго-Западный административный округ		район Коньково
5		Пункт проката велосипедов № 100		40		37.5933		55.7352		Центральный административный округ		район Хамовники

Останется выбрать для каждого столбца соответствующий формат данных:

- для названий станции, административного округа и района (**Name**, **AdmArea** и **District**) – текстовый;
- для количества велосипедов на станции (**StationCapacity**) – целое число;
- для широты и долготы (**Longitude**, **Latitude**) → **Используя локаль → Английский США** (Use local → English (USA)), чтобы точка корректно распозналась как разделитель целой и дробной части (или просто заменить точку на запятую).

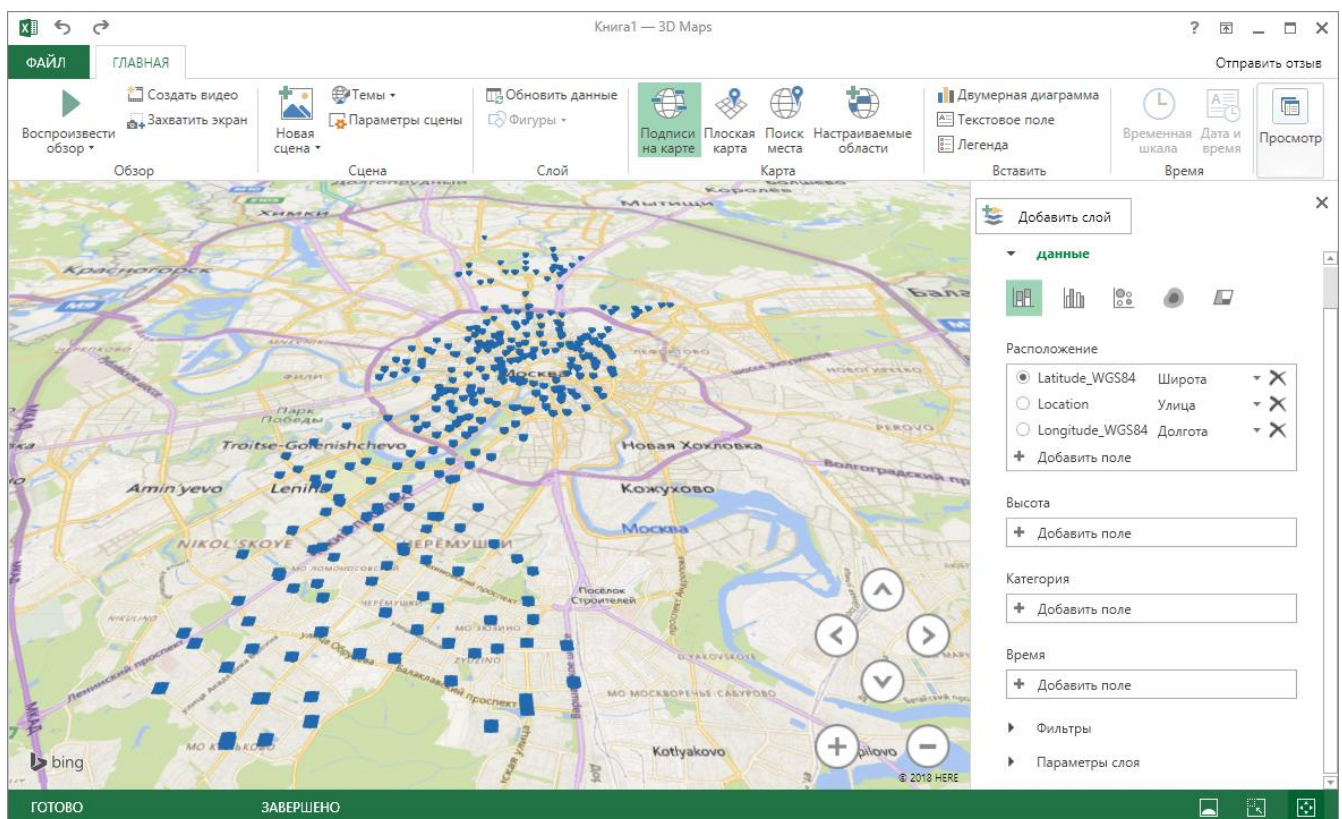
Получившуюся таблицу для разнообразия давайте выгрузим не на лист, как мы уже делали неоднократно в прошлых главах, а в Модель Данных Power Pivot для дальнейшего анализа и визуализации. Для этого на вкладке **Главная** нажмём **Заккрыть и загрузить** → **Заккрыть и загрузить в...** (Home → Close & Load → Close & Load to...) и в открывшемся окне выберем опцию **Только создать подключение** (Only Create Connection) и включим флажок **Добавить эти данные в модель данных** (Add this data to the Data Model).

При необходимости полюбоваться на загруженные данные можно, если открыть окно надстройки Power Pivot с помощью кнопки **Управление моделью данных** на вкладке **Данные** (Data → Manage Data Model):



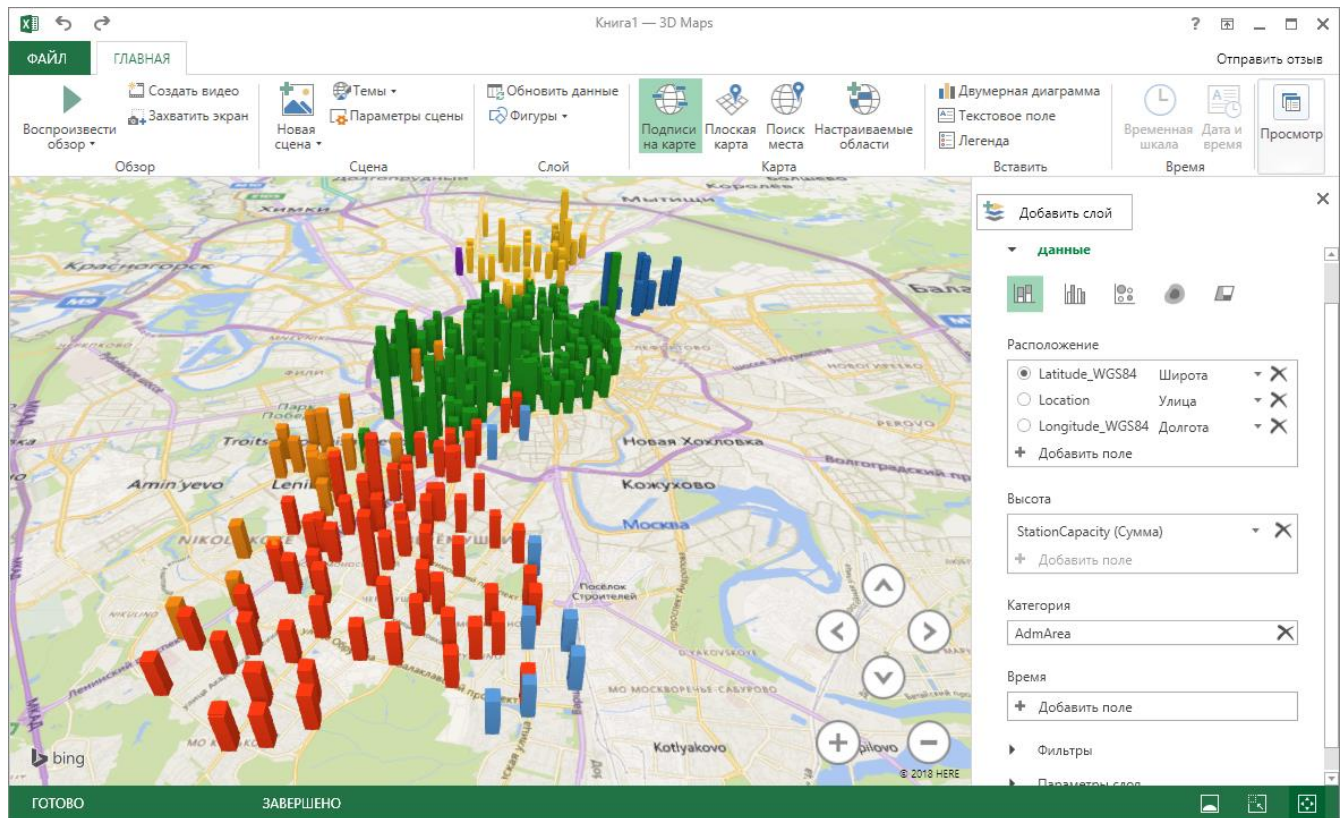
Name	StationCapacity	Longitude_WGS84	Latitude_WGS84	AdmArea	District	Location
Пункт проката велосипедов № 463	12	37,650337	55,776218	Центральный административный округ	Красносельский район	улица Каланчи
Пункт проката велосипедов № 4	12	37,5998	55,7526	Центральный административный округ	Пресненский район	Никитский бульвар
Пункт проката велосипедов № 22	12	37,6388	55,7481	Центральный административный округ	Таганский район	Москворецкая набережная
Пункт проката велосипедов № 122	12	37,62121	55,77314	Центральный административный округ	Тверской район	Цветной бульвар
Пункт проката велосипедов № 420	12	37,616504	55,785259	Центральный административный округ	Мещанский район	улица Советская
Пункт проката велосипедов № 439	12	37,632133	55,776031	Центральный административный округ	Мещанский район	Проспект Мира
Пункт проката велосипедов № 73	12	37,61947	55,73623	Центральный административный округ	район Якиманка	улица Большая Якиманка
Пункт проката велосипедов № 403	12	37,5997	55,7742	Центральный административный округ	Тверской район	улица Фадеева
Пункт проката велосипедов № 231	12	37,583	55,7771	Центральный административный округ	Тверской район	площадь Твердышкина
Пункт проката велосипедов № 238	12	37,5612	55,7637	Центральный административный округ	Пресненский район	улица 1905 года
Пункт проката велосипедов № 244	12	37,5767	55,7402	Центральный административный округ	район Хамовники	улица Плющиха

Теперь давайте попробуем визуализировать наши данные на карте, используя еще одну надстройку Microsoft Excel, которая называется Power Map и использует загруженные в Модель Данных координаты. Вернёмся в окно Microsoft Excel и на вкладке **Вставка** (Insert) нажмем на кнопку **3D-карта** (3D-map):

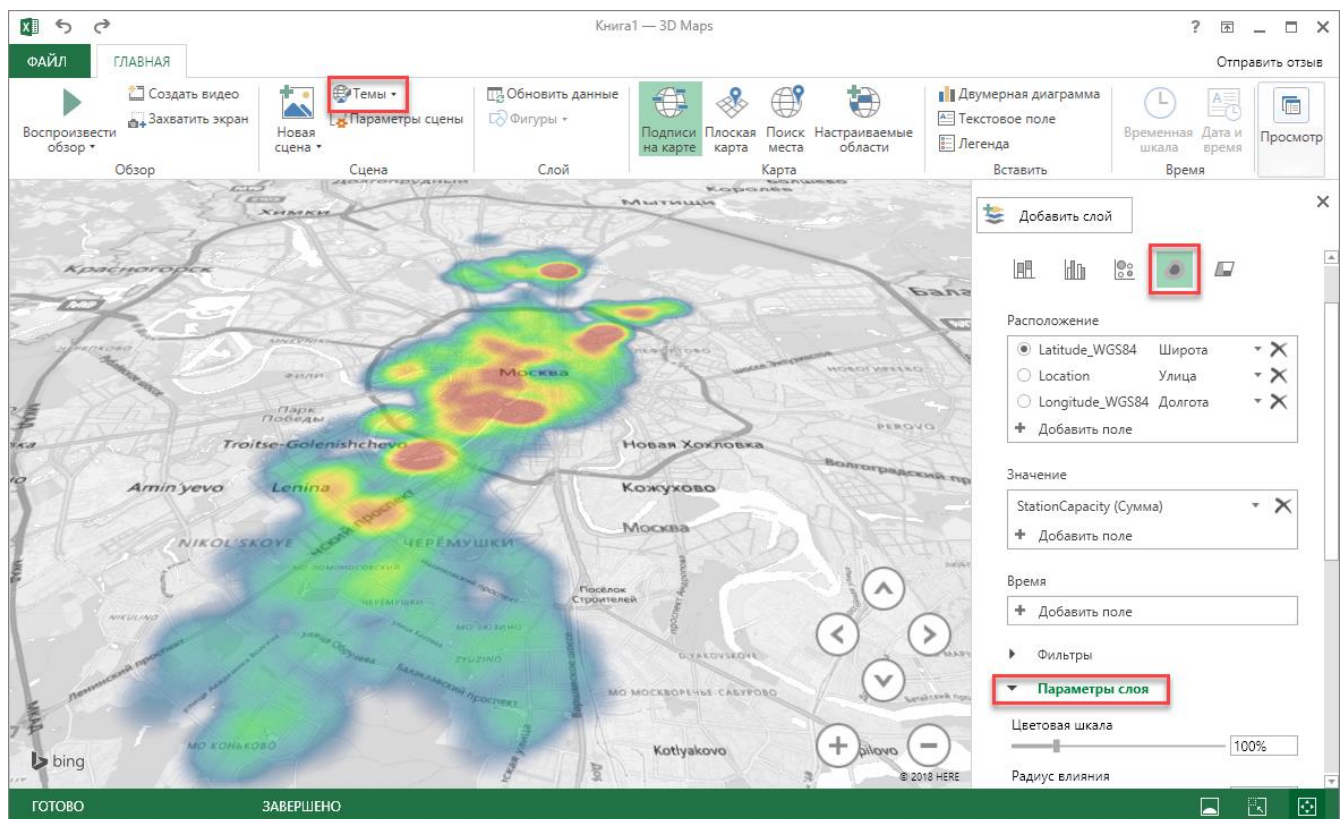


Дополнительно можно:

- включить на карте отображение названий кнопкой **Подписи на карте** (Labels on map);
- добавить на правой панели в поле **Высота** (Height) количество велосипедов на станции (StationCapacity);
- отобразить каждый округ своим цветом, добавив там же в поле **Категория** (Category) столбец AdmArea.



А еще можно поиграться с цветовой темой (выпадающий список **Тема**), с настройками отображения (**Параметры слоя** в правом нижнем углу) и переключиться в другой тип отображения – тепловую карту – для имитации «прогноза погоды»:



Вывод: используя связку Power Query – Power Pivot – Power Map, можно легко делать совершенно замечательные вещи.

Загрузка данных из PDF через Word

Задача переноса данных из таблицы в PDF-файле на лист Microsoft Excel – это всегда «весело». Особенно если у вас нет дорогих программ распознавания типа FineReader или чего-то подобного. Прямое копирование обычно ни к чему хорошему не приводит, т. к. после вставки скопированных данных на лист они, скорее всего, слипнутся в один столбец. Так что их потом придется кропотливо разделять с помощью инструмента **Текст по столбцам** с вкладки **Данные** (Data - Text to Columns).

И само собой разумеется, копирование возможно только для тех PDF-файлов, где есть текстовый слой, т. е. с только что отсканированным с бумаги в PDF документом это не сработает в принципе.

Но все не так грустно на самом деле. Связка из Word и Power Query с большой вероятностью может помочь в этом вопросе.

Для примера давайте возьмем вот такой PDF-отчет с кучей текста, формул и таблиц с сайта Европейской экономической комиссии (файл **European Commission Report 2013.pdf** в папке с примерами к этой книге):

The image shows a PDF document titled "STATISTICAL ANNEX" from the "European Commission Report 2013". The document is divided into sections, with the visible part being "7.1. SECTORAL COMPETITIVENESS INDICATORS".

7.1. SECTORAL COMPETITIVENESS INDICATORS
 7.1.1. Employment rates
 Occupational coverage, all indicators refer to EU-28
 Production Index⁽¹⁾: The production index is actually an index of final production in volume terms.
 Labour productivity: This indicator is calculated by combining the indices of production and number of persons employed or number of hours worked⁽²⁾. Therefore, this indicator measures final production per person or final production per hour worked.
 Unit Labour Cost: It is calculated from the production index and the index of wages and salaries and measures labour cost per unit of production. "Wages and salaries" is defined (Eurostat) as "the total remuneration, in cash or in kind, payable to all persons employed on the ground (including home-workers), to receive the work done during the accounting period, regardless of whether it is paid on the basis of working time, output or piecework and whether it is paid regularly or not and whether it includes social contributions payable by the employer".
 Relative Trade Balance: It is calculated, for each sector "X", as $(X/M)/(X/C/M)$, where X, C and M are EU-28 exports and imports of products of sector "X" and then the rest of the World.
 Revised Comparative Advantage (RCA):
 The RCA indicator for product "P" is defined as follows:

$$RCA_P = \frac{\frac{X_{P,EU}}{\sum X_{P,EU}}}{\frac{X_{P,World}}{\sum X_{P,World}}}$$

where X is value of exports for reference group ("P") in the EU-28 plus 105 other countries (see list below); the numerator is the EU-28 share of the total exports of product "P", while the denominator is the rest of the world (including intra-EU trade) and X_{World} measures exports to the rest of the world by the countries in the reference group. The latter consists of the EU-28 plus the following countries: Albania, Algeria, Andorra, Argentina, Australia, Bahrain, Bangladesh, Belarus, Belgium, Bolivia (Plurinational State of), Bosnia Herzegovina, Brazil, Bulgaria, Cambodia, Canada, Chile, China, Costa Rica, Cuba, Cyprus, Czechia, Colombia, Costa Rica, Denmark, Dominican Republic, Ecuador, El Salvador, Equatorial Guinea, Ethiopia, Fiji, French Polynesia, Georgia, Grenada, Guatemala, Honduras, Hungary, Iceland, India, Indonesia, Israel, Italy, Jamaica, Jordan, Kazakhstan, Korea, Republic of, Kyrgyzstan, Lebanon, Liechtenstein, Lithuania, Luxembourg, Madagascar, Maldives, Malaysia, Mali, Mauritania, Mauritius, Mexico, Other Asia, Republic of Moldova, Montenegro, Mozambique, Myanmar, Nicaragua, Oman, New Zealand, Norway, Pakistan, Panama, Papua New Guinea, Paraguay, Peru, Russian Federation, Rwanda, Saint Vincent and the Grenadines, Sao Tome and Principe, Saudi Arabia, Senegal, Serbia, Seychelles, Singapore, Vietnam, South Africa, Sri Lanka, Sudan, Switzerland, Thailand, Timor, Tonga, Tunisia, Turkey, Ukraine and Central Europe, Ukraine, Ukraine, TPR of Macedonia, Egypt, United Republic of Tanzania, US, Uruguay, Vanuatu, Yemen, Zambia.

Statistical annexes to the indicators in Tables 7.1 to 7.6 are presented at the level of divisions of the statistical classification of economic activities in the European Community (NACE Rev. 2)⁽³⁾, while those in Tables 7.7 to 7.10 are presented in terms of divisions of the statistical classification of products by activity (CPA). Table 7.11 uses extended indices of payments services classification, by terms of data source. Tables 7.1 to 7.6 are based on Eurostat's short-term indicators data. Tables 7.7 to 7.10 are based on United Nations' COMTRADE. Table 7.11 is based on ILO's database of payments. Republic and former territories not included in EU are not related to a specific activity.

Footnote 1: The data are missing or not reported for some countries.
 Footnote 2: The data are missing or not reported for some countries.
 Footnote 3: The data are missing or not reported for some countries.

Table 7.1 EU-28 Industry production index, annual growth rate (%)
 Table 7.2 EU-28 Number of persons employed, annual growth rate (%)
 Table 7.3 EU-28 Labour productivity per person employed, annual growth rate (%)
 Table 7.4 EU-28 Relative trade balance, annual growth rate (%)
 Table 7.5 EU-28 Revised Comparative Advantage (RCA) indicator

и попробуем вытащить из него в Excel, скажем, первую таблицу:

Table 7.1: EU-28 - Industry production index, annual growth rate (%)

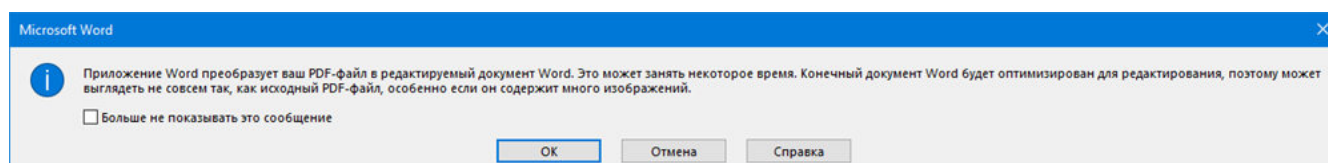
Code (NACE Rev. 2)	Sector	2002	2003	2004	2005	2006	2007	2008	2009
B	MINING AND QUARRYING	0.7	-3.0	-1.9	-5.3	-3.1	-0.1	-3.6	-10.9
C	MANUFACTURING	-0.7	0.2	2.5	1.8	4.8	4.2	-1.9	-15.3
C10	Manufacture of food products	1.9	0.5	2.2	2.4	1.3	2.0	-0.4	-1.1
C11	Manufacture of beverages	1.7	1.2	-2.3	1.0	3.9	1.3	-2.1	-3.2
C12	Manufacture of tobacco products	-2.3	-5.9	-11.6	-5.4	-4.8	1.5	-11.9	-0.9
C13	Manufacture of textiles	-4.5	-3.4	-4.9	-5.9	-0.8	-1.1	-10.4	-17.9
C14	Manufacture of wearing apparel	-11.5	-7.3	-5.7	-10.4	-0.5	-0.5	-7.6	-13.9
C15	Manufacture of leather and related products	-8.3	-6.9	-10.2	-9.1	-2.9	-5.7	-8.1	-14.2
C16	Manufacture of wood and of products of wood and cork, except furniture; manufacture of articles of straw and plaiting materials	0.6	2.2	3.2	0.2	4.2	1.0	-9.1	-15.1
C17	Manufacture of paper and paper products	3.4	1.4	2.8	-0.1	3.9	2.6	-3.2	-8.8
C18	Printing and reproduction of recorded media	-0.6	-1.3	1.4	2.3	0.2	0.7	-2.2	-7.8
C19	Manufacture of coke and refined petroleum products	0.9	1.3	4.6	0.7	-0.7	0.2	1.0	-8.0
C20	Manufacture of chemicals and chemical products	1.8	-0.2	3.5	2.3	3.7	3.1	-3.2	-12.2
C21	Manufacture of basic pharmaceutical products and pharmaceutical preparations	8.5	4.7	-0.2	4.8	5.9	0.4	0.7	2.9
C22	Manufacture of rubber and plastic products	-0.1	1.8	1.8	0.9	3.9	4.5	-4.6	-14.0
C23	Manufacture of other non-metallic mineral products	-1.6	0.3	1.6	0.6	4.3	1.9	-6.8	-19.4
C24	Manufacture of basic metals	-0.4	-0.5	5.3	-0.8	6.4	1.5	-3.4	-27.3
C25	Manufacture of fabricated metal products, except machinery and equipment	-0.5	0.9	2.6	1.6	4.8	6.2	-3.0	-22.7
C26	Manufacture of computer, electronic and optical products	-10.4	0.5	6.3	2.6	8.8	7.5	0.8	-17.4
C27	Manufacture of electrical equipment	-4.3	-1.5	2.3	1.0	8.5	4.3	-0.7	-21.0
C28	Manufacture of machinery and equipment n.e.c.	-1.8	-0.8	4.1	4.0	8.4	8.4	1.5	-26.9
C29	Manufacture of motor vehicles, trailers and semi-trailers	0.7	1.6	4.4	1.4	3.3	6.1	-5.9	-25.1

Шаг 1. Открываем PDF в Word

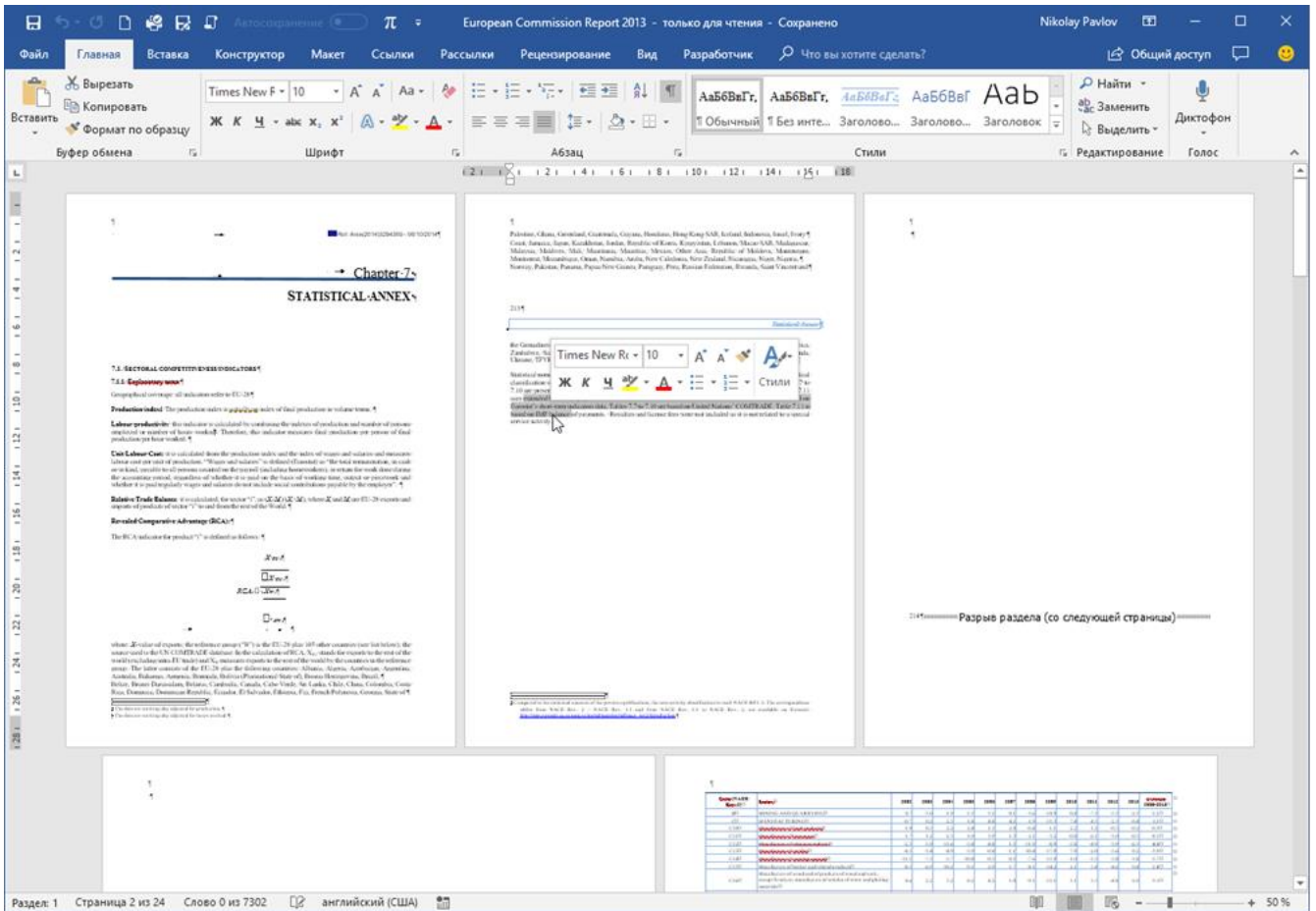
Почему-то мало кто знает, но начиная с 2013 года Microsoft Word научился открывать и распознавать PDF-файлы (даже отсканированные, т. е. без текстового слоя!). Сразу хотелось бы предупредить, что не всегда этот процесс происходит идеально, и наверняка найдутся особо зверские PDF-файлы, с распознаванием которых Word не справится. Тогда вас спасут только специализированные программы распознавания (всё тот же FineReader, например), а иногда не спасут даже они (вспомните почерк врачей).

В любом случае попробовать стоит, как мне кажется. Тем более что делается это совершенно банальным образом: открываем Word, жмем **Файл - Открыть (File - Open)** и уточняем PDF-формат в выпадающем списке в правом нижнем углу окна.

Затем выбираем нужный нам PDF-файл и жмем **Открыть (Open)**. Word сообщает нам, что собирается запустить распознавание этого документа в текст:



Соглашаемся и через несколько секунд увидим наш PDF открытым для редактирования уже в Word:



Конечно, у документа частично слетит дизайн, стили, шрифты, колонтитулы и т. п., но для нас это не важно, нам нужны только данные из таблиц. В принципе на этом этапе уже возникает соблазн дальше просто скопировать таблицу из распознанного документа в Word и просто вставить ее в Excel. Иногда это срабатывает, но чаще приводит ко всевозможным искажениям данных: например, числа могут превратиться в даты или остаться текстом, как в нашем случае, т. к. в PDF используется не российские разделители:

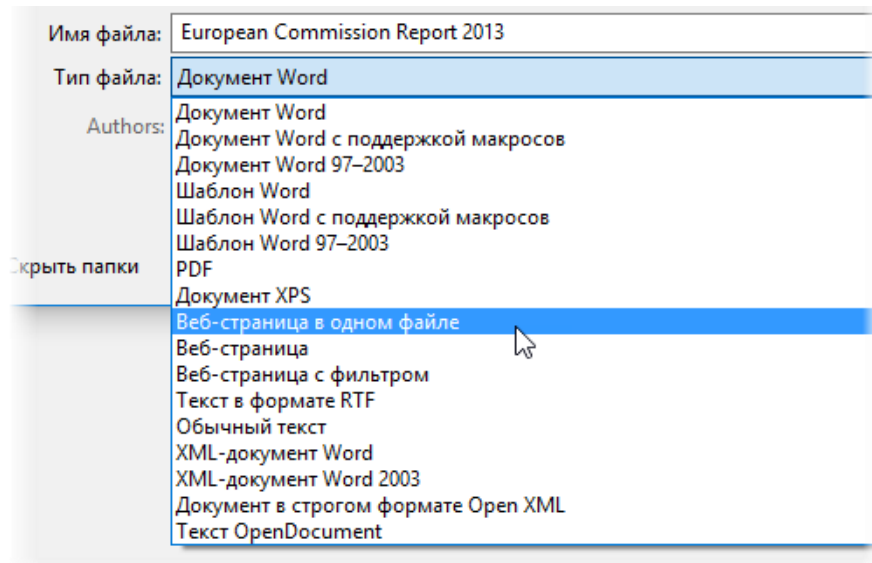
	A	B	C	D	E	F	G	H	I
1	Code (NAI Sector		2002	2003	2004	2005	2006	2007	2008
2	B	MINING AND QUARRYING	0.7	-3.0	-1.9	-5.3	-3.1	-0.1	-3.6
3	C	MANUFACTURING	-0.7	0.2	02.май	01.авг	04.авг	04.фев	-1.9
4	C10	Manufacture of food products	01.сен	0.5	02.фев	02.апр	01.мар	2.0	-0.4
5	C11	Manufacture of beverages	01.июл	01.фев	-2.3	1.0	03.сен	01.мар	-2.1
6	C12	Manufacture of tobacco products	-2.3	-5.9	-11.6	-5.4	-4.8	01.май	-11.9
7	C13	Manufacture of textiles	-4.5	-3.4	-4.9	-5.9	-0.8	-1.1	-10.4
8	C14	Manufacture of wearing apparel	-11.5	-7.3	-5.7	-10.4	-0.5	-0.5	-7.6
9	C15	Manufacture of leather and related produ	-8.3	-6.9	-10.2	-9.1	-2.9	-5.7	-8.1
10	C16	Manufacture of wood and of products of v	0.6	02.фев	03.фев	0.2	04.фев	1.0	-9.1
11	C17	Manufacture of paper and paper products	03.апр	01.апр	02.авг	-0.1	03.сен	02.июн	-3.2
12	C18	Printing and reproduction of recorded me	-0.6	-1.3	01.апр	02.мар	0.2	0.7	-2.2
13	C19	Manufacture of coke and refined petroleu	0.9	01.мар	04.июн	0.7	-0.7	0.2	1.0
14	C20	Manufacture of chemicals and chemical p	01.авг	-0.2	03.май	02.мар	03.июл	03.январ	-3.2
15	C21	Manufacture of basic pharmaceutical proc	08.май	04.июл	-0.2	04.авг	05.сен	0.4	0.7
16	C22	Manufacture of rubber and plastic produc	-0.1	01.авг	01.авг	0.9	03.сен	04.май	-4.6

Так что давайте не будем «срезать углы», а сделаем все чуть сложнее, но правильно.

Этап 2. Сохраняем документ как веб-страницу

Чтобы потом загрузить полученные данные в Excel (через Power Query), наш документ в Word нужно сохранить в формате веб-страницы: этот формат является в данном случае неким общим знаменателем между Word и Excel.

Для этого идем в меню **Файл** → **Сохранить как** (File → Save As) или жмем клавишу **F12** на клавиатуре и в открывшемся окне выбираем тип файла **Веб-страница в одном файле** (Webpage - Single file):

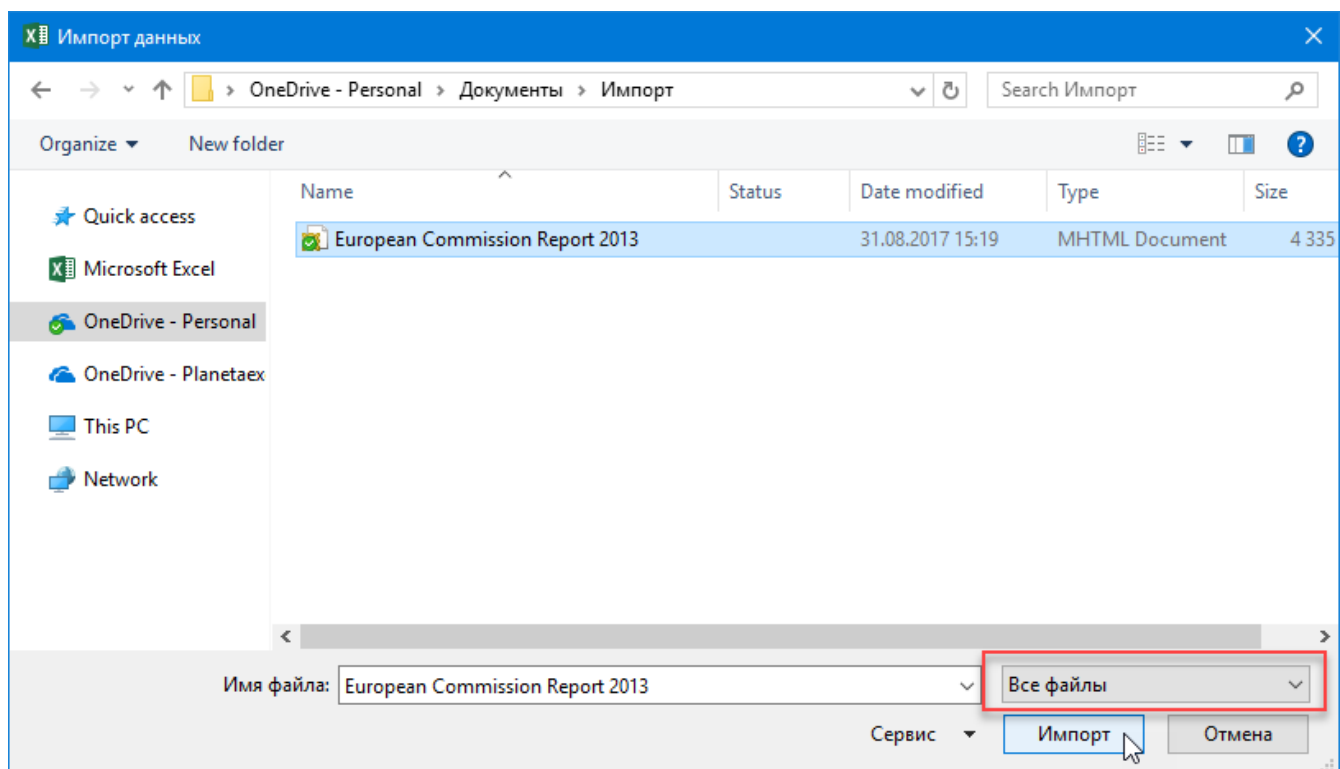


После сохранения должен получиться файл с расширением **mhtml** (если у вас в Проводнике видны расширения файлов).

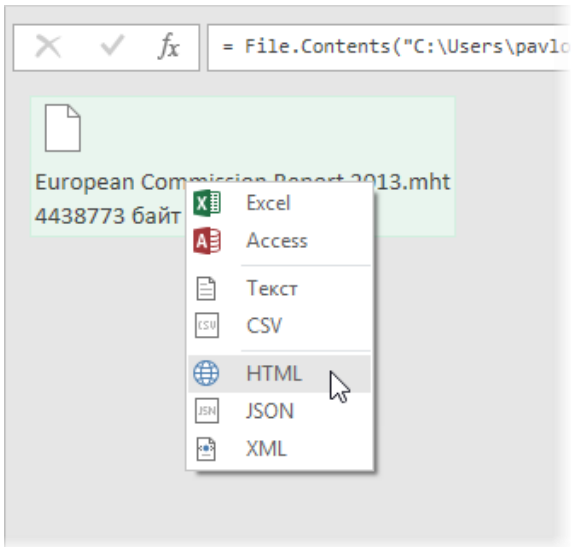
Этап 3. Загружаем файл в Excel через Power Query

Можно открыть созданный MHTML-файл в Excel напрямую, но тогда мы получим, во-первых, сразу все содержимое PDF вместе с текстом и кучей ненужных таблиц, а во-вторых, опять потеряем данные из-за неправильных разделителей. Поэтому импорт в Excel мы будем делать как раз через Power Query.

Так что идем на вкладку **Данные** (Data) и выбираем команду **Получить данные** или **Создать запрос** → **Из файла** → **Из XML** (Get Data → From file → From XML). Чтобы были видны не только XML-файлы меняем в выпадающем списке в правом нижнем углу окна фильтры на **Все файлы** (All files) и указываем наш MHTML-файл:



Обратите внимание, что импорт успешно не завершится, т. к. Power Query ждет от нас XML, а у нас на самом деле HTML. Поэтому в следующем появившемся окне нужно будет щелкнуть правой кнопкой мыши по непонятному для Power Query файлу и уточнить его формат:



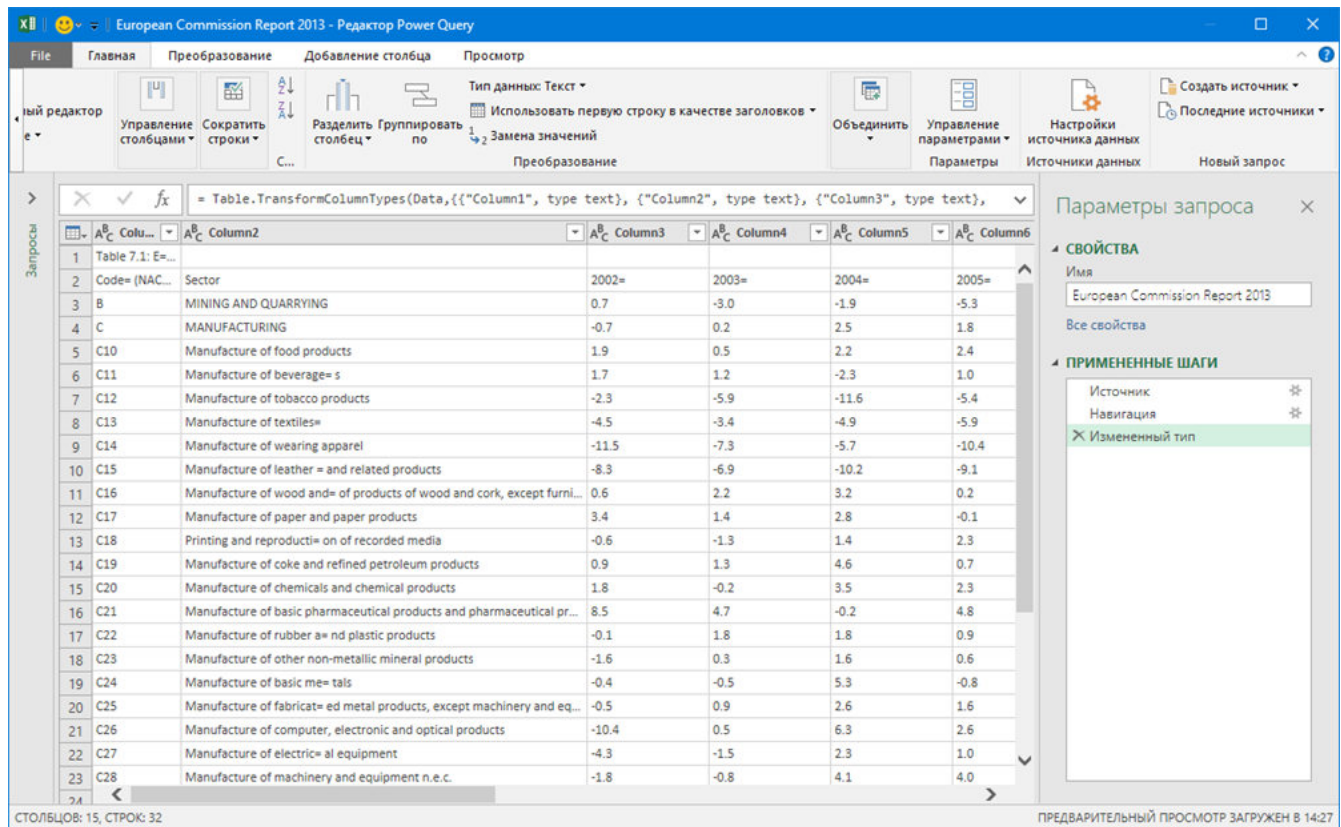
После этого файл будет корректно распознан, и мы увидим список всех таблиц, которые в нем есть:

Caption	Source	ClassName	Id	Data
1	null	Table	3DTableGrid	null Table
2	null	Table	3DTableGrid	null Table
3	null	Table	3DTableGrid	null Table
4	null	Table	3DTableGrid	null Table
5	null	Table	3DTableGrid	null Table
6	null	Table	3DTableGrid	null Table
7	null	Table	3DTableGrid	null Table
8	null	Table	3DTableGrid	null Table
9	null	Table	3DTableGrid	null Table
10	null	Table	3DTableGrid	null Table
11	null	Table	3DTableGrid	null Table
12	Document	Service	null	null Table

Column2	Column3	Column4	Column5	Column6	Column7	Column8
g growth rate (%)						
Sector	2002=	2003=	2004=	2005=	2006=	
MINING AND QUARRYING	0.7	-3.0	-1.9	-5.3	-3.1	
MANUFACTURING	-0.7	0.2	2.5	1.8	4.8	
Manufacture of food products	1.9	0.5	2.2	2.4	1.3	
Manufacture of beverage* s	1.7	1.2	-2.3	1.0	3.9	

Посмотреть содержимое таблиц можно, если щелкать левой кнопкой мыши в белый фон (не в слово Table!) ячеек в столбце **Data**.

Когда нужная таблица определена, щелкните по зеленому слову **Table** – и вы «провалитесь» в её содержимое:



Останется проделать несколько простых действий, чтобы «причесать» её содержимое:

1. Удалить ненужные столбцы: правой кнопкой мыши по заголовку столбца **Удалить (Remove Columns)**.
2. Заменить точки на запятые (если у вас на компьютере они установлены как разделить целой и дробной части): выделить столбцы, щелкнуть правой **Замена значений (Replace Values)**.
3. Удалить знаки «равно» в шапке: выделить столбцы, щелкнуть правой **Замена значений (Replace Values)**.
4. Удалить верхнюю строку: **Главная - Удалить строки - Удаление верхних строк (Home - Remove rows - Remove top rows)**.
5. Удалить пустые строки: **Главная - Удалить строки - Удаление пустых строк (Home - Remove rows - Remove empty rows)**.
6. Поднять первую строку в шапку таблицы: **Главная - Использовать первую строку в качестве заголовков (Home - Use first row as header)**.
7. Отфильтровать лишние данные с помощью фильтров в шапке таблицы.

Когда таблица будет приведена в нормальный вид, ее можно выгрузить на лист уже знакомым нам образом – командой **Закрыть и загрузить (Close & Load)** на **Главной (Home)** вкладке. И мы получим вот такую красоту, с которой уже можно работать:

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Sector	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013
2	MINING AND QUARRYING	0,7	-3	-1,9	-5,3	-3,1	-0,1	-3,6	-10,9	0,4	-7,3	-5,3	-2,3
3	MANUFACTURING	-0,7	0,2	2,5	1,8	4,8	4,2	-1,9	-15,3	7,4	4,5	-2,3	-0,4
4	Manufacture of food products	1,9	0,5	2,2	2,4	1,3	2	-0,4	-1,1	2,2	1,2	-0,5	-0,2
5	Manufacture of beverage s	1,7	1,2	-2,3	1	3,9	1,3	-2,1	-3,2	-0,8	6,1	-3	-0,5
6	Manufacture of tobacco products	-2,3	-5,9	-11,6	-5,4	-4,8	1,5	-11,9	-0,9	-5,8	-4,9	-3,9	-6,3
7	Manufacture of textiles	-4,5	-3,4	-4,9	-5,9	-0,8	-1,1	-10,4	-17,9	7,9	-2	-5,6	0,2
8	Manufacture of wearing apparel	-11,5	-7,3	-5,7	-10,4	-0,5	-0,5	-7,6	-13,9	-1	-3,5	-5,8	-3,8
9	Manufacture of leather and related products	-8,3	-6,9	-10,2	-9,1	-2,9	-5,7	-8,1	-14,2	2,1	5,4	-4,6	0,4
10	Manufacture of wood and of products of wood and cork	0,6	2,2	3,2	0,2	4,2	1	-9,1	-15,1	3,1	3,5	-4,8	-1
11	Manufacture of paper and paper products	3,4	1,4	2,8	-0,1	3,9	2,6	-3,2	-8,8	6,2	-0,6	-1,6	-0,5
12	Printing and reproduction of recorded media	-0,6	-1,3	1,4	2,3	0,2	0,7	-2,2	-7,8	-0,2	-1,9	-6,1	-3,4
13	Manufacture of coke and refined petroleum products	0,9	1,3	4,6	0,7	-0,7	0,2	1	-8	-2	-1,3	-1,8	-1,7
14	Manufacture of chemicals and chemical products	1,8	-0,2	3,5	2,3	3,7	3,1	-3,2	-12,2	10,6	1,9	-2,3	-0,1
15	Manufacture of basic pharmaceutical products and pharmaceutical preparations	8,5	4,7	-0,2	4,8	5,9	0,4	0,7	2,9	4,9	1,8	-0,3	3,4
16	Manufacture of rubber and plastic products	-0,1	1,8	1,8	0,9	3,9	4,5	-4,6	-14	7,5	4,2	-3,2	0,4
17	Manufacture of other non-metallic mineral products	-1,6	0,3	1,6	0,6	4,3	1,9	-6,8	-19,4	1,9	3,1	-8,5	-3

Загрузка данных почты и календаря из Microsoft Exchange

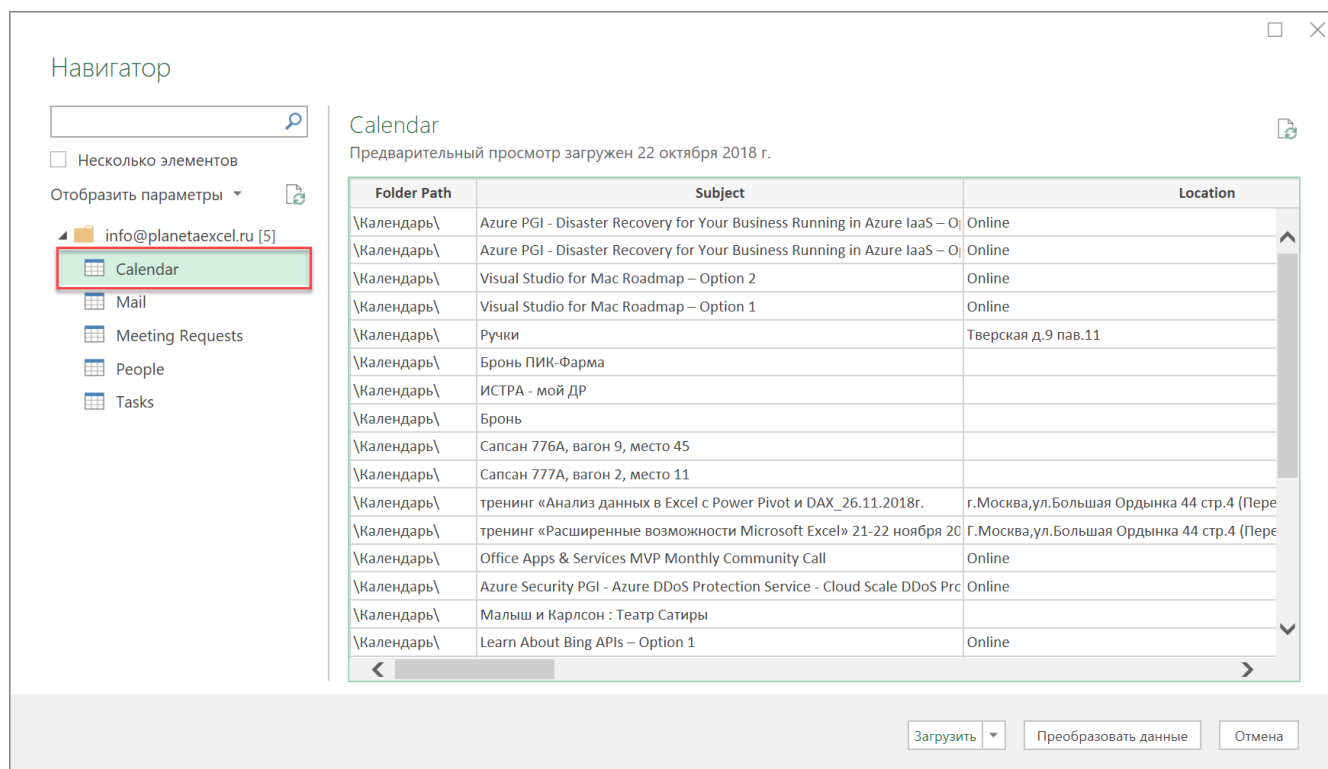
Еще одной приятной возможностью может стать загрузка данных в Excel из вашего почтового ящика Microsoft Exchange, с которым вы работаете через Microsoft Outlook. Такой вариант может быть полезен, например, в следующих сценариях.

- Вы хотите **подсчитать, сколько времени** вы ~~потеряли~~ потратили на совещаниях в этом году. Можно загрузить все ваши встречи и события из Календаря и подсчитать по ним статистику. Очень удобно для хронометража, если вы практикуете тайм-менеджмент.
- При совершении продажи (закрытии сделки, выполнении отгрузки) ваша CRM-система автоматически шлет вам уведомление определенного вида на email. Можно загрузить почту из вашего ящика и подсчитать количество писем, удовлетворяющих определенному условию, т.е. **динамически мониторить продажи**.
- Вам необходимо собрать в одну таблицу **ответы на опрос**, который вы затеяли по электронной почте. Power Query легко может собрать текст заданных писем и подсчитать по ним итоги. Причем можно не волноваться, если кто-то пришлет свой ответ позже, достаточно будет просто обновить запрос.
- Вы создали шаблон сбора бюджетных заявок и разослали его своим коллегам. Коллеги заполняют шаблоны и высылают вам их обратно. Power Query может запросто **собрать вложения** и сформировать на выходе одну общую таблицу с данными из всех присланных книг.

Сразу хочу предупредить, что описанный далее механизм работает только с почтовыми ящиками на серверах Microsoft Exchange (обычно это корпоративная почта) и не актуален для простых почтовых ящиков типа mail.ru, gmail.com, Yandex.ru и им подобных.

Подключиться к вашему ящику можно, выбрав на вкладке **Данные** команду **Получить данные → Из других источников → Из Microsoft Exchange** (Get Data → From other sources → From Microsoft Exchange).

В следующем после этого диалоговом окне вам будет предложено ввести свои реквизиты (адрес и пароль), а потом вы попадете в знакомое уже по прошлым примерам окно **Навигатора (Navigator)**, где будет перечислено всё содержимое вашего почтового ящика:



The screenshot shows the 'Навигатор' (Navigator) window in Excel. On the left, the folder 'Calendar' is selected under the account 'info@planetaexcel.ru [5]'. The main area displays a table of calendar items:

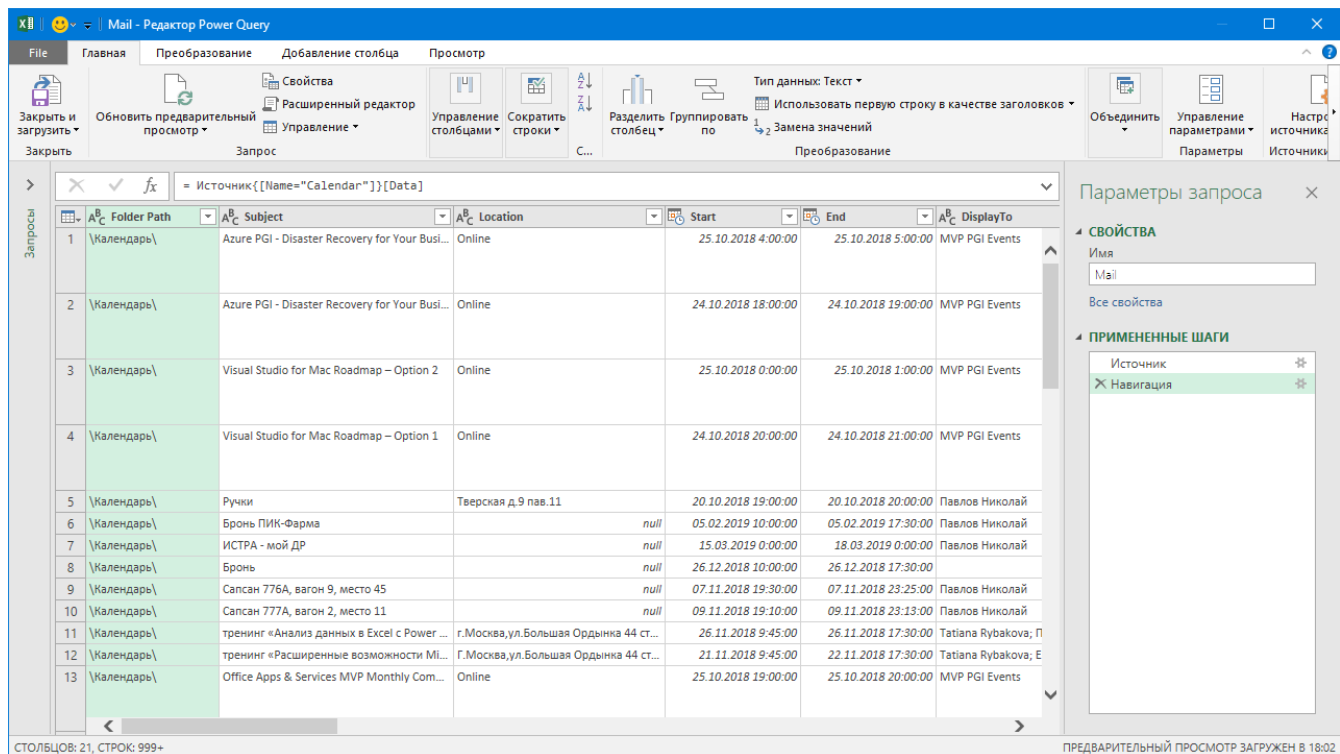
Folder Path	Subject	Location
\Календарь\	Azure PGI - Disaster Recovery for Your Business Running in Azure IaaS – O	Online
\Календарь\	Azure PGI - Disaster Recovery for Your Business Running in Azure IaaS – O	Online
\Календарь\	Visual Studio for Mac Roadmap – Option 2	Online
\Календарь\	Visual Studio for Mac Roadmap – Option 1	Online
\Календарь\	Ручки	Тверская д.9 пав.11
\Календарь\	Бронь ПИК-Фарма	
\Календарь\	ИСТРА - мой ДР	
\Календарь\	Бронь	
\Календарь\	Сапсан 776А, вагон 9, место 45	
\Календарь\	Сапсан 777А, вагон 2, место 11	
\Календарь\	тренинг «Анализ данных в Excel с Power Pivot и DAX_26.11.2018г.	г.Москва,ул.Большая Ордынка 44 стр.4 (Пере
\Календарь\	тренинг «Расширенные возможности Microsoft Excel» 21-22 ноября 2018г.	г.Москва,ул.Большая Ордынка 44 стр.4 (Пере
\Календарь\	Office Apps & Services MVP Monthly Community Call	Online
\Календарь\	Azure Security PGI - Azure DDoS Protection Service - Cloud Scale DDoS Prc	Online
\Календарь\	Малыш и Карлсон : Театр Сатиры	
\Календарь\	Learn About Bing APIs – Option 1	Online

At the bottom of the window, there are buttons for 'Загрузить' (Load), 'Преобразовать данные' (Transform Data), and 'Отмена' (Cancel).

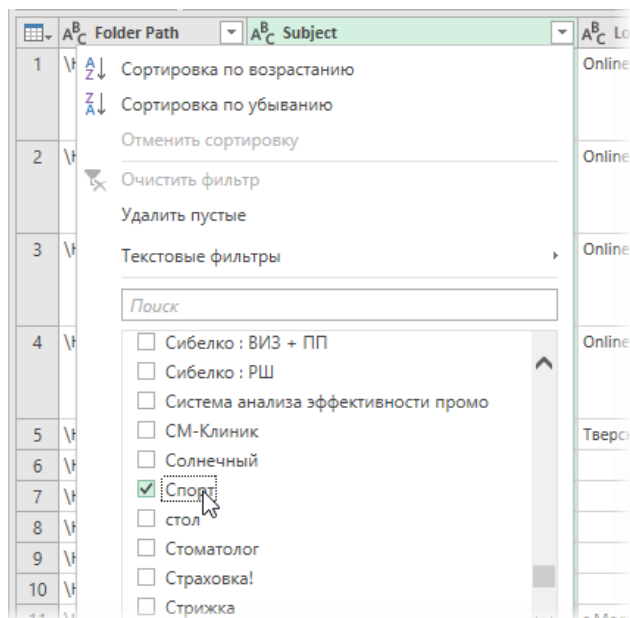
Как видите, здесь отображаются все основные элементы из Microsoft Outlook в виде таблиц: календарь (Calendar), почта (Mail), запросы на собрания (Meeting Requests), ваша адресная книга (People) и список задач (Tasks). Весь дальнейший процесс импорта ничем не отличается от загрузки данных из того же, например, текстового файла или книги Excel.

Давайте для наглядности разберем один пример. Попробуем загрузить данные из календаря, чтобы подсчитать, например, статистику посещений спортзала за последнее время. У меня в Outlook такие встречи помечены как **Спорт**.

Итак, выбираем в Навигаторе **Calendar** и жмем кнопку **Преобразовать данные (Transform Data)** или **Изменить (Edit)** в правом нижнем углу. Спустя какое-то время получаем список всех встреч и событий из календаря в окне редактора запросов Power Query:



Стандартным образом фильтруем нужные нам встречи по столбцу **Subject** и дополнительно при необходимости по любому нужному интервалу дат:



Убираем лишние столбцы, выделив в только нужные и используя правую кнопку мыши и команду **Удалить другие столбцы (Remove Other Columns)**:

	Folder Path	Subject	Location	Start	End
1	\Календарь\	Спорт		00	05.11.2018 15:00:00
2	\Календарь\	Спорт		00	25.10.2018 15:00:00
3	\Календарь\	Спорт		00	22.10.2018 15:00:00
4	\Календарь\	Спорт		00	19.10.2018 12:00:00
5	\Календарь\	Спорт		00	11.10.2018 15:00:00
6	\Календарь\	Спорт		00	08.10.2018 14:00:00
7	\Календарь\	Спорт		00	29.09.2018 12:00:00
8	\Календарь\	Спорт		00	20.09.2018 15:00:00
9	\Календарь\	Спорт		00	24.09.2018 16:00:00
10	\Календарь\	Спорт		00	17.09.2018 14:00:00
11	\Календарь\	Спорт		00	13.09.2018 10:30:00

Если нас интересует длительность каждой встречи, то её можно легко получить, если выделить (удерживая **Ctrl**) сначала столбец **End**, а потом столбец **Start** и выбрать затем на вкладке **Добавить столбец** команду **Время → Вычесть** (Add Column → Time → Subtract):

	Subject	Start	End	Вычитание
1	Спорт	05.11.2018 15:00:00	05.11.2018 16:00:00	0.01:00:00
2	Спорт	25.10.2018 15:00:00	25.10.2018 16:10:00	0.01:10:00
3	Спорт	22.10.2018 15:00:00	22.10.2018 16:30:00	0.01:30:00
4	Спорт	19.10.2018 12:00:00	19.10.2018 13:20:00	0.01:20:00
5	Спорт	11.10.2018 15:00:00	11.10.2018 16:00:00	0.01:00:00
6	Спорт	08.10.2018 14:00:00	08.10.2018 15:30:00	0.01:30:00
7	Спорт	29.09.2018 12:00:00	29.09.2018 13:25:00	0.01:25:00
8	Спорт	20.09.2018 15:00:00	20.09.2018 16:00:00	0.01:00:00
9	Спорт	24.09.2018 16:00:00	24.09.2018 17:15:00	0.01:15:00
10	Спорт	17.09.2018 14:00:00	17.09.2018 15:10:00	0.01:10:00
11	Спорт	13.09.2018 10:30:00	13.09.2018 11:30:00	0.01:00:00
12	Спорт	11.07.2018 17:00:00	11.07.2018 18:00:00	0.01:00:00
13	Спорт	16.07.2018 16:00:00	16.07.2018 17:00:00	0.01:00:00

Ну, и если мы хотим получить общее количество походов в спортзал и их суммарную длительность, то можно применить группировку на вкладке **Преобразование – Группировать по** (Transform – Group by) со следующими настройками:

Группировать по

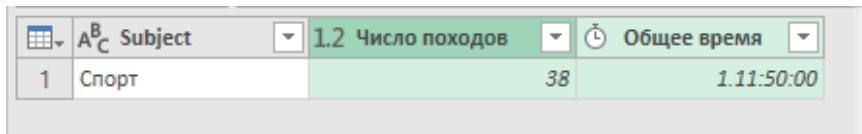
Базовый Подробнее

Укажите столбцы для группировки и желаемые выходные данные.

Группировка

Имя нового столбца Операция Столбец

Получим на выходе компактную таблицу с итогами:



	Subject	Число походов	Общее время
1	Спорт	38	1.11:50:00

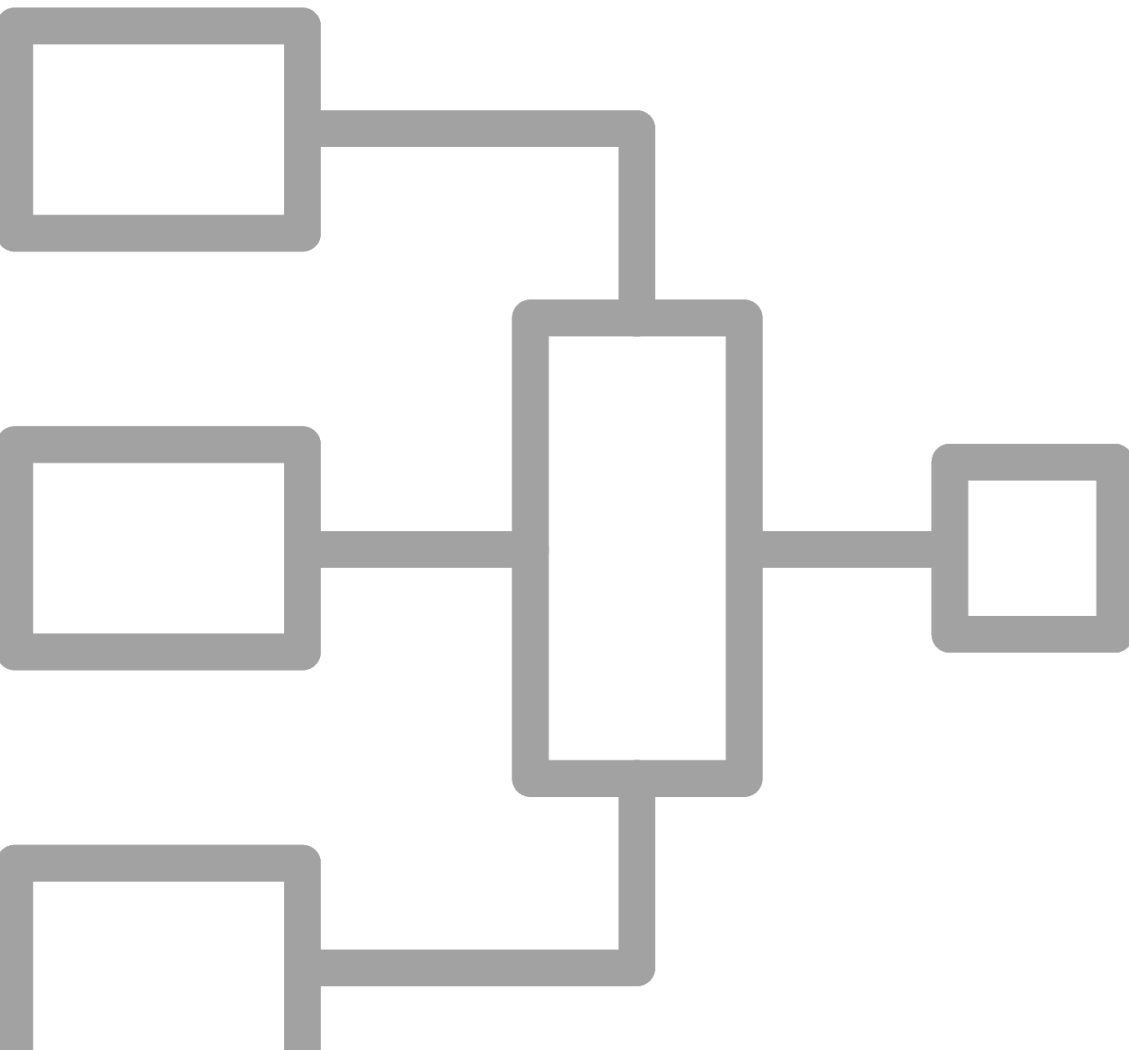
Выходит, что за последние четыре месяца я был в спортзале 38 раз и провел там в сумме 1 день 11 часов и 50 минут. Неплохо!

Единственный отрицательный момент, который хочется честно отметить вдогон, – это не очень высокая скорость общения Power Query с сервером Microsoft Exchange. Если у вас много писем или забитый календарь, то при обновлении такого запроса придется подождать.

Слияние запросов

Мало загрузить таблицы в Power Query, часто их после этого нужно ещё и «поженить» между собой. В этой главе мы подробно разберем способы и техники объединения запросов, а именно:

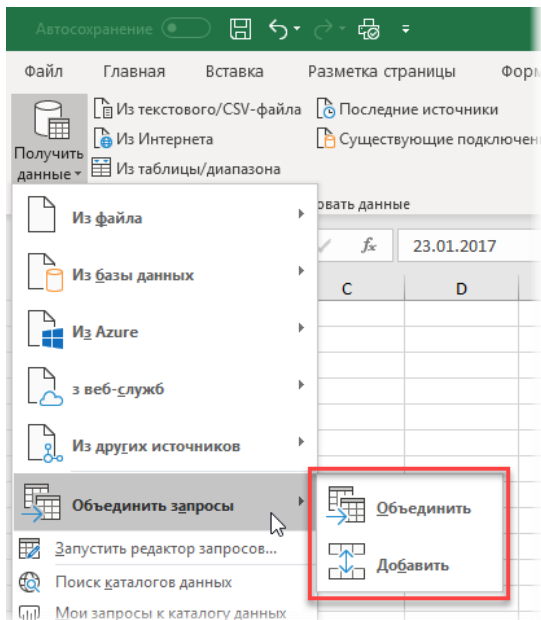
- выясним, какие **два основных типа слияния** запросов в Power Query бывают в принципе и чем они отличаются;
- как выполнять **добавление** таблиц, стыкуя их друг под друга;
- как заменить функцию ВПР (VLOOKUP) на **объединение** запросов в Power Query;
- научимся **сравнивать таблицы** между собой по одному или нескольким столбцам разными способами;
- разберем **шесть типов соединения** таблиц и их применение на практике.



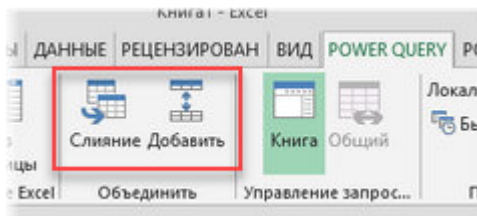
Типы слияния в Power Query

Прежде чем продолжать, необходимо разобраться с двумя основными типами слияния запросов (таблиц), которые бывают в Power Query, – *добавлением* и *объединением*. Обе эти команды доступны:

либо на вкладке **Данные** → **Получить данные** → **Объединить запросы** (Data → Get Data → Combine Queries):



либо в виде двух отдельных кнопок **Слияние** (Merge) и **Добавить** (Append) прямо на вкладке **Power Query**, если он у вас установлен как отдельная надстройка в Excel 2010–2013:



Добавление (Append)

Вы когда-нибудь вручную собирали данные с нескольких листов на один, механически копируя таблицы друг под друга? Это как раз и есть то, что в Power Query называется *добавлением* (*append*) запросов:

Менеджер	Дата сделки	Количество
Александра	09.02.2017	57
Руслан	27.04.2010	26
Елизавета	15.02.2014	55



Менеджер	Количество	Дата сделки
Алина	8	24.11.2011
Мирон	92	08.07.2016

Менеджер	Дата сделки	Количество
Александра	09.02.2017	57
Руслан	27.04.2010	26
Елизавета	15.02.2014	55
Алина	24.11.2011	8
Мирон	08.07.2016	92

- Добавлять друг под друга можно две или более таблиц.
- Важно, чтобы названия соответствующих столбцов совпадали, иначе они не встанут друг под друга.
- Порядок и количество столбцов в каждой таблице могут быть различными.

Объединение (Merge)

Если вы читаете эту книгу, то предполагаю, что в Excel вы уже не новички и должны быть знакомы с «легендарной» функцией **ВПР** (**VLOOKUP**). Если это вдруг почему-то еще не случилось, то очень рекомендую прямо сейчас отложить книгу и потратить 10 минут на её изучение, посмотрев короткое видео и прочитав статью, например, тут: <https://www.planetaexcel.ru/techniques/25/106/>. Я серьезно: не продолжайте читать, пока это не сделаете!

Дело в том, что второй тип слияния – *объединение* (*merge*) – это как раз то, что делает функция **ВПР** (**VLOOKUP**). Технически это поиск и подстановка данных из одной таблицы в другую по совпадению заданного параметра:

Продавец	Товар	Количество
Диана	Инжир	2
Антон	Оливки	5
Злата	Финики	1
Елена	Ячмень	3



Продукт	Цена за кг
Ячмень	72
Авокадо	61
Персик	95
Цветная капуста	11
Оливки	52
Имбирь	30
Куриные яйца	88
Финики	56
Орехи кешью	69
Салат зеленый	45
Инжир	68



Продавец	Товар	Количество	Цена
Диана	Инжир	2	68
Антон	Оливки	5	52
Злата	Финики	1	56
Елена	Ячмень	3	72

При этом Power Query делает это на порядок проще и красивее, чем ВПР.

- Имена столбцов и их порядок в таблицах не играют роли.
- Можно объединять таблицы по совпадению одного или нескольких значений из нескольких столбцов.
- Power Query умеет делать несколько вариантов объединения (левое внешнее, правое внешнее, внутреннее и т. д.), что приближает нас по возможностям уже к полноценной базе данных (аналог команды JOIN в SQL).
- Объединять можно только две таблицы за раз.

Добавление двух таблиц

Давайте начнем со слияния добавлением, т. к. оно технически и логически проще. Предположим, что у нас есть две таблицы с продажами по Москве и Калуге вот такого вида:

	A	B	C	D	E	F	G	H	I	J
1	Москва					Калуга				
2										
3	Менеджер	Товар	Стоимость	Дата		товар	Менеджер	Дата сделки	Стоимость	
4	Дмитрий	Капуста	2 806	29.03.2017		Киви	Мария	28.03.2017	7 886	
5	Сергей	Капуста	6 052	22.03.2017		Земляника	Мария	12.02.2017	112	
6	Елена	Яблоко	3 474	07.03.2017		Картофель	Андрей	20.01.2017	5 895	
7	Дмитрий	Капуста	9 109	21.03.2017		Овес	Сергей	21.01.2017	539	
8	Елена	Просо	371	12.01.2017		Просо	Анна	03.03.2017	2 500	
9	Елена	Дыня	6 455	26.01.2017		Картофель	Елена	19.01.2017	3 313	
10	Анастасия	Картофель	5 469	15.02.2017		Апельсин	Анастасия	27.02.2017	7 661	
11	Анастасия	Яблоко	6 006	10.01.2017		Дыня	Анна	03.02.2017	3 632	
12	Елена	Апельсин	3 532	18.01.2017		Овес	Мария	23.01.2017	547	
13	Елена	Фасоль	7 924	25.01.2017		Киви	Мария	16.03.2017	9 692	
14	Сергей	Капуста	715	13.01.2017		Помидор	Мария	13.01.2017	9 464	
15	Дарья	Просо	3 818	07.03.2017		Авокадо	Елена	02.01.2017	7 903	
16	Анна	Яблоко	1 954	03.01.2017		Просо	Мария	24.01.2017	1 195	
17	Андрей	Картофель	9 584	09.03.2017		Свекла	Елена	28.01.2017	9 869	
18						Шпинат	Андрей	09.02.2017	5 068	
19						Нут	Елена	18.01.2017	5 187	
20						Рис	Елена	23.02.2017	9 560	
21						Спаржа	Максим	16.03.2017	6 624	
22						Свекла	Анастасия	08.02.2017	1 759	
23						Лосось	Максим	27.02.2017	6 746	
24						Дыня	Дмитрий	17.01.2017	2 485	
25						Тыква	Максим	13.02.2017	7 549	
26										
27										

Обратите внимание на следующее:

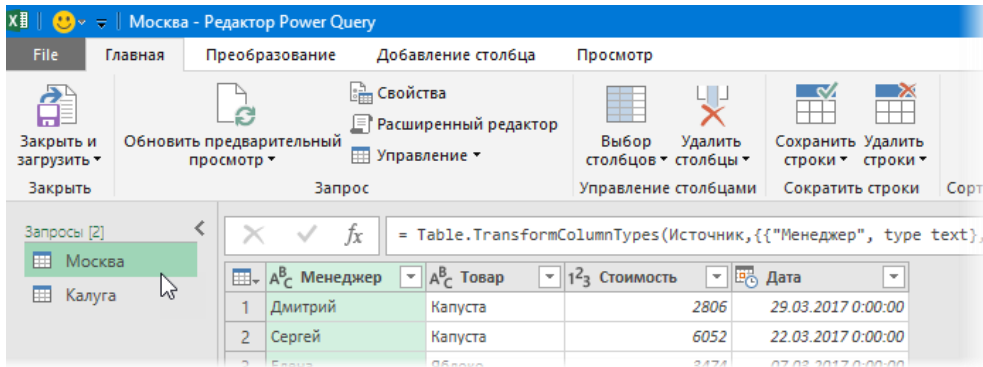
- таблицы разного размера;
- последовательность столбцов в них различается;
- названия столбцов точно не совпадают.

Допустим, что на выходе нам нужно построить сводную таблицу, которая отражала бы общие данные из обеих таблиц – суммарную стоимость продаж по товарам и менеджерам в каждом городе. Если вы раньше сталкивались со сводными, то, наверное, помните, что в Excel нормальную полноценную сводную можно построить только по одной таблице, а не по нескольким. Поэтому наша первая задача – объединить эти две таблицы в одну, состыковав их в одно целое.

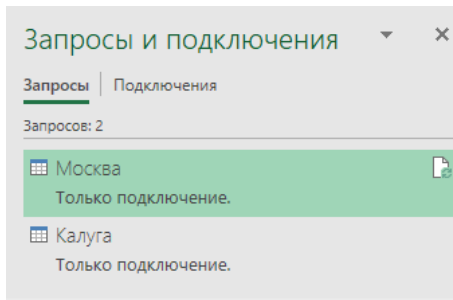
Если есть вероятность, что в будущем размер списков может меняться, т. к. к ним могут дописывать новые сделки-строки, то правильнее будет сначала конвертировать наши таблицы в «умные», а потом импортировать их в Power Query, как мы делали в предыдущей главе. Для этого для каждой из таблиц по очереди проделываем следующие действия:

1. Встаём в любую ячейку таблицы и превращаем её в «умную» сочетанием клавиш **Ctrl+T** или используя кнопку **Форматировать как таблицу** на вкладке **Главная** (Home → Format as Table). При желании можно дать таблице понятное имя на вкладке **Конструктор** (Design), назвав их *Москва* и *Калуга* соответственно.
2. Жмём кнопку **Из таблицы / диапазона** на вкладке **Данные** (Data → From Table / Range), чтобы загрузить созданную «умную» таблицу в Power Query.
3. После загрузки в редактор запросов выбираем команду **Закреть и загрузить → Закреть и загрузить в... → Только создать подключение** (Close & Load → Close & Load to... → Only Create Connection), т. к. выгружать их обратно на лист, дублируя исходные данные, нам не требуется.

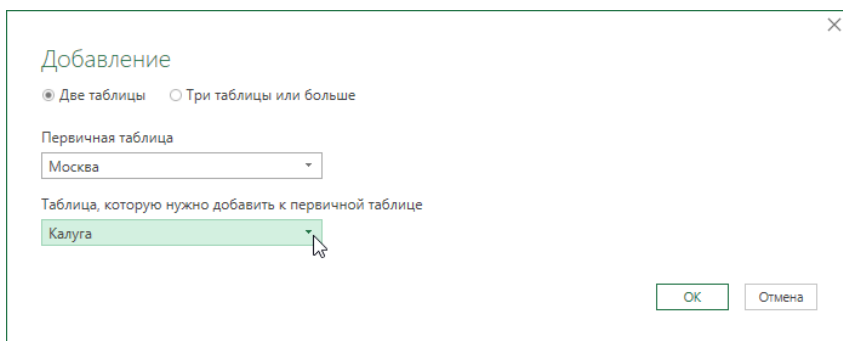
В итоге мы должны получить два запроса с нашими данными, между которыми можно легко переключаться в левой панели **Запросы (Queries)** окна Power Query:



или в правой панели **Запросы и подключения** (Queries & Connections) в окне Microsoft Excel:



Теперь давайте соединим наши запросы в единое целое. Для этого выбираем на вкладке **Данные** → **Получить данные** → **Объединить запросы** → **Добавить** (Data → Combine queries → Append). В открывшемся окне выберем из выпадающих списков имена первичной и вторичной таблиц и нажмем на **ОК**:



На выходе получим еще один запрос со стандартным именем **Append**:

	A^B Менеджер	A^B Товар	1^2 Стоимость	Дата	A^B товар	Дата сделки
1	Дмитрий	Капуста	2806	29.03.2017 0:00:00	null	null
2	Сергей	Капуста	6052	22.03.2017 0:00:00	null	null
3	Елена	Яблоко	3474	07.03.2017 0:00:00	null	null
4	Дмитрий	Капуста	9109	21.03.2017 0:00:00	null	null
5	Елена	Просо	371	12.01.2017 0:00:00	null	null
6	Елена	Дыня	6455	26.01.2017 0:00:00	null	null
7	Анастасия	Картофель	5469	15.02.2017 0:00:00	null	null
8	Анастасия	Яблоко	6006	10.01.2017 0:00:00	null	null
9	Елена	Апельсин	3532	18.01.2017 0:00:00	null	null
10	Елена	Фасоль	7924	25.01.2017 0:00:00	null	null
11	Сергей	Капуста	715	13.01.2017 0:00:00	null	null
12	Дарья	Просо	3818	07.03.2017 0:00:00	null	null
13	Анна	Яблоко	1954	03.01.2017 0:00:00	null	null
14	Андрей	Картофель	9584	09.03.2017 0:00:00	null	null
15	Мария	null	7886	null	Киви	28.03.2017 0:00:00
16	Мария	null	112	null	Земляника	12.02.2017 0:00:00
17	Андрей	null	5895	null	Картофель	20.01.2017 0:00:00
18	Сергей	null	539	null	Овес	21.01.2017 0:00:00
19	Анна	null	2500	null	Просо	03.03.2017 0:00:00

С ходу видно проблемы со столбцами **Товар** и **Дата сделки**, которые не встали на положенное им место. Как легко сообразить, это произошло потому, что имена столбцов не совпали (с точностью до регистра). При этом различная последовательность столбцов в исходных таблицах, напротив, совершенно не помешала: столбец **Менеджер**, например, замечательно состыковался в обеих таблицах.

Исправить ситуацию можно легко: достаточно вернуться в запрос **Калуга** и переименовать соответствующие колонки **товар** в «**Товар**» и **Дата сделки** в «**Дата**» двойным щелчком левой кнопки мыши по заголовку столбца. Вернувшись после этого в запрос **Append**, мы сразу увидим последствия переименования: всё встанет на место:

	АВ Менеджер	АВ Товар	1 ² 3 Стоимость	Дата
1	Дмитрий	Капуста	2806	29.03.2017 0:00:00
2	Сергей	Капуста	6052	22.03.2017 0:00:00
3	Елена	Яблоко	3474	07.03.2017 0:00:00
4	Дмитрий	Капуста	9109	21.03.2017 0:00:00
5	Елена	Просо	371	12.01.2017 0:00:00
6	Елена	Дыня	6455	26.01.2017 0:00:00
7	Анастасия	Картофель	5469	15.02.2017 0:00:00
8	Анастасия	Яблоко	6006	10.01.2017 0:00:00
9	Елена	Апельсин	3532	18.01.2017 0:00:00
10	Елена	Фасоль	7924	25.01.2017 0:00:00
11	Сергей	Капуста	715	13.01.2017 0:00:00
12	Дарья	Просо	3818	07.03.2017 0:00:00
13	Анна	Яблоко	1954	03.01.2017 0:00:00
14	Андрей	Картофель	9584	09.03.2017 0:00:00
15	Мария	Киви	7886	28.03.2017 0:00:00
16	Мария	Земляника	112	12.02.2017 0:00:00
17	Андрей	Картофель	5895	20.01.2017 0:00:00
18	Сергей	Овес	539	21.01.2017 0:00:00
19	Анна	Просо	2500	03.03.2017 0:00:00
20	Елена	Картофель	3313	19.01.2017 0:00:00

Единственный отрицательный момент в том, что мы потеряли информацию о городах: совершенно неясно, из какой именно таблицы (города) какая строка в нашей сборке. Как же быть, если нужно сравнить продажи по городам? Эту задачу можно решить добавлением к нашим исходным таблицам дополнительного вычисляемого столбца с именем города. Для этого вернемся в запрос **Москва** и на вкладке **Добавить столбец** нажмем на кнопку **Настраиваемый столбец** (Add Column → Custom Column):

The screenshot shows the Power Query Editor interface. The ribbon is set to 'Добавление столбца' (Add Column). The 'Настраиваемый столбец' (Custom Column) button is highlighted with a red circle and the number '2'. The dialog box 'Настраиваемый столбец' (Custom Column) is open, showing the following fields:

- Имя нового столбца (New column name):
- Пользовательская формула столбца (Custom column formula):
- Доступные столбцы (Available columns): Менеджер, Товар, Стоимость, Дата

At the bottom of the dialog box, there is a message: 'Синтаксические ошибки не обнаружены.' (No syntax errors detected.) and buttons for 'OK' and 'Отмена' (Cancel).

В открывшемся окне введем имя нового столбца (**Город**) и его формулу:

= "Москва"

После нажатия на **ОК** должен получиться столбец с повторяющимися значениями города:

	А ^В Менеджер	А ^В Товар	1 ² Стоимость	Дата	А ^В Город
1	Дмитрий	Капуста	2806	29.03.2017 0:00:00	Москва
2	Сергей	Капуста	6052	22.03.2017 0:00:00	Москва
3	Елена	Яблоко	3474	07.03.2017 0:00:00	Москва
4	Дмитрий	Капуста	9109	21.03.2017 0:00:00	Москва
5	Елена	Просо	371	12.01.2017 0:00:00	Москва
6	Елена	Дыня	6455	26.01.2017 0:00:00	Москва

Затем нужно повторить аналогичную операцию со второй таблицей. После этого в итоговом запросе **Append** автоматически появится наш столбец **Город** со словами *Москва* или *Калуга* в каждой строке.

Теперь с помощью команды **Закрыть и загрузить (Close & Load)** можно выгрузить получившуюся таблицу на лист Excel или же оставить её как подключение, которое затем выбрать при построении сводной таблицы, как мы уже делали (см. главу [Построение сводной таблицы по результатам запроса](#)). Тогда мы фактически построим сводную по двум таблицам одновременно:

2					
3	Сумма по полю Стоимость	Назван			
4		январь	фев	мар	Общий итог
5	Названия строк				
6	Калуга	46397	42087	26702	115186
7	Анастасия		9420		9420
8	Андрей	5895	5068		10963
9	Анна		3632	2500	6132
10	Дмитрий	2485			2485
11	Елена	26272	9560		35832
12	Максим		14295	6624	20919
13	Мария	11206	112	17578	28896
14	Сергей	539			539
15	Москва	26957	5469	34843	67269
16	Анастасия	6006	5469		11475
17	Андрей			9584	9584
18	Анна	1954			1954
19	Дарья			3818	3818
20	Дмитрий			11915	11915
21	Елена	18282		3474	21756
22	Сергей	715		6052	6767
23	Общий итог	73354	47556	61545	182455
24					

Причем если в будущем к исходным «умным» таблицам будут дописаны новые строки, то они автоматически «растянутся», и достаточно будет просто обновить наши запросы и сводную, нажав кнопку **Обновить всё** на вкладке **Данные (Data → Refresh All)** или сочетание клавиш **Ctrl+Alt+F5**.

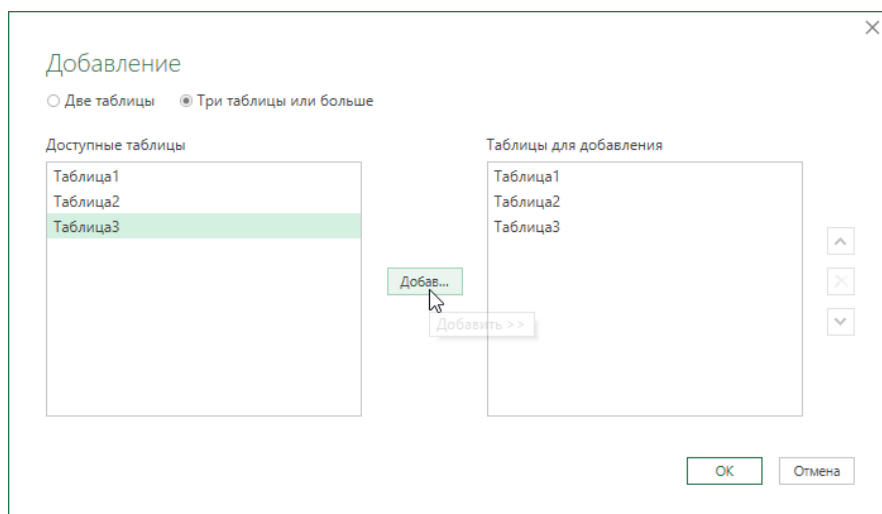
ПРИМЕЧАНИЕ: Начиная с 2016 версии Microsoft Excel при построении сводной автоматически группирует даты-дни в месяцы-кварталы-годы. В приведенной выше сводной таблице поле **Дата** было просто помещено в область столбцов сводной таблицы и сразу автоматически сгруппировалось таким образом. Если у вас более старые версии Excel, то группировку придется сделать вручную, щелкнув по любой дате в области столбцов и выбрав команду **Группировать (Group)**. Там же рядом находится и команда **Разгруппировать (Ungroup)** – для обратного преобразования в дни.

Добавление трех и более таблиц с загрузкой в Модель Данных

Поскольку Power Query, в отличие от Excel, не ограничен миллионом строк, то мы можем спокойно собирать в единое целое таблицы, чей общий размер превышает размер листа. Допустим, что у нас есть три «умные» таблицы, чей суммарный размер составляет 2 млн строк:

Дата	Менеджер	Товар	Стоимость	Дата	Товар	Менеджер	Стоимость	Дата	Менеджер	Товар	Стоимость
06.12.2015	Елена	Земляника	470	08.12.2015	Земляника	Сергей	2771	03.12.2016	Дмитрий	4547	Картофель
03.09.2017	Дмитрий	Дыня	2476	31.10.2016	Капуста	Анастасия	16	21.03.2016	Анастасия	4223	Просо
21.12.2017	Сергей	Капуста	2104	05.05.2017	Дыня	Елена	94	02.07.2015	Мария	943	Дыня
11.04.2015	Анастасия	Земляника	349	23.03.2015	Лосось	Анна	1736	20.06.2015	Елена	4477	Яблоко
27.02.2016	Дмитрий	Яблоко	2040	18.04.2015	Яблоко	Дмитрий	2249	18.03.2015	Александр	4692	Лосось
06.03.2017	Игорь	Капуста	2252	26.10.2017	Яблоко	Александр	2201	16.11.2017	Елена	98	Яблоко
02.06.2017	Игорь	Капуста	2076	30.10.2017	Яблоко	Сергей	1432	02.10.2017	Елена	11	Яблоко
27.09.2017	Максим	Просо	280	08.05.2016	Земляника	Александр	1790	18.10.2015	Анастасия	11	Яблоко
15.10.2016	Дмитрий	Яблоко	452	09.12.2016	Картофель	Максим	2627	15.09.2017	Мария	877	Фасоль
21.01.2016	Сергей	Яблоко	1151	02.02.2017	Просо	Елена	198	24.08.2017	Мария	4787	Яблоко
25.01.2015	Сергей	Яблоко	2724	03.06.2015	Картофель	Елена	1800	16.05.2016	Дмитрий	1	Капуста
22.03.2016	Анастасия	Яблоко	1057	31.07.2016	Лосось	Елена	361	14.11.2016	Анастасия	1	Капуста
09.08.2016	Анна	Дыня	2712	25.06.2015	Фасоль	Максим	2637	30.05.2015	Анна	934	Капуста
22.04.2015	Анастасия	Яблоко	2344	07.02.2017	Просо	Мария	1747	06.07.2016	Мария	4537	Лосось
04.05.2016	Максим	Капуста	2967	09.02.2017	Дыня	Анна	2164	13.08.2017	Дарья	322	Фасоль
24.01.2016	Андрей	Просо	1606	11.12.2015	Просо	Анна	2988	16.05.2017	Елена	4126	Яблоко
23.07.2015	Андрей	Лосось	982	08.03.2017	Брокколи	Дмитрий	2351	13.12.2016	Дмитрий	1423	Земляника

Последовательность столбцов в этих таблицах различается, но имена столбцов идентичны, а это главное условие для успешной сборки, как мы выяснили в прошлой главе. После выбора на вкладке **Данные** команды **Получить данные** → **Объединить запросы** → **Добавить** (Data → Combine Queries → Append) нужно будет переключиться в режим **Три таблицы или больше** (Three or more tables) и выбрать все таблицы для консолидации из имеющегося списка:

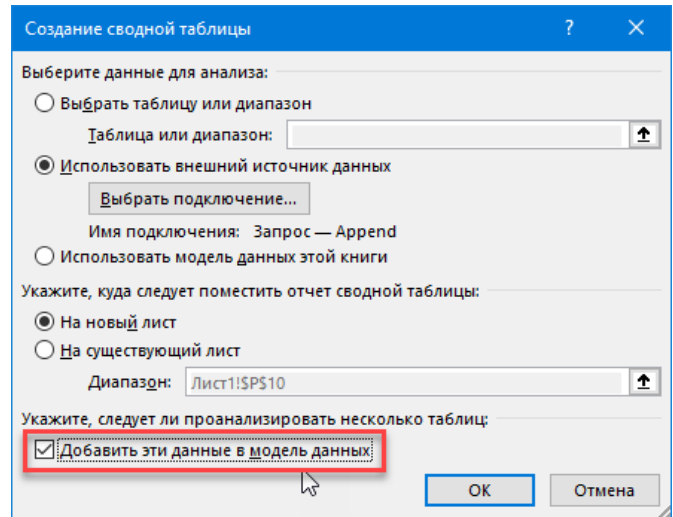
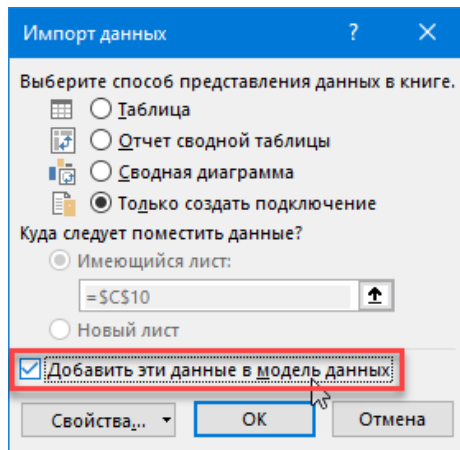


ПРИМЕЧАНИЕ: В старых версиях Power Query не было селектора **Две таблицы / Три таблицы или больше**, он появился с одним из обновлений в 2017 году. Если эта возможность у вас отсутствует, то придется либо делать два запроса с последовательными попарными добавлениями, либо обновить вашу версию Excel и Power Query.

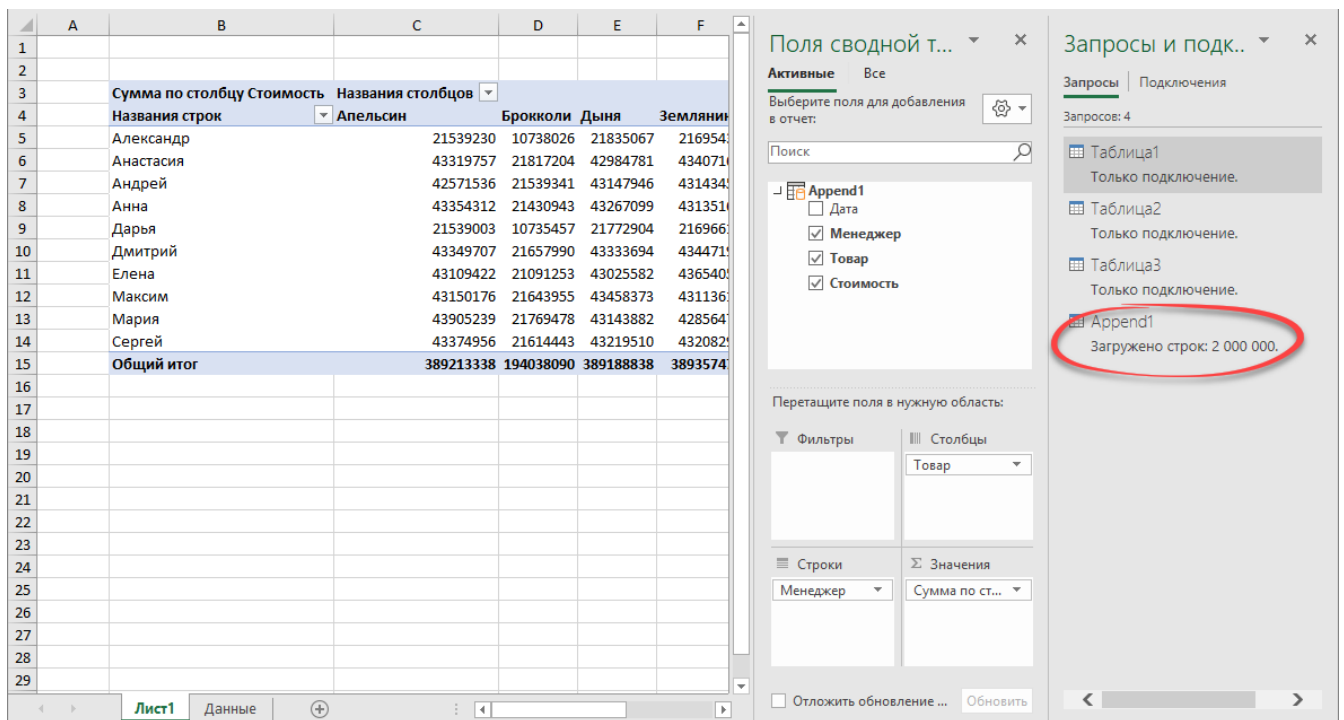
После нажатия на **ОК** мы получим уже знакомый нам запрос с именем **Append**, который склеивает друг с другом наши исходные таблицы. Дальше начинается самое интересное.

При выгрузке созданного запроса в Excel с помощью команды **Главная** → **Закреть и загрузить** → **Закреть и загрузить в...** у нас уже нет особого выбора: на лист в виде таблицы мы поместить их не сможем, т. к. число строк в результате превышает количество строк на листе Excel (1048567). Поэтому хочешь не хочешь, а придется выбрать вариант **Только создать подключение** (Create Only Connection).

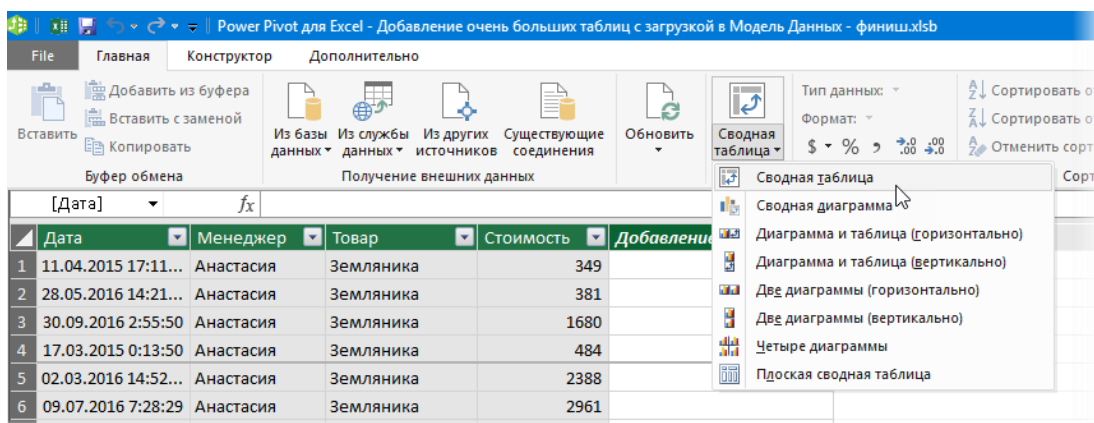
В принципе, можно на этом и остановиться и пойти дальше по проторенному пути из прошлой главы – создать сводную таблицу по созданному подключению. Однако при анализе такого большого объема данных во многих случаях удобнее воспользоваться возможностями надстройки Power Pivot и загрузить все данные в неё. Это можно сделать, либо включив флажок **Добавить эти данные в модель данных** (Add this data to Data Model) в окне выгрузки данных в Power Query, либо включив такой же флажок непосредственно при построении сводной таблицы:



Если вы выбрали второй способ, то дальше всё происходит по стандартному сценарию: ждем загрузки данных, видим феерическую фразу «Загружено 2 000 000 строк» и строим сводную таблицу обычным образом:



Если же вы загрузили данные в Модель Данных при выгрузке из Power Query, то дальше нужно открыть окно Power Pivot с помощью кнопки **Управление моделью данных (Manage Data Model)** на вкладке **Данные (Data)** и построить затем сводную оттуда, используя выпадающий список **Сводная таблица (Pivot Table)**:



Объединение таблиц: забудьте про ВПР

*Любая достаточно развитая технология неотличима от волшебства.
(Артур Кларк)*

Теперь давайте на примерах рассмотрим второй тип слияния запросов в Power Query – объединение (merge). Предположим, что у нас имеются три таблицы: с данными по продажам, прайс-лист на товары и справочник по регионам и менеджерам:

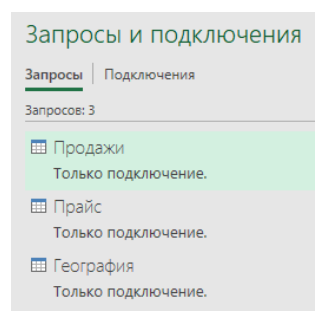
Продажи				Прайс			География			
Дата	Товар	Количество	Продавец	Продукт	Цена	Скидка	Город	Менеджер	Статус	Регион
17.10.2017	Джемпер	1	Дмитрий	Джемпер	9055	0%	Новосибирск	Дмитрий	менеджер	Сибирь
02.06.2017	Брюки	3	Дмитрий	Брюки	4625	10%	Екатеринбург	Елена	менеджер	Сибирь
15.11.2017	Кардиган	4	Елена	Кардиган	5810	0%	Санкт-Петербург	Андрей	менеджер	Запад
01.10.2017	Свитер	3	Андрей	Свитер	7925	0%	Москва	Анастасия	стажер	Центр
25.01.2017	джинсы	3	Анастасия	Джинсы	6435	0%	Москва	Анна	менеджер	Центр
22.04.2017	Кофта	4	Дмитрий	Кофта	735	0%	Екатеринбург	Сергей	менеджер	Сибирь
17.11.2017	Пуловер	4	Анна	Пуловер	5225	0%	Новосибирск	Максим	стажер	Сибирь
27.11.2017	Свитер	2	Елена	Юбка	8980	20%	Екатеринбург	Александр	менеджер	Сибирь
16.10.2017	Юбка	4	Сергей	Блуза	7900	0%	Екатеринбург	Мария	стажер	Сибирь
12.12.2017	Блуза	4	Анастасия	Комбинезон	4940	0%	Санкт-Петербург	Дарья	стажер	Запад
24.07.2017	Кардиган	1	Максим	Рубашка	9285	5%				
20.05.2017	Комбинезон	3	Елена	Водолазка	9255	0%				
23.10.2017	Джемпер	2	Дмитрий	Бриджи	2220	5%				
20.11.2017	Рубашка	2	Максим	Лосины	9070	0%				
04.11.2017	Джинсы	4	Анна	Шорты	8150	15%				
09.05.2017	Кофта	1	Андрей							
23.09.2017	Брюки	2	Александр							
23.07.2017	Кардиган	2	Дмитрий							
22.08.2017	Джинсы	1	Анастасия							
23.02.2017	Водолазка	1	Максим							

На выходе необходимо посчитать суммарную выручку по каждому товару (с учетом скидок) для каждого региона. При этом количество проданного у нас в первой таблице, цены надо подтягивать из второй, а информация о регионах для каждого менеджера есть только в третьей. Если решать эту задачу классическим путем, то пришлось бы как минимум к первой таблице добавлять пару вычисляемых столбцов с функциями **ВПР (VLOOKUP)**, **ПОИСКПОЗ (MATCH)** и **ИНДЕКС (INDEX)**, а потом вычислять итоги с помощью сводной таблицы. Давайте посмотрим, как всё вышеперечисленное можно изящно сделать не формулами, а через Power Query.

Загружаем все таблицы как подключения

Сначала превратим таблицы в «умные», дадим им понятные имена (я назвал их *Продажи*, *Прайс* и *География* соответственно) и загрузим их в Power Query по очереди, используя кнопку **Из таблицы / диапазона** на вкладке **Данные (Data → From Table / Range)** и вернемся потом обратно в режиме **Только создать подключение (Only Create Connection)**.

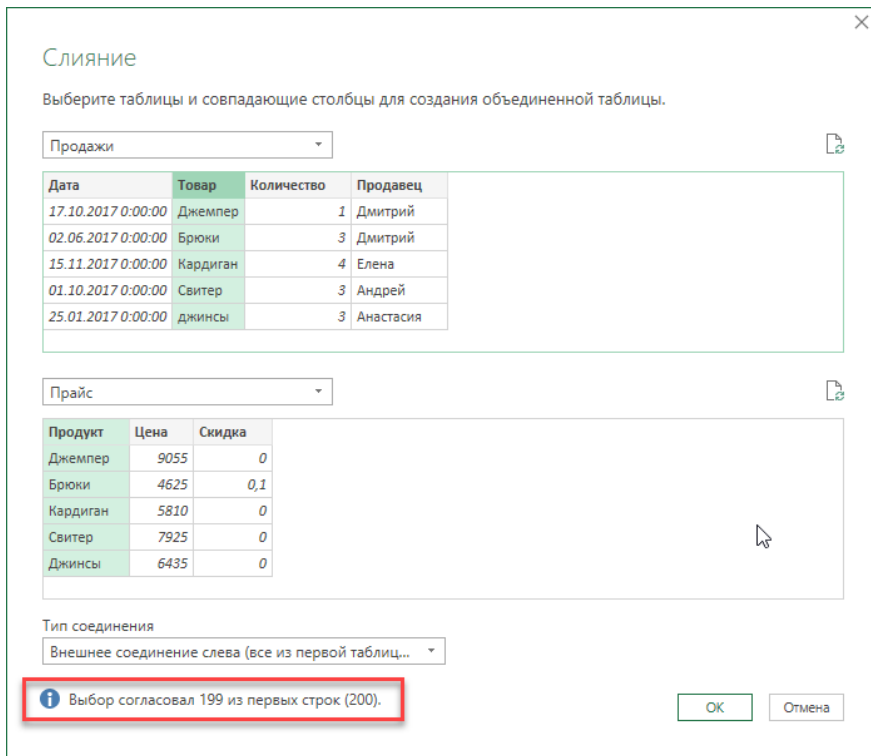
Чуть позже в главе **Загрузка «умных» таблиц в Power Query макросом** мы научимся быстро делать массовую загрузку сразу всех «умных» таблиц из книги в Power Query, но пока этот относительно скучный шаг надо проделать вручную.



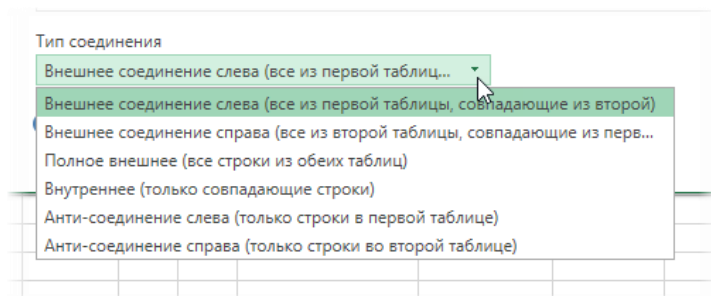
Выполняем слияние

Чтобы добавить цены из прайса к таблице продаж, на вкладке **Данные** выберем команду **Получить данные → Объединение запросов → Объединить (Data → Get Data → Combine queries → Merge)**.

В открывшемся окне сверху выберем таблицу, к которой мы хотим добавить данные (*Продажи*), а снизу таблицу, откуда мы хотим их подставить (*Прайс*). Затем выделим мышкой столбцы *Товар* и *Продукт* в обеих таблицах, по которым должен идти поиск и подстановка:



Обратите внимание, что в нижней части окна есть выпадающий список, где можно выбрать тип соединения:



Чуть позже мы подробно разберем каждый вариант, а пока что достаточно знать, что первый пункт из этого списка – **Внешнее соединение слева (Left Outer Join)** – это как раз то, что делает функция **ВПР**, и то, что нам нужно в этом примере.

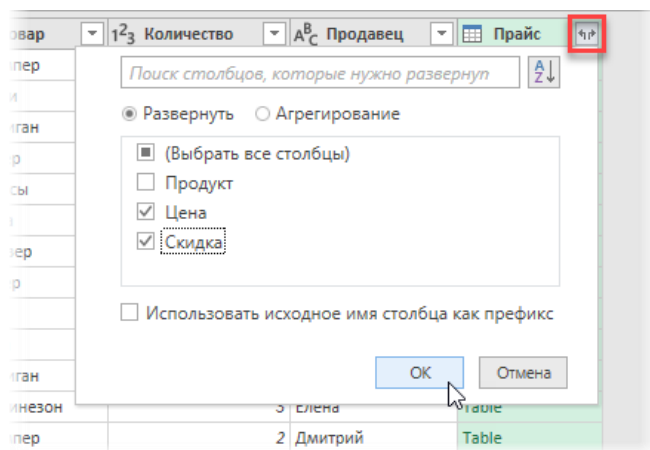
Ещё один важный нюанс в том, что внизу окна Power Query напишет фразу «**Выбор согласовал 199 из первых строк (200)**». В переводе на человеческий язык это означает, что он смог найти только 199 цен к 200 наших исходных сделок, т. е. один из товаров в таблице *Продажи* был не найден в прайс-листе. Чуть позже мы разберемся и с этим, а пока нажмем **OK**.

Должно открыться окно редактора запросов, где мы увидим новый запрос со стандартным именем *Merge1*, в котором будет наша исходная таблица продаж с добавленным к ней столбцом *Прайс*. В каждой ячейке этого столбца будет таблица с фрагментом прайс-листа, соответствующим данному товару. Увидеть её содержимое можно, щелкнув мышью в белый фон ячейки со словом **Table** (но не в слово **Table!**):

	Дата	Товар	Количество	Продавец	Прайс
1	17.10.2017 0:00:00	Джемпер		1 Дмитрий	Table
2	02.06.2017 0:00:00	Брюки		3 Дмитрий	Table
3	15.11.2017 0:00:00	Кардиган		4 Елена	Table
4	01.10.2017 0:00:00	Свитер		3 Андрей	Table
5	25.01.2017 0:00:00	джинсы		3 Анастасия	Table
6	22.04.2017 0:00:00	Кофта		4 Дмитрий	Table
7	17.11.2017 0:00:00	Пуловер		4 Анна	Table
8	27.11.2017 0:00:00	Свитер		2 Елена	Table
9	16.10.2017 0:00:00	Юбка		4 Сергей	Table
10	12.12.2017 0:00:00	Блуза		4 Анастасия	Table
11	24.07.2017 0:00:00	Кардиган		1 Максим	Table

Продукт	Цена	Скидка
Брюки	4625	0,1

Теперь развернем вложенные таблицы, используя кнопку с двойными стрелками в заголовке столбца *Прайс*. Из выпадающего списка можно выбрать те столбцы прайс-листа, которые мы хотим подставить, и снять флажок **Использовать исходное имя столбца как префикс** (Use original column name as prefix), чтобы новые столбцы назывались просто *Цена* и *Скидка*, а не *Прайс.Цена* и *Прайс.Скидка*:



После нажатия на **ОК** мы достигнем желанной цели: к нашей таблице продаж добавятся колонки с ценами и скидками из прайс-листа:

	Дата	Товар	Количество	Продавец	Цена	Скидка
1	17.10.2017 0:00:00	Джемпер		1 Дмитрий	9055	0
2	23.10.2017 0:00:00	Джемпер		2 Дмитрий	9055	0
3	02.06.2017 0:00:00	Брюки		3 Дмитрий	4625	0,1
4	15.11.2017 0:00:00	Кардиган		4 Елена	5810	0
5	24.07.2017 0:00:00	Кардиган		1 Максим	5810	0
6	01.10.2017 0:00:00	Свитер		3 Андрей	7925	0
7	27.11.2017 0:00:00	Свитер		2 Елена	7925	0
8	25.01.2017 0:00:00	джинсы		3 Анастасия	null	null
9	04.11.2017 0:00:00	Джинсы		4 Анна	6435	0
10	22.04.2017 0:00:00	Кофта		4 Дмитрий	735	0
11	09.05.2017 0:00:00	Кофта		1 Андрей	735	0

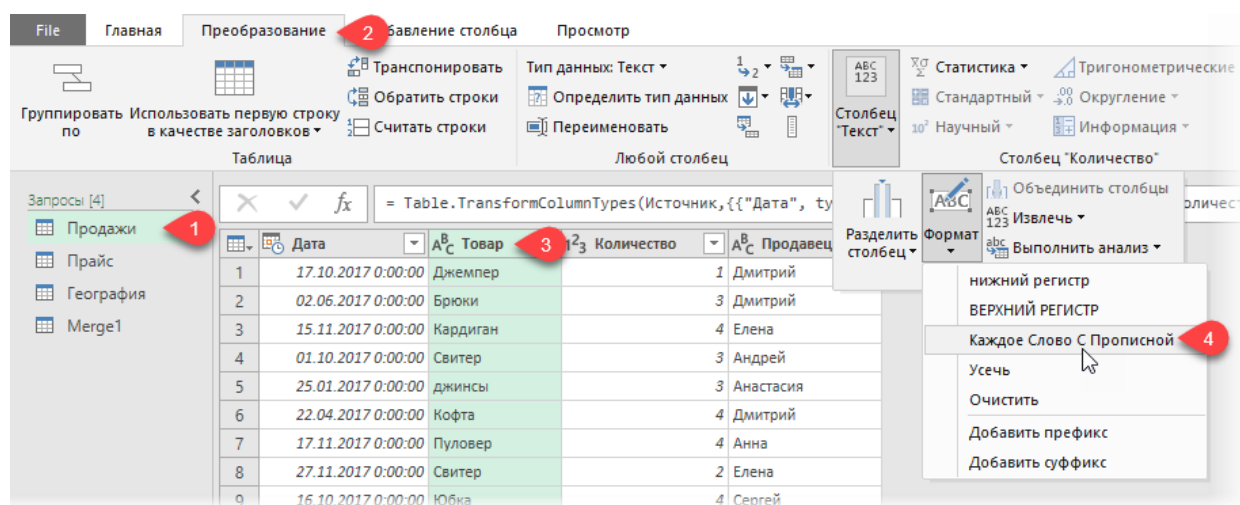
Исправляем ошибки

Думаю, вам известно, что когда функция **ВПР** (VLOOKUP) не находит искомого значения, то она выдает ошибку **#Н/Д** (#N/A). Power Query же реагирует на подобные ситуации по-другому, что хорошо видно в 8-й строке нашей итоговой таблицы. Поскольку (в отличие от **ВПР**) объединение происходит с учётом регистра, «джинсы» с маленькой буквы не нашлись в прайс-листе (где они есть, но с заглавной), и напротив них в таблице появилось значение **null**.

null – это специальное слово (и тип данных) в Power Query, обозначающее пустоту¹. В отличие от Excel, где пустая ячейка может быть чем угодно – от формулы, выводящей пустую строку "", до форматирования «белым на белом», в Power Query этот вопрос имеет однозначный ответ: **null** – значит пусто.

Вопрос, что с этими **null** теперь делать. Тут есть три варианта.

1. **Ничего не делать.** Ячейки с **null** в Power Query потом на листе Excel превратятся просто в пустые ячейки.
2. **Заменить **null** на что-то полезное.** Щёлкнув правой кнопкой мыши по заголовку столбца с ошибками, можно выбрать в контекстном меню команду **Заменить ошибки (Replace Errors)** и ввести значение, на которое вы хотите их заменить (например, на 0 или на «здесь ошибка»).
3. Если ошибку можно **исправить на лету**, как в нашем случае, то проще всего будет сделать это тут же. Для этого переключимся в запрос **Продажи** через левую панель в редакторе, выделим столбец **Товар** и исправим в нем регистр с помощью команды **Формат → Каждое Слово С Прописной** на вкладке **Преобразование (Transform → Format → Capitalize Each Word)**:

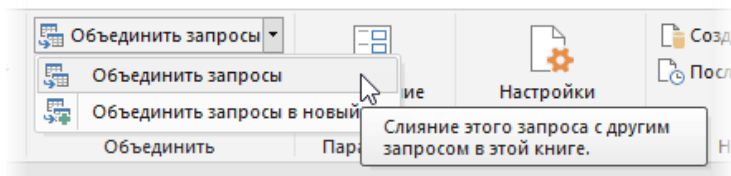


Вернувшись после этого в запрос **Merge1**, мы увидим, что ошибка исчезла.

Объединение в этом же запросе

Чтобы объединить две таблицы, не обязательно создавать отдельный третий запрос: во многих случаях бывает удобнее выполнить это здесь же, на месте, в текущем запросе, с которым мы имеем дело.

Давайте подобным образом добавим к нашей сборке данные из третьей таблицы **География**. Для этого, находясь в запросе **Merge1**, выберем на вкладке **Главная** команду **Объединить запросы → Объединить запросы (Home → Merge Queries → Merge Queries)**:



Мы увидим уже знакомое окно, где нужно будет выбрать запрос **География** в качестве второй таблицы для объединения и выделить общие столбцы (**Продавец** и **Менеджер**) в этих таблицах, как мы делали ранее:

¹ Подробнее про **null** см. главу **Тип null** ближе к концу этой книги.

Слияние

Выберите таблицу и совпадающие столбцы для создания объединенной таблицы.

Merge1

Дата	Товар	Количество	Продавец	Цена	Скидка
17.10.2017 0:00:00	Джемпер	1	Дмитрий	9055	0
23.10.2017 0:00:00	Джемпер	2	Дмитрий	9055	0
02.06.2017 0:00:00	Брюки	3	Дмитрий	4625	0,1
15.11.2017 0:00:00	Кардиган	4	Елена	5810	0
24.07.2017 0:00:00	Кардиган	1	Максим	5810	0

География

Город	Менеджер	Статус	Регион
Новосибирск	Дмитрий	менеджер	Сибирь
Екатеринбург	Елена	менеджер	Сибирь
Санкт-Петербург	Андрей	менеджер	Запад
Москва	Анастасия	стажер	Центр
Москва	Анна	менеджер	Центр

Тип соединения
Внешнее соединение слева (все из первой таблиц...

✓ Выбор согласовал 200 из первых строк (200).

OK Отмена

После нажатия на **OK** и разворачивания вложенных таблиц, как в предыдущем случае, мы увидим подставленные данные из третьей таблицы – столбцы **Город**, **Статус** и **Регион**:

Дата	Товар	Количество	Продавец	Цена	Скидка	Город	Статус	Регион
17.10.2017 0:00:00	Джемпер	1	Дмитрий	9055	0	Новосибирск	менеджер	Сибирь
23.10.2017 0:00:00	Джемпер	2	Дмитрий	9055	0	Новосибирск	менеджер	Сибирь
02.06.2017 0:00:00	Брюки	3	Дмитрий	4625	0,1	Новосибирск	менеджер	Сибирь
22.04.2017 0:00:00	Кофта	4	Дмитрий	735	0	Новосибирск	менеджер	Сибирь
15.11.2017 0:00:00	Кардиган	4	Елена	5810	0	Екатеринбург	менеджер	Сибирь
27.11.2017 0:00:00	Свитер	2	Елена	7925	0	Екатеринбург	менеджер	Сибирь
01.10.2017 0:00:00	Свитер	3	Андрей	7925	0	Санкт-Петербург	менеджер	Запад
09.05.2017 0:00:00	Кофта	1	Андрей	735	0	Санкт-Петербург	менеджер	Запад
25.01.2017 0:00:00	Джинсы	3	Анастасия	6435	0	Москва	стажер	Центр
24.07.2017 0:00:00	Кардиган	1	Максим	5810	0	Новосибирск	стажер	Сибирь

Теперь можно добавить к нашей таблице вычисляемый столбец для выручки. Для этого на вкладке **Добавление столбца** (Add Column) нажмем на кнопку **Настраиваемый столбец** (Custom Column) и введем в открывшееся окно имя нового столбца и формулу для его расчёта:

Настраиваемый столбец

Имя нового столбца
Выручка

Пользовательская формула столбца:
=[Количество]*[Цена]*(1-[Скидка])

Доступные столбцы:
Дата
Товар
Количество
Продавец
Цена
Скидка
Город

<< Вставить

Сведения о формулах Power Query

✓ Синтаксические ошибки не обнаружены.

OK Отмена

Что интересно, после нажатия на **ОК** и создания вычисляемого столбца для выручки можно совершенно безнаказанно удалить колонки, которые участвуют в её вычислении, например **Количество**, **Цену** и **Скидку**. Это никак не повлияет на столбец **Выручка**:

	Дата	Товар	Продавец	Город	Статус	Регион	Выручка
1	17.10.2017 0:00:00	Джемпер	Дмитрий	Новосибирск	менеджер	Сибирь	9055
2	23.10.2017 0:00:00	Джемпер	Дмитрий	Новосибирск	менеджер	Сибирь	18110
3	02.06.2017 0:00:00	Брюки	Дмитрий	Новосибирск	менеджер	Сибирь	12487,5
4	22.04.2017 0:00:00	Кофта	Дмитрий	Новосибирск	менеджер	Сибирь	2940
5	15.11.2017 0:00:00	Кардиган	Елена	Екатеринбург	менеджер	Сибирь	23240
6	27.11.2017 0:00:00	Свитер	Елена	Екатеринбург	менеджер	Сибирь	15850
7	01.10.2017 0:00:00	Свитер	Андрей	Санкт-Петербург	менеджер	Запад	23775
8	09.05.2017 0:00:00	Кофта	Андрей	Санкт-Петербург	менеджер	Запад	735
9	25.01.2017 0:00:00	Джинсы	Анастасия	Москва	стажер	Центр	19305

Для большинства пользователей, привыкших к формулам на листе Excel, этот момент будет слегка необычным. Однако, если подумать, это даёт нам возможность избавляться в запросах от ненужных более столбцов в любой момент, не задумываясь о том, в каких вычислениях они раньше участвовали. А чем раньше вы избавитесь от лишних данных, тем быстрее будет работать ваш запрос.

Объединение по нескольким столбцам

В реальной практике весьма часто встречается случай, когда поиск и подстановка данных из одной таблицы в другую должны происходить по совпадению не одного, а сразу нескольких параметров в нескольких столбцах. Например, нам может потребоваться подставить цены из Прайс-листа в таблицу Заказы по совпадению связки из трех параметров – модели телефона, его цвета и объема памяти:

	A	B	C	D	E	F	G	H	I	J	K
1											
2		Заказы					Прайс-лист				
3											
4		Модель	Цвет	Память			Модель	Память	Цвет	Цена	
5		iPhone 5	серебро	64			iPhone 4	32	черный	99	
6		iPhone 5	черный	64			iPhone 4	32	белый	99	
7		iPhone 4	белый	64			iPhone 4	64	черный	149	
8		iPhone 5	черный	64			iPhone 4	64	белый	149	
9		iPhone 6	розовый	128			iPhone 5	32	розовый	199	
10		iPhone 6	розовый	64			iPhone 5	32	серебро	199	
11		iPhone 5	золотой	64			iPhone 5	32	черный	199	
12		iPhone 4	черный	32			iPhone 5	32	золотой	199	
13		iPhone 6	черный	32			iPhone 5	32	красный	199	
14		iPhone 6	серебро	64			iPhone 5	64	розовый	299	
15		iPhone 6	розовый	128			iPhone 5	64	серебро	299	
16		iPhone 6	красный	128			iPhone 5	64	черный	299	
17		iPhone 5	золотой	64			iPhone 5	64	золотой	299	
18							iPhone 5	64	красный	299	
19							iPhone 6	32	розовый	199	
20							iPhone 6	32	серебро	199	
21							iPhone 6	32	черный	199	
22							iPhone 6	32	золотой	199	
23							iPhone 6	32	красный	199	

Классический подход при использовании функций а-ля **ВПР (VLOOKUP)** предполагает обычно склеивание нескольких столбцов в один, чтобы получить на выходе дополнительный столбец с уникальным ключом-сцепкой, или использование функции **СУММЕСЛИМН¹**.

Если же вы используете Power Query, то всё гораздо проще.

1. Превратим обе наши таблицы в «умные» и загрузим в Power Query в режиме «только подключение», как мы делали ранее.
2. Создадим еще один запрос для объединения исходных таблиц через **Данные → Получить данные → Объединение запросов → Объединить (Data → Get Data → Combine Queries → Merge)**.
3. В открывшемся окне выберем исходную таблицу (**Заказы**) и таблицу, откуда хотим подставить данные (**Прайс**) из выпадающих списков, – опять же точно так, как уже делали в предыдущем примере.
4. Выделим в первой таблице, **удерживая клавишу Ctrl**, те столбцы, которые хотим учитывать при подстановке в любой последовательности, например, так: **Модель-Цвет-Память**. Обратите внимание, что рядом с именами столбцов появятся их порядковые номера при выделении.

Слияние

Выберите таблицы и совпадающие столбцы для создания объединенной таблицы

Заказы

Модель	1	Цвет	2	Память	3
iPhone 5		серебро		64	
iPhone 5		черный		64	
iPhone 4		белый		64	
iPhone 5		черный		64	
iPhone 6		розовый		128	

Прайс

Модель	1	Память	3	Цвет	2	Цена
iPhone 4		32		черный		99
iPhone 4		32		белый		99
iPhone 4		64		черный		149
iPhone 4		64		белый		149
iPhone 5		32		розовый		199

Тип соединения

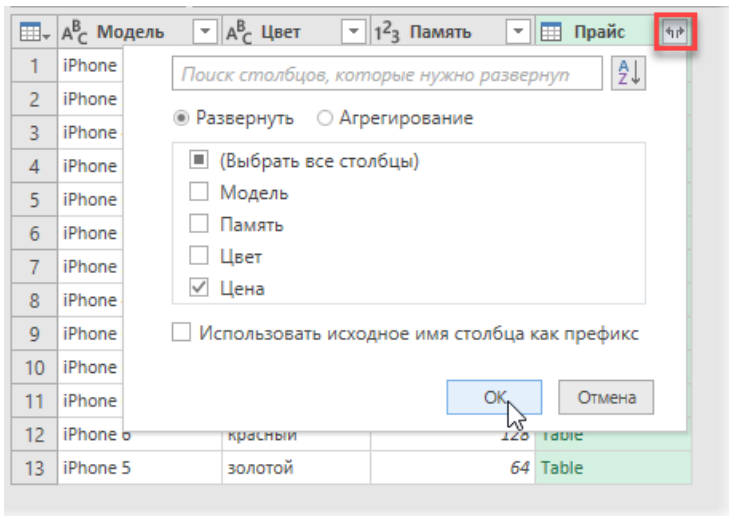
Внешнее соединение слева (все из первой таблиц...

✓ Выбор согласовал 13 из первых строк (13).

¹ Подробнее об этом можно почитать в моей книге «Мастер Формул» или на сайте на странице <https://www.planetaexcel.ru/techniques/2/224/>.

5. Затем аналогичным образом, **удерживая Ctrl**, выделим те же столбцы во второй таблице, соблюдая **исходную последовательность Модель-Цвет-Память**.
6. Всё. Можно жать на **ОК**.

Дальше останется развернуть кнопкой с двойными стрелками содержимое вложенных таблиц в столбце **Прайс** и выбрать в раскрывающемся списке те колонки, которые мы хотим подставить (**Цена**):



Вот, собственно, и все хитрости. Красиво, не правда ли?

Подстановка сразу всех найденных значений

Если в исходных данных существует больше одного совпадения с искомым значением, то функция ВПР, как известно, выдает только первое. А что, если нам нужно найти их все?

Предположим, что у нас есть большая таблица заказов (с повторами и неотсортированная, разумеется) и рядом маленькая таблица с названиями интересующих нас товаров, по каждому из которых мы хотим вывести все номера заказов (например, через запятую):

	A	B	C	D	E	F	G	H
1	Номер заказа	Дата заказа	Товар	Стоимость				
2	30	07.04.2017	Свекла	26 329,47				
3	221	10.06.2017	Малина	86 451,50			Товар	Все номера заказов
4	127	06.08.2017	Земляника	33 870,03			Земляника	127, 146, 672 и т.д.
5	475	08.07.2017	Авокадо	8 965,43			Малина	221, 37, 125 и т.д.
6	146	15.04.2017	Земляника	8 260,50				
7	536	27.04.2017	Тыква	82 477,97				
8	37	18.05.2017	Малина	39 971,96				
9	793	06.03.2017	Спаржа	44 636,39				
10	120	21.07.2017	Крапива	57 154,40				
11	425	13.01.2017	Апельсин	53 152,22				
12	95	12.12.2017	Нут	6 618,23				
13	125	23.07.2017	Малина	37 674,91				
14	672	12.05.2017	Земляника	59 560,09				
15	638	29.12.2017	Спаржа	66 390,70				

Эту задачу с помощью ВПР и ей подобных функций уже не решить, придется привлекать тяжелую артиллерию в виде дополнительных вычисляемых столбцов, формул массива или даже макросов.

Power Query же с подобным справится относительно легко.

1. Сначала, естественно, придется загрузить исходные данные в Power Query. Для этого, как и ранее, превратим обе наши таблицы в «умные», дадим им имена (я назвал их *Заказы* и *Товары* соответственно) и зальем их в редактор запросов как подключения (**Only Create Connection**).
2. Создадим третий запрос на объединение двух предыдущих через **Данные → Получить данные → Объединение запросов → Объединить** (Data → Get Data → Combine Queries → Merge).
3. В открывшемся окне в качестве первой таблицы (к которой подтягиваем данные) выберем маленькую **Товары**, а в качестве второй (из которой берем данные) – большую таблицу **Заказы**, выделим в обеих столбцы *Товар* и нажмем на **ОК**. Тип соединения оставим стандартный – **Внешнее соединение слева** (Left Outer Join):

Слияние

Выберите таблицы и совпадающие столбцы для создания объединенной таблицы

Товары

Товар
Земляника
Малина

Заказы

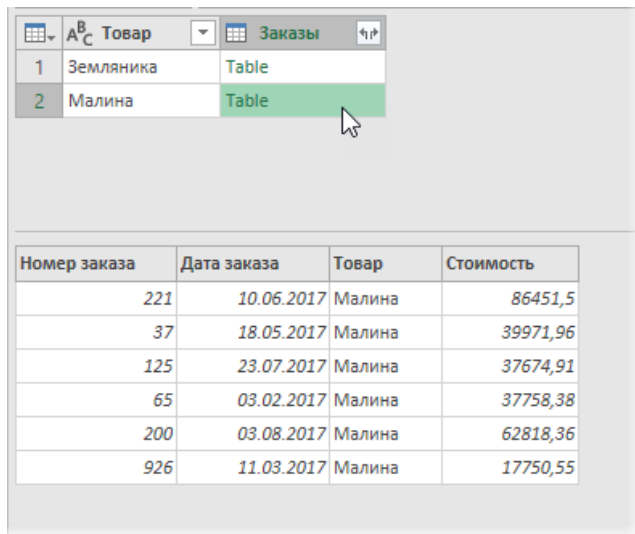
Номер заказа	Дата заказа	Товар	Стоимость
30	07.04.2017	Свекла	26329,47
221	10.06.2017	Малина	86451,5
127	06.08.2017	Земляника	33870,03
475	08.07.2017	Авокадо	8965,43
146	15.04.2017	Земляника	8260,5

Тип соединения

Внешнее соединение слева (все из первой таблиц...

✓ Выбор согласовал 2 из первых строк (2).

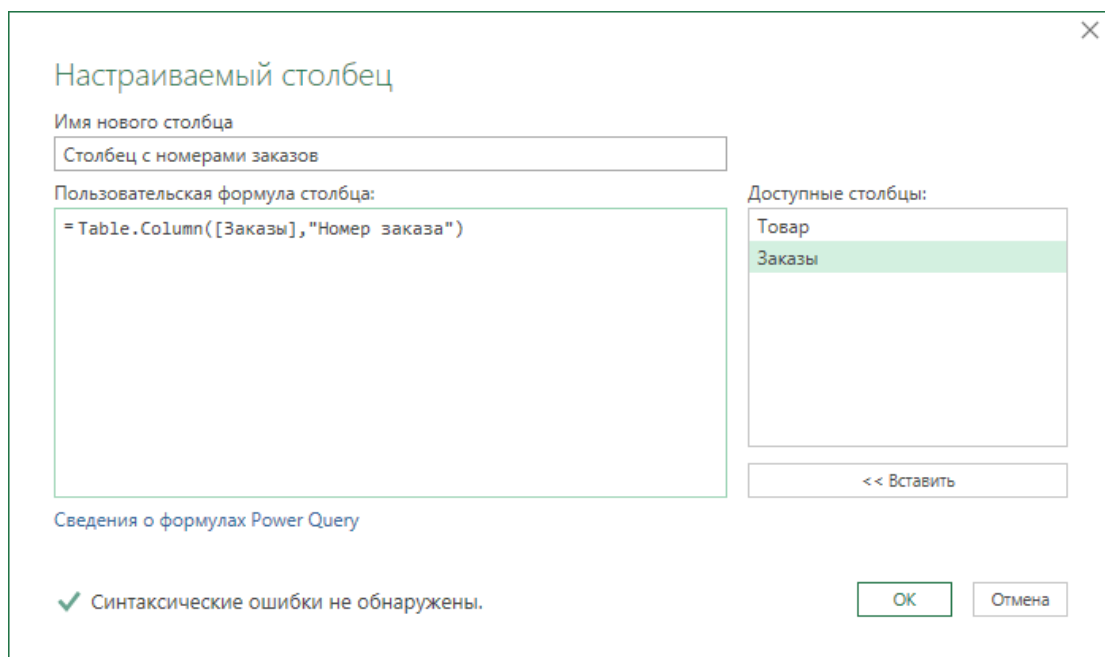
4. Далее на экране должна возникнуть уже знакомая картина: к нашей таблице добавится новый столбец **Заказы** с вложенными в ячейки таблицами-фрагментами из заказов по каждому товару. Если щелкнуть мышью в белый фон (но не в зеленое слово **Table**), то в нижней части окна можно увидеть содержимое каждой такой таблицы:



Номер заказа	Дата заказа	Товар	Стоимость
221	10.06.2017	Малина	86451,5
37	18.05.2017	Малина	39971,96
125	23.07.2017	Малина	37674,91
65	03.02.2017	Малина	37758,38
200	03.08.2017	Малина	62818,36
926	11.03.2017	Малина	17750,55

5. Теперь самое главное. Разворачивать таблицы целиком с помощью кнопки с двойными стрелками мы на этот раз не будем. Поскольку нам требуется только один столбец, то мы используем язык M и специальную функцию для извлечения колонки **Номер заказа** из каждой таблицы.

Для этого на вкладке **Добавление столбца** выберем команду **Настраиваемый столбец** (Add Column → Custom Column) и введем туда имя нового столбца (любое) и следующую формулу:



Настраиваемый столбец

Имя нового столбца
Столбец с номерами заказов

Пользовательская формула столбца:
= Table.Column([Заказы], "Номер заказа")

Доступные столбцы:
Товар
Заказы

<< Вставить

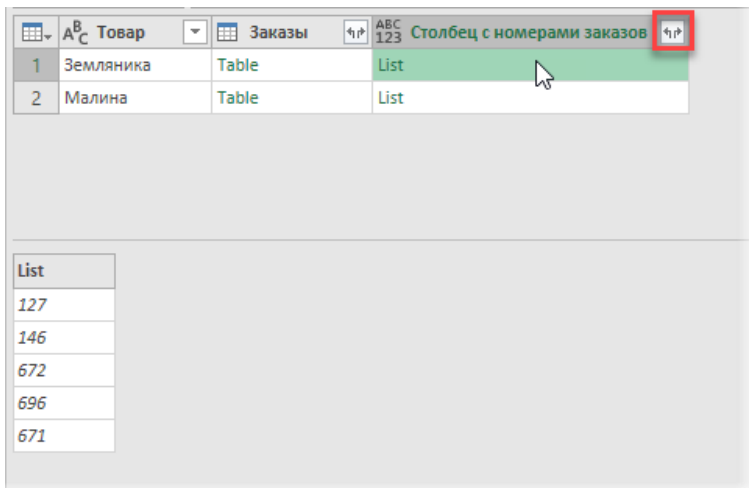
Сведения о формулах Power Query

✓ Синтаксические ошибки не обнаружены.

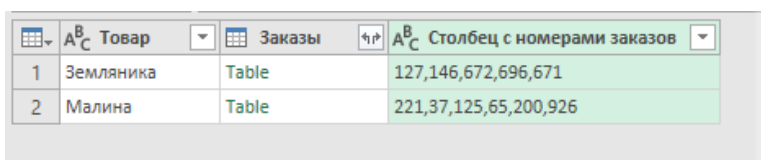
OK Отмена

Функция **Table.Column** имеет два аргумента:

- **Таблица**, откуда мы извлекаем данные (эти таблицы вложены у нас в ячейки столбца **Заказы**);
 - **Имя столбца** (в кавычках), который нам нужен.
6. После нажатия на **OK** мы увидим, как к нашей таблице добавится ещё один столбец, в каждой ячейке которого будет уже не таблица (**Table**), а другой объект – *список (List)*. Технически список – это одномерная таблица-столбец, с которой тоже можно производить различные манипуляции в Power Query. В частности, столбец-список в Power Query можно развернуть в строку, используя заданный символ-разделитель.



7. Развернуть списки можно знакомой кнопкой с двойными стрелками в шапке таблицы, выбрав затем вариант **Извлечь значения** (Extract values). В следующем окне задаем символ-разделитель и после нажатия на **ОК** видим желаемый результат – все номера заказов по каждому товару, разделенные запятыми:



Теперь можно удалить все столбцы, кроме последнего, и выгрузить получившуюся мини-табличку на лист рядом с названиями интересующих нас товаров, используя кнопку **Заккрыть и загрузить → Закрыть и загрузить в...** (Home → Close & Load → Close & Load to...):

	A	B	C	D	E	F	G	H	I	J
1	Номер заказа	Дата заказа	Товар	Стоимость						
2	30	07.04.2017	Свекла	26 329,47						
3	221	10.06.2017	Малина	86 451,50			Товар			
4	127	06.08.2017	Земляника	33 870,03			Земляника	Номера заказов	127,146,672,696,671	
5	475	08.07.2017	Авокадо	8 965,43			Малина		221,37,125,65,200,926	
6	146	15.04.2017	Земляника	8 260,50						
7	536	27.04.2017	Тыква	82 477,97						
8	37	18.05.2017	Малина	39 971,96						
9	793	06.03.2017	Спаржа	44 636,39						
10	120	21.07.2017	Крапива	57 154,40						

Что особенно приятно, так это то, что теперь можно смело добавлять к таблице новые интересующие нас товары и потом просто обновлять запрос кнопкой **Обновить всё** на вкладке **Данные** (Data → Refresh All) или сочетанием клавиш **Ctrl+Alt+F5** – и мы тут же увидим номера всех заказов, которые по ним были сделаны:

	A	B	C	D	E	F	G	H	I	J
1	Номер заказа	Дата заказа	Товар	Стоимость						
2	30	07.04.2017	Свекла	26 329,47						
3	221	10.06.2017	Малина	86 451,50			Товар			
4	127	06.08.2017	Земляника	33 870,03			Земляника	Номера заказов	127,146,672,696,671	
5	475	08.07.2017	Авокадо	8 965,43			Малина		221,37,125,65,200,926	
6	146	15.04.2017	Земляника	8 260,50			Лосось		711,353,770,74	
7	536	27.04.2017	Тыква	82 477,97			Апельсин		425,257,275,908,337	
8	37	18.05.2017	Малина	39 971,96			Авокадо		475,236,639	
9	793	06.03.2017	Спаржа	44 636,39						
10	120	21.07.2017	Крапива	57 154,40						

Интервальный ВПР

*Многие вещи кажутся невыполнимыми до тех пор, пока их не сделаешь.
(Нельсон Мандела)*

Если вы знакомы с функцией ВПР (VLOOKUP), то должны помнить, что у неё есть четвёртый аргумент, называемый **Интервальным просмотром (Range lookup)**, который может принимать значение **ИСТИНА (TRUE)** или **ЛОЖЬ (FALSE)** или, что то же самое, 1 или 0. Этот параметр задаёт режим точного или приблизительного поиска и используется в основном при поиске чисел. Классический пример такой задачи — это поиск скидки, соответствующей покупаемому количеству товара:

ПРОДАЖИ			СКИДКИ		
Товар	Количество	Скидка	От	Скидка	
Арбуз	19	ИСТИНА)	0	0	
Рис	86	15%	10	0,05	
Огурец	52	15%	20	0,1	
Кукуруза	120	20%	50	0,15	
Сельдь	16	5%	100	0,2	
Абрикос	2	0%			
Манго	44	10%			

Особенность функции **ВПР (VLOOKUP)** в таком режиме заключается в том, что она ищет именно ближайшее наименьшее количество в правой таблице и выдаёт соответствующую ему скидку, что нам и требуется. Такой подход позволяет избавиться от большого количества вложенных друг в друга функций **ЕСЛИ (IF)**, которыми обычно решают подобные задачи проверки попадания в один из заданных интервалов. Просто и изящно.

Вопрос в том, как реализовать подобное в Power Query,

Можно, конечно, недолго думая, симитировать вложенные друг в друга функции ЕСЛИ, используя **Условный столбец** с вкладки **Добавление столбца (Add Column → Conditional Column)** и введя затем проверку на попадание в каждый интервал отдельным условием:

Добавление условного столбца

Добавьте условный столбец, который вычисляется из других столбцов или значений.

Имя нового столбца

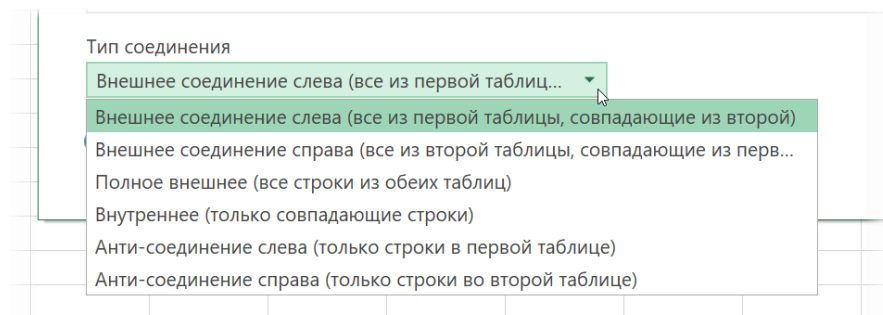
	Имя столбца	Оператор	Значение	То	Вывод
Если	Количество	больше или равно	100	To	20%
Инач...	Количество	больше или равно	50	To	15%
Инач...	Количество	больше или равно	20	To	10%
Инач...	Количество	больше или равно	10	To	5%

В противном случае

При этом надо понимать, что такой способ далёк от идеального, т. к.:

- надо вводить все условия вручную, нажимая кнопку **Добавить правило (Add rule)**;
- условия нужно вводить строго в правильном порядке – от наибольшего к наименьшему, иначе логика нарушается, т. к. срабатывает первое выполнившееся правило;
- чтобы подправить логику скидок, недостаточно будет изменить таблицу со скидками на листе, придётся лезть в редактор Power Query.

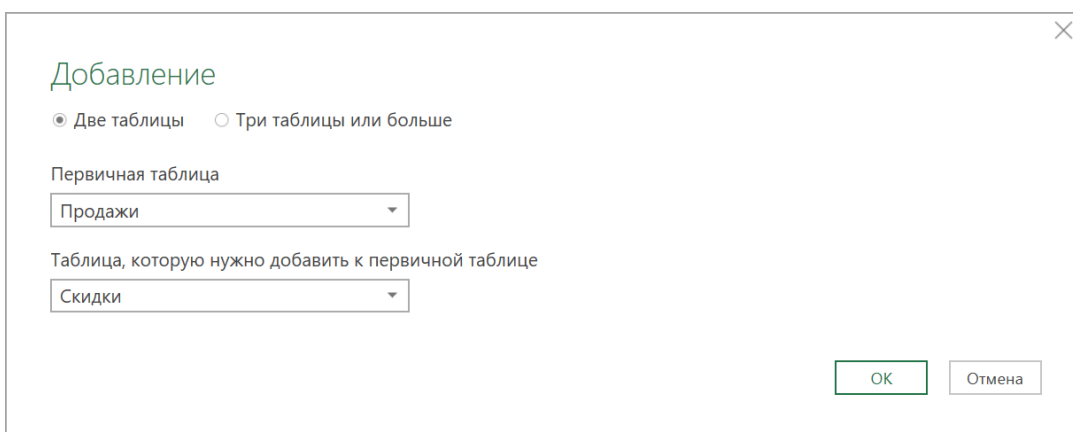
Так что лучше бы реализовать это не условным столбцом, а именно слиянием таблиц, но тут возникает проблема, т. к. в списке вариантов объединения на самом деле нет ничего подходящего для такого случая:



Несмотря на это, решить такую задачу вполне можно, но нам нужен будет другой тип слияния, который мы уже разбирали, – **Добавление (Append)**, а не **Объединение (Merge)**.

Итак, делаем следующее.

1. Превращаем наши исходные таблицы в «умные» и грузим их в Power Query стандартным образом – кнопкой **Из таблицы/диапазона** на вкладке **Данные (Data → From Table/Range)**. Первый запрос назовём, допустим, **Продажи**, а второй, соответственно, **Скидки**.
2. Настроим форматы данных для столбцов в таблицах, если это необходимо.
3. В таблице скидок переименуем столбец **От** в **Количество**, т. е. назовём его так же, как называется столбец с количеством закупаемого товара в таблице продаж.
4. Оба запроса оставим как подключения, воспользовавшись кнопкой **Главная → Закрывать и загружать → Закрывать и загружать в... → Только создать подключение (Home → Close&Load → Close&Load to... → Create only connection)**.
5. Теперь самое интересное. Создадим новый запрос, соединяющий эти две таблицы, – через **Данные → Получить данные → Объединить запросы → Добавить (Data → Get Data → Combine queries → Append)** и выберем наши таблицы в открывшемся окне:



После нажатия на **ОК** обе наши таблицы должны состыковаться по совпадающим столбцам и встать друг под друга:

	АВС Товар	1.2.3 Количество	1.2 Скидка
1	Арбуз	19	null
2	Рис	86	null
3	Огурец	52	null
4	Кукуруза	120	null
5	Сельдь	16	null
6	Абрикос	2	null
7	Манго	44	null
8	null	0	0
9	null	10	0,05
10	null	20	0,1
11	null	50	0,15
12	null	100	0,2

6. Теперь добавим вспомогательный столбец с номерами строк, используя команду **Добавление столбца → Столбец индекса → От 1** (Add Column → Index Column → From 1). Он нужен, чтобы потом, после всех преобразований, восстановить исходную последовательность товаров (Арбуз, Рис, Огурец и т. д.). Если нарушение последовательности вас не волнует, то этот шаг можно пропустить:

	АВС Товар	1.2.3 Количество	1.2 Скидка	1.2 Индекс
1	Арбуз	19	null	1
2	Рис	86	null	2
3	Огурец	52	null	3
4	Кукуруза	120	null	4
5	Сельдь	16	null	5
6	Абрикос	2	null	6
7	Манго	44	null	7
8	null	0	0	8
9	null	10	0,05	9
10	null	20	0,1	10
11	null	50	0,15	11
12	null	100	0,2	12

7. Теперь отсортируем нашу таблицу по столбцу **Количество** по возрастанию, используя кнопку фильтра в соответствующем заголовке сверху:

	АВС Товар	1.2.3 Количество	1.2 Скидка	1.2 Индекс
1	null	0	0	8
2	Абрикос	2	null	6
3	null	10	0,05	9
4	Сельдь	16	null	5
5	Арбуз	19	null	1
6	null	20	0,1	10
7	Манго	44	null	7
8	null	50	0,15	11
9	Огурец	52	null	3
10	Рис	86	null	2
11	null	100	0,2	12
12	Кукуруза	120	null	4

8. Теперь выделим столбец **Скидка** и заполним пустые (null) ячейки значениями из предыдущих (вышестоящих) ячеек командой **Преобразование → Заполнить → Заполнить вниз** (Transform → Fill → Fill Down):

	А ^В С Товар	1 ² 3 Количество	1.2 Скидка	1.2 Индекс
1	<i>null</i>	0	0	8
2	Абрикос	2	0	6
3	<i>null</i>	10	0,05	9
4	Сельдь	16	0,05	5
5	Арбуз	19	0,05	1
6	<i>null</i>	20	0,1	10
7	Манго	44	0,1	7
8	<i>null</i>	50	0,15	11
9	Огурец	52	0,15	3
10	Рис	86	0,15	2
11	<i>null</i>	100	0,2	12
12	Кукуруза	120	0,2	4

9. Удалим строки с *null* в колонке **Товар**, используя фильтр:

	А ^В С Товар	1 ² 3 Количество	1.2 Скидка	1.2 Индекс
1	Абрикос	2	0	6
2	Сельдь	16	0,05	5
3	Арбуз	19	0,05	1
4	Манго	44	0,1	7
5	Огурец	52	0,15	3
6	Рис	86	0,15	2
7	Кукуруза	120	0,2	4

10. И, наконец, восстановим исходную последовательность строк, отсортировав таблицу по столбцу **Индекс** и затем удалив его за ненадобностью:

	А ^В С Товар	1 ² 3 Количество	1.2 Скидка
1	Арбуз	19	0,05
2	Рис	86	0,15
3	Огурец	52	0,15
4	Кукуруза	120	0,2
5	Сельдь	16	0,05
6	Абрикос	2	0
7	Манго	44	0,1

Задача решена!

Сравнение таблиц объединением разных типов

Задача сравнения двух и более таблиц является, несомненно, одной из самых распространенных в повседневной работе почти любого пользователя Microsoft Excel. Решать её можно по-разному:

- использовать функцию **ВПР (VLOOKUP)** для поиска данных из одной таблицы в другой. Там, где ВПР не сможет найти искомое значение, мы получим ошибку #Н/Д, которую потом можно отловить фильтрацией. Вместо **ВПР** подойдут и другие функции с похожей логикой: **ПОИСКПОЗ (MATCH)**, **СЧЁТЕСЛИМН (COUNTIFS)** и т. д.;
- использовать сводные таблицы, предварительно соединив два списка в один и добавив к исходным данным дополнительный столбец с идентификатором списка;
- использовать макросы.

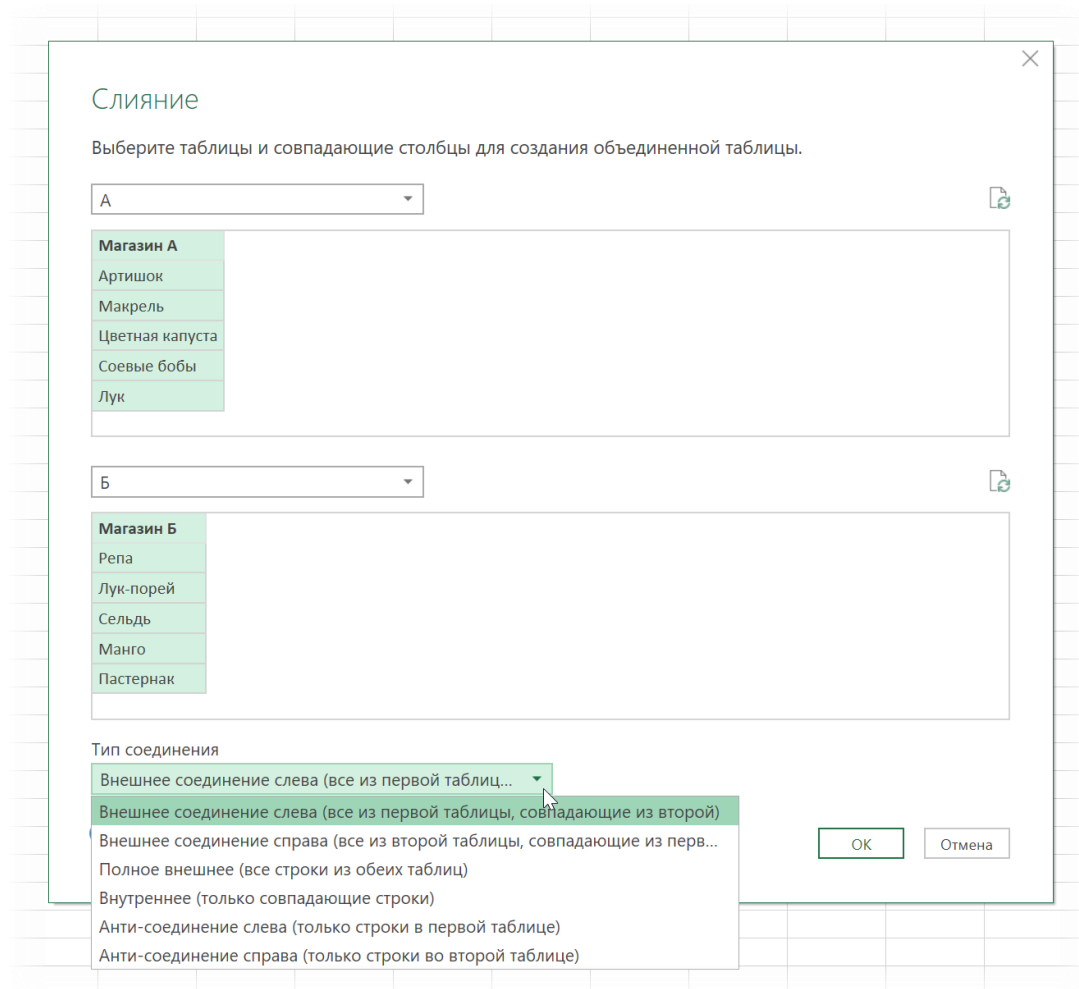
У всех этих подходов есть свои плюсы и минусы, идеальных решений не бывает. Однако после появления Power Query у нас с вами появился ещё один способ, если не идеальный, то очень к нему близкий. В основе его лежит уже знакомое нам объединение запросов с разным типом слияния.

Представим себе два магазина с разным ассортиментом товаров – для наглядности я подсветил в них совпадения с помощью условного форматирования, выделив оба списка и выбрав **Главная → Условное форматирование → Правила выделения ячеек → Повторяющиеся значения (Home → Conditional Formatting → Highlight Cell Rules → Duplicate Values)**:

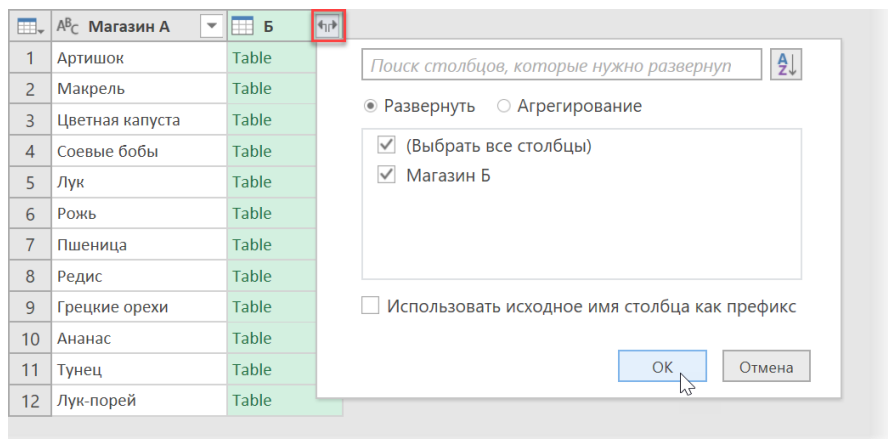
	A	B	C	D	E	F	G
1							
2		Магазин А		Магазин Б			
3		Артишок		Репа			
4		Макрель		Лук-порей			
5		Цветная капуста		Сельдь			
6		Соевые бобы		Манго			
7		Лук		Пастернак			
8		Рожь		Редис			
9		Пшеница		Свекла			
10		Редис		Йогурт			
11		Грецкие орехи		Овес			
12		Ананас		Макрель			
13		Тунец		Помидор			
14		Лук-порей		Рис			
15				Ананас			
16				Ячмень			
17							

Само собой, исходные списки не отсортированы и вдобавок разного размера. Наша задача состоит в том, чтобы быстро определять отличия и совпадения в этих двух списках и выводить в виде отдельных мини-таблиц. Естественно, в будущем ассортимент обоих магазинов может меняться, тогда нам достаточно будет просто обновить наши запросы, чтобы отследить все изменения.

Поэтому превратим наши исходные таблицы в «умные» и загрузим их в Power Query как подключение, как мы уже неоднократно делали ранее. Суть всех дальнейших построений основана на уже знакомой нам процедуре объединения (**Merge**) запросов, но с той разницей, что раньше мы использовали только один тип слияния по умолчанию – **Внешнее соединение слева (Left Outer)**, а сейчас попробуем поиграть с другими вариантами. Так что выберем на вкладке **Данные** команду **Получить данные → Объединить запросы → Объединить (Data → Get Data → Combine Queries → Merge)** и в открывшемся окне выберем наши таблицы и выделим в них столбцы для поиска совпадений:

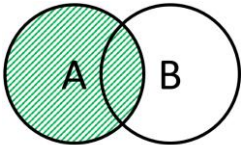
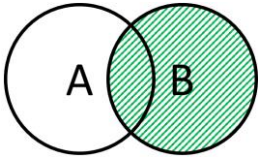
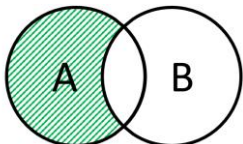


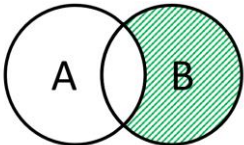
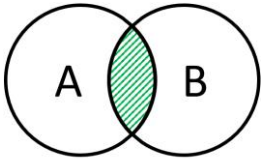
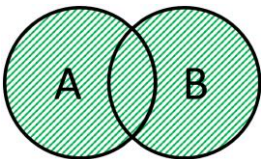
После нажатия на **OK**, вне зависимости от выбранного типа соединения нужно будет развернуть добавившийся столбец с вложенными таблицами с помощью значка с двойными стрелками в шапке таблицы:



В выпадающем списке можно выбрать, какие именно столбцы из второй таблицы мы хотим подставлять (если их не как у нас, один, а несколько). Флажок **Использовать имя столбца как префикс** (Use original column name as prefix) в большинстве случаев можно смело отключить.

Здесь уместно будет сразу же привести диаграммы всех возможных типов объединения таблиц в Power Query и дать по каждому из них небольшой комментарий с описанием возможных результатов:

Тип объединения	Описание																																													
 <p data-bbox="296 409 469 465">Внешнее слева (Left Outer Join)</p>	<p data-bbox="639 237 1474 416">Этот тип слияния имитирует поведение классической функции ВПР (VLOOKUP), т. е. на выходе мы получим все элементы из первой таблицы и в дополнительном столбце найденное совпадение из второй таблицы. Если во второй таблице такого элемента не нашлось, то увидим <i>null</i>:</p> <table border="1" data-bbox="855 439 1257 846"> <thead> <tr> <th></th> <th>АБС Магазин А</th> <th>АБС Магазин Б</th> </tr> </thead> <tbody> <tr><td>1</td><td>Макрель</td><td>Макрель</td></tr> <tr><td>2</td><td>Лук-порей</td><td>Лук-порей</td></tr> <tr><td>3</td><td>Редис</td><td>Редис</td></tr> <tr><td>4</td><td>Ананас</td><td>Ананас</td></tr> <tr><td>5</td><td>Артишок</td><td><i>null</i></td></tr> <tr><td>6</td><td>Цветная капуста</td><td><i>null</i></td></tr> <tr><td>7</td><td>Соевые бобы</td><td><i>null</i></td></tr> <tr><td>8</td><td>Лук</td><td><i>null</i></td></tr> <tr><td>9</td><td>Рожь</td><td><i>null</i></td></tr> <tr><td>10</td><td>Пшеница</td><td><i>null</i></td></tr> <tr><td>11</td><td>Грецкие орехи</td><td><i>null</i></td></tr> <tr><td>12</td><td>Тунец</td><td><i>null</i></td></tr> </tbody> </table>		АБС Магазин А	АБС Магазин Б	1	Макрель	Макрель	2	Лук-порей	Лук-порей	3	Редис	Редис	4	Ананас	Ананас	5	Артишок	<i>null</i>	6	Цветная капуста	<i>null</i>	7	Соевые бобы	<i>null</i>	8	Лук	<i>null</i>	9	Рожь	<i>null</i>	10	Пшеница	<i>null</i>	11	Грецкие орехи	<i>null</i>	12	Тунец	<i>null</i>						
	АБС Магазин А	АБС Магазин Б																																												
1	Макрель	Макрель																																												
2	Лук-порей	Лук-порей																																												
3	Редис	Редис																																												
4	Ананас	Ананас																																												
5	Артишок	<i>null</i>																																												
6	Цветная капуста	<i>null</i>																																												
7	Соевые бобы	<i>null</i>																																												
8	Лук	<i>null</i>																																												
9	Рожь	<i>null</i>																																												
10	Пшеница	<i>null</i>																																												
11	Грецкие орехи	<i>null</i>																																												
12	Тунец	<i>null</i>																																												
 <p data-bbox="292 1077 485 1133">Внешнее справа (Right Outer Join)</p>	<p data-bbox="639 891 1474 999">Обратный вариант по сравнению с предыдущим. В результате мы увидим все товары из второго списка и рядом с ними совпадения из первого (или <i>null</i>, если совпадения нет):</p> <table border="1" data-bbox="863 1021 1249 1473"> <thead> <tr> <th></th> <th>АБС Магазин А</th> <th>АБС Магазин Б</th> </tr> </thead> <tbody> <tr><td>1</td><td><i>null</i></td><td>Репа</td></tr> <tr><td>2</td><td>Макрель</td><td>Макрель</td></tr> <tr><td>3</td><td>Лук-порей</td><td>Лук-порей</td></tr> <tr><td>4</td><td><i>null</i></td><td>Сельдь</td></tr> <tr><td>5</td><td><i>null</i></td><td>Манго</td></tr> <tr><td>6</td><td><i>null</i></td><td>Пастернак</td></tr> <tr><td>7</td><td>Редис</td><td>Редис</td></tr> <tr><td>8</td><td><i>null</i></td><td>Свекла</td></tr> <tr><td>9</td><td><i>null</i></td><td>Йогурт</td></tr> <tr><td>10</td><td><i>null</i></td><td>Овес</td></tr> <tr><td>11</td><td>Ананас</td><td>Ананас</td></tr> <tr><td>12</td><td><i>null</i></td><td>Помидор</td></tr> <tr><td>13</td><td><i>null</i></td><td>Рис</td></tr> <tr><td>14</td><td><i>null</i></td><td>Ячмень</td></tr> </tbody> </table>		АБС Магазин А	АБС Магазин Б	1	<i>null</i>	Репа	2	Макрель	Макрель	3	Лук-порей	Лук-порей	4	<i>null</i>	Сельдь	5	<i>null</i>	Манго	6	<i>null</i>	Пастернак	7	Редис	Редис	8	<i>null</i>	Свекла	9	<i>null</i>	Йогурт	10	<i>null</i>	Овес	11	Ананас	Ананас	12	<i>null</i>	Помидор	13	<i>null</i>	Рис	14	<i>null</i>	Ячмень
	АБС Магазин А	АБС Магазин Б																																												
1	<i>null</i>	Репа																																												
2	Макрель	Макрель																																												
3	Лук-порей	Лук-порей																																												
4	<i>null</i>	Сельдь																																												
5	<i>null</i>	Манго																																												
6	<i>null</i>	Пастернак																																												
7	Редис	Редис																																												
8	<i>null</i>	Свекла																																												
9	<i>null</i>	Йогурт																																												
10	<i>null</i>	Овес																																												
11	Ананас	Ананас																																												
12	<i>null</i>	Помидор																																												
13	<i>null</i>	Рис																																												
14	<i>null</i>	Ячмень																																												
 <p data-bbox="260 1691 521 1747">Анти-соединение слева (Left Anti Join)</p>	<p data-bbox="639 1518 1474 1585">Этот тип выводит в результате только те позиции, которые есть в первом списке, но при этом отсутствуют во втором:</p> <table border="1" data-bbox="871 1597 1241 1854"> <thead> <tr> <th></th> <th>АБС Магазин А</th> <th>АБС Магазин Б</th> </tr> </thead> <tbody> <tr><td>1</td><td>Артишок</td><td><i>null</i></td></tr> <tr><td>2</td><td>Цветная капуста</td><td><i>null</i></td></tr> <tr><td>3</td><td>Соевые бобы</td><td><i>null</i></td></tr> <tr><td>4</td><td>Лук</td><td><i>null</i></td></tr> <tr><td>5</td><td>Рожь</td><td><i>null</i></td></tr> <tr><td>6</td><td>Пшеница</td><td><i>null</i></td></tr> <tr><td>7</td><td>Грецкие орехи</td><td><i>null</i></td></tr> <tr><td>8</td><td>Тунец</td><td><i>null</i></td></tr> </tbody> </table>		АБС Магазин А	АБС Магазин Б	1	Артишок	<i>null</i>	2	Цветная капуста	<i>null</i>	3	Соевые бобы	<i>null</i>	4	Лук	<i>null</i>	5	Рожь	<i>null</i>	6	Пшеница	<i>null</i>	7	Грецкие орехи	<i>null</i>	8	Тунец	<i>null</i>																		
	АБС Магазин А	АБС Магазин Б																																												
1	Артишок	<i>null</i>																																												
2	Цветная капуста	<i>null</i>																																												
3	Соевые бобы	<i>null</i>																																												
4	Лук	<i>null</i>																																												
5	Рожь	<i>null</i>																																												
6	Пшеница	<i>null</i>																																												
7	Грецкие орехи	<i>null</i>																																												
8	Тунец	<i>null</i>																																												

Тип объединения	Описание																																																																					
 <p data-bbox="204 409 481 465">Анти-соединение справа (Right Anti Join)</p>	<p data-bbox="592 232 1430 300">Работает аналогично предыдущему, но выводит, наоборот, все элементы второго списка, за вычетом тех, что совпадают с первым:</p> <table border="1" data-bbox="815 322 1206 663"> <thead> <tr> <th></th> <th>Магазин А</th> <th>Магазин Б</th> </tr> </thead> <tbody> <tr><td>1</td><td><i>null</i></td><td>Репа</td></tr> <tr><td>2</td><td><i>null</i></td><td>Сельдь</td></tr> <tr><td>3</td><td><i>null</i></td><td>Манго</td></tr> <tr><td>4</td><td><i>null</i></td><td>Пастернак</td></tr> <tr><td>5</td><td><i>null</i></td><td>Свекла</td></tr> <tr><td>6</td><td><i>null</i></td><td>Йогурт</td></tr> <tr><td>7</td><td><i>null</i></td><td>Овес</td></tr> <tr><td>8</td><td><i>null</i></td><td>Помидор</td></tr> <tr><td>9</td><td><i>null</i></td><td>Рис</td></tr> <tr><td>10</td><td><i>null</i></td><td>Ячмень</td></tr> </tbody> </table>		Магазин А	Магазин Б	1	<i>null</i>	Репа	2	<i>null</i>	Сельдь	3	<i>null</i>	Манго	4	<i>null</i>	Пастернак	5	<i>null</i>	Свекла	6	<i>null</i>	Йогурт	7	<i>null</i>	Овес	8	<i>null</i>	Помидор	9	<i>null</i>	Рис	10	<i>null</i>	Ячмень																																				
	Магазин А	Магазин Б																																																																				
1	<i>null</i>	Репа																																																																				
2	<i>null</i>	Сельдь																																																																				
3	<i>null</i>	Манго																																																																				
4	<i>null</i>	Пастернак																																																																				
5	<i>null</i>	Свекла																																																																				
6	<i>null</i>	Йогурт																																																																				
7	<i>null</i>	Овес																																																																				
8	<i>null</i>	Помидор																																																																				
9	<i>null</i>	Рис																																																																				
10	<i>null</i>	Ячмень																																																																				
 <p data-bbox="272 902 414 958">Внутреннее (Inner Join)</p>	<p data-bbox="592 705 1430 813">Этот тип слияния реализует пересечение двух множеств, т. е. на выходе мы получим список только тех товаров, которые обязательно присутствуют в обоих списках одновременно:</p> <table border="1" data-bbox="810 831 1211 990"> <thead> <tr> <th></th> <th>Магазин А</th> <th>Магазин Б</th> </tr> </thead> <tbody> <tr><td>1</td><td>Макрель</td><td>Макрель</td></tr> <tr><td>2</td><td>Лук-порей</td><td>Лук-порей</td></tr> <tr><td>3</td><td>Редис</td><td>Редис</td></tr> <tr><td>4</td><td>Ананас</td><td>Ананас</td></tr> </tbody> </table>		Магазин А	Магазин Б	1	Макрель	Макрель	2	Лук-порей	Лук-порей	3	Редис	Редис	4	Ананас	Ананас																																																						
	Магазин А	Магазин Б																																																																				
1	Макрель	Макрель																																																																				
2	Лук-порей	Лук-порей																																																																				
3	Редис	Редис																																																																				
4	Ананас	Ананас																																																																				
 <p data-bbox="240 1234 446 1290">Полное внешнее (Full Outer Join)</p>	<p data-bbox="592 1034 1430 1178">Этот тип слияния выводит общую таблицу, где присутствуют абсолютно все товары из обоих списков. Те, что совпадают, встанут друг напротив друга в одной строке, остальные будут иметь в паре все тот же <i>null</i>:</p> <table border="1" data-bbox="807 1198 1214 1933"> <thead> <tr> <th></th> <th>Магазин А</th> <th>Магазин Б</th> </tr> </thead> <tbody> <tr><td>1</td><td><i>null</i></td><td>Репа</td></tr> <tr><td>2</td><td>Макрель</td><td>Макрель</td></tr> <tr><td>3</td><td>Лук-порей</td><td>Лук-порей</td></tr> <tr><td>4</td><td><i>null</i></td><td>Сельдь</td></tr> <tr><td>5</td><td><i>null</i></td><td>Манго</td></tr> <tr><td>6</td><td><i>null</i></td><td>Пастернак</td></tr> <tr><td>7</td><td>Редис</td><td>Редис</td></tr> <tr><td>8</td><td><i>null</i></td><td>Свекла</td></tr> <tr><td>9</td><td><i>null</i></td><td>Йогурт</td></tr> <tr><td>10</td><td><i>null</i></td><td>Овес</td></tr> <tr><td>11</td><td>Ананас</td><td>Ананас</td></tr> <tr><td>12</td><td><i>null</i></td><td>Помидор</td></tr> <tr><td>13</td><td><i>null</i></td><td>Рис</td></tr> <tr><td>14</td><td><i>null</i></td><td>Ячмень</td></tr> <tr><td>15</td><td>Артишок</td><td><i>null</i></td></tr> <tr><td>16</td><td>Цветная капуста</td><td><i>null</i></td></tr> <tr><td>17</td><td>Соевые бобы</td><td><i>null</i></td></tr> <tr><td>18</td><td>Лук</td><td><i>null</i></td></tr> <tr><td>19</td><td>Рожь</td><td><i>null</i></td></tr> <tr><td>20</td><td>Пшеница</td><td><i>null</i></td></tr> <tr><td>21</td><td>Грецкие орехи</td><td><i>null</i></td></tr> <tr><td>22</td><td>Тунец</td><td><i>null</i></td></tr> </tbody> </table> <p data-bbox="592 1955 1425 2022">Применение такого варианта соединения мы подробно рассмотрим на примере в следующей главе.</p>		Магазин А	Магазин Б	1	<i>null</i>	Репа	2	Макрель	Макрель	3	Лук-порей	Лук-порей	4	<i>null</i>	Сельдь	5	<i>null</i>	Манго	6	<i>null</i>	Пастернак	7	Редис	Редис	8	<i>null</i>	Свекла	9	<i>null</i>	Йогурт	10	<i>null</i>	Овес	11	Ананас	Ананас	12	<i>null</i>	Помидор	13	<i>null</i>	Рис	14	<i>null</i>	Ячмень	15	Артишок	<i>null</i>	16	Цветная капуста	<i>null</i>	17	Соевые бобы	<i>null</i>	18	Лук	<i>null</i>	19	Рожь	<i>null</i>	20	Пшеница	<i>null</i>	21	Грецкие орехи	<i>null</i>	22	Тунец	<i>null</i>
	Магазин А	Магазин Б																																																																				
1	<i>null</i>	Репа																																																																				
2	Макрель	Макрель																																																																				
3	Лук-порей	Лук-порей																																																																				
4	<i>null</i>	Сельдь																																																																				
5	<i>null</i>	Манго																																																																				
6	<i>null</i>	Пастернак																																																																				
7	Редис	Редис																																																																				
8	<i>null</i>	Свекла																																																																				
9	<i>null</i>	Йогурт																																																																				
10	<i>null</i>	Овес																																																																				
11	Ананас	Ананас																																																																				
12	<i>null</i>	Помидор																																																																				
13	<i>null</i>	Рис																																																																				
14	<i>null</i>	Ячмень																																																																				
15	Артишок	<i>null</i>																																																																				
16	Цветная капуста	<i>null</i>																																																																				
17	Соевые бобы	<i>null</i>																																																																				
18	Лук	<i>null</i>																																																																				
19	Рожь	<i>null</i>																																																																				
20	Пшеница	<i>null</i>																																																																				
21	Грецкие орехи	<i>null</i>																																																																				
22	Тунец	<i>null</i>																																																																				

Сравнение таблиц с помощью условного столбца

Это ещё один и при этом весьма изящный способ сравнения двух таблиц. Он основан на полном объединении с последующим применением *условного столбца* для выявления имеющихся отличий.

Предположим, что нам с вами нужно сравнить старую и новую версии прайс-листа:

	A	B	C	D	E	F	G	H	
1	Старый прайс			Новый прайс					
2									
3	Товар	Цена			Продукт	Цена, руб			
4	Инжир	99			Лук	23			
5	Салат зеленый	18			Репа	20			
6	Редис	39			Авокадо	139			
7	Петрушка	24			Цветная капуста	60			
8	Малина	120			Лук-порей	85			
9	Лук	23			Редис	39			
10	Груша	60			Инжир	133		Поменялась цена	
11	Репа	20			Манго	40			
12	Авокадо	139			Грибы	17			
13	Ежевика	130			Киви	210			
14	Манго	40			Салат зеленый	18			
15	Черника	130			Петрушка	24			
16	Цветная капуста	60			Ячмень	14			
17	Дыня	37			Дыня	40			
18	Грибы	17			Чеснок	51			
19					Крапива				
20					Финики	150		Новый товар	
21					Пшеница	29			
22					Груша	60			
23									

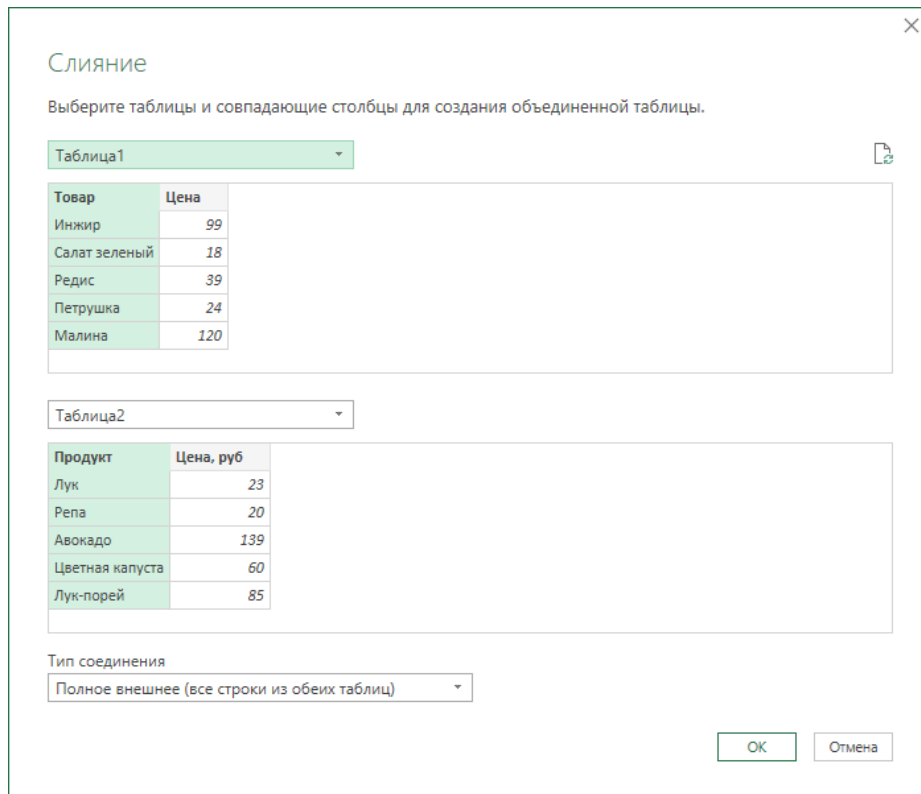
С ходу видно, что в новом прайсе что-то добавилось (*финики, чеснок...*), что-то пропало (*ежевика, малина...*), у каких-то товаров изменилась цена (*инжир, дыня...*). Нужно быстро найти и компактно вывести все эти изменения в одной удобной для просмотра таблице.

Как и ранее, преобразуем сначала наши таблицы в «умные» сочетанием **Ctrl+T** или командой **Главная → Форматировать как таблицу** (Home → Format as Table). Имена созданных таблиц можно подкорректировать на вкладке **Конструктор** (я оставлю стандартные *Таблица1* и *Таблица2*, которые получаются по умолчанию).

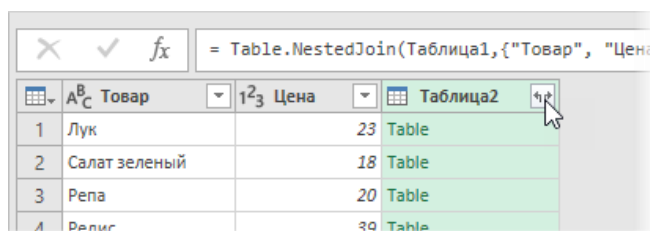
Затем загрузим по очереди оба прайс-листа в Power Query с помощью кнопки **Из таблицы/диапазона** (From Table/Range) с вкладки **Данные** (Data). После загрузки вернемся обратно в Excel из Power Query командой **Закреть и загрузить → Закреть и загрузить в...** (Close & Load → Close & Load To...), выбрав в появившемся окне опцию **Только создать подключение** (Only Create Connection).

Теперь создадим третий запрос, который будет объединять и сравнивать данные из предыдущих двух. Для этого выберем в Excel на вкладке **Данные → Получить данные → Объединить запросы → Объединить** (Data → Get Data → Combine Queries → Merge).

В окне слияния выберем в выпадающих списках наши таблицы, выделим в них столбцы с названиями товаров и в нижней части зададим способ объединения **Полное внешнее** (Full Outer):



После нажатия на **OK** должна появиться таблица из трех столбцов, где в третьем столбце нужно развернуть содержимое вложенных таблиц с помощью двойной стрелки в шапке:



Ну, и можно дополнительно переименовать столбцы для пуцтей наглядности, щелкнув дважды на заголовках:

	АВС Старый товар	123 Старая цена	АВС Новый товар	123 Новая цена
1	Инжир	99	Инжир	133
2	Лук	23	Лук	23
3	Салат зеленый	18	Салат зеленый	18
4	Репа	20	Репа	20
5	Редис	39	Редис	39
6	Авокадо	120	Авокадо	139
7	Петрушка	24	Петрушка	24
8	Цветная капуста	60	Цветная капуста	60
9	null	null	Лук-порей	85
10	Манго	40	Манго	40
11	Грибы	17	Грибы	17
12	null	null	Киви	210
13	null	null	Ячмень	14
14	Дыня	37	Дыня	40
15	null	null	Чеснок	51
16	null	null	Крапива	10
17	null	null	Финики	150
18	null	null	Пшеница	29
19	Груша	60	Груша	60
20	Малина	120	null	null
21	Ежевика	150	null	null
22	Черника	130	null	null

В итоге получим картинку, немного отличающуюся от тех, что мы видели в прошлых примерах, когда делали подстановку данных из одной таблицы в другую методом **Внешнее соединение слева (Left Outer Join)**. Здесь у нас произошло полное слияние данных из обоих прайс-листов.

- Совпадающие по названиям товары из старого и нового прайс-листов встали рядом друг с другом в одну строку.
- Новые товары, имеющиеся только во второй таблице, получили **null** в первых двух столбцах.
- Убранные товары, имеющиеся только в старом прайс-листе, напротив, получили null в последних двух колонках.

А теперь самое интересное. Чтобы отловить все эти отличия, добавим к нашей таблице ещё один столбец, имитирующий кучу вложенных друг в друга функций **ЕСЛИ (IF)**. Для этого идем на вкладку **Добавить столбец (Add Column)** и жмем на кнопку **Условный столбец (Conditional Column)**. А затем в открывшемся окне вводим несколько условий проверки с соответствующими им значениями на выходе:

Добавление условного столбца

Добавьте условный столбец, который вычисляется из других столбцов или значений.

Имя нового столбца
Статус

Имя столбца	Оператор	Значение	То	Выходные данные
Если Старый товар	равно	ABC 123	то	ABC 123 добавлен
Иначе Новый товар	равно	ABC 123	то	ABC 123 удален
Иначе Старая цена	не равно	Новая цена	то	ABC 123 изменилась цена

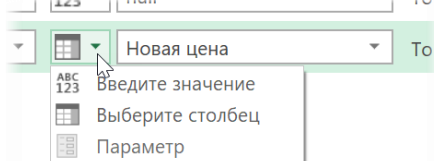
Добавить правило

В противном случае
ABC 123 OK

OK Отмена

Здесь есть пара нюансов.

- При вводе третьего правила нужно будет поменять режим с **Введите значение (Enter value)** на **Выберите столбец (Select column)** с помощью иконки слева от поля ввода:



- При вводе важно соблюдать правильную логическую последовательность правил, т. к. они проверяются именно в том порядке, в котором указаны в этом окне. Так, например, если третье правило «Старая цена <> Новая цена» ввести первым, то оно начнёт срабатывать на всех удаленных и добавленных товарах, т. к. **null** не равен любому числу. Для изменения порядка уже введенных правил можно использовать пиктограмму с многоточием справа от строки.

После нажатия на **OK** мы увидим колонку со статусом каждого товара, которую дополнительно при желании можно отфильтровать, оставив в ней только те строки, где у нас не «OK». Останется выгрузить получившийся отчет в Excel с помощью все той же кнопки **Закрыть и загрузить (Close & Load)** на вкладке **Главная (Home)**:

Старый товар	Старая цена	Новый товар	Новая цена	Статус
Инжир	99	Инжир	133	цена изменилась
Лук	23	Лук	23	ОК
Салат зеленый	18	Салат зеленый	50	цена изменилась
Репа	20	Репа	20	ОК
Редис	39	Редис	39	ОК
Авокадо	120	Авокадо	139	цена изменилась
Петрушка	24	Петрушка	24	ОК
Цветная капуста	60	Цветная капуста	60	ОК
		Лук-порей	85	добавлен
Манго	40	Манго	40	ОК
Грибы	17	Грибы	17	ОК
		Киви	210	добавлен
		Ячмень	14	добавлен
Дыня	37	Дыня	40	цена изменилась
		Чеснок	51	добавлен
		Крапива	10	добавлен
		Финики	150	добавлен
		Пшеница	29	добавлен

Причем если в будущем в прайс-листах произойдут любые изменения (добавятся или удалятся строки, изменятся цены и т. д.), то достаточно будет лишь обновить наши запросы сочетанием клавиш **Ctrl+Alt+F5** или кнопкой **Обновить все (Refresh All)** на вкладке **Данные (Data)**.

Настройка уровней конфиденциальности источников данных

*Тогда лишь двое тайну соблюдают,
Когда один из них её не знает.
(Уильям Шекспир, «Ромео и Джульетта»)*

Если вы начнёте создавать запросы к разным источникам данных с целью объединить потом полученные результаты, то очень быстро столкнётесь с одним не очень понятным с ходу нюансом – настройками уровней конфиденциальности Power Query. Неправильное понимание и применение этих настроек может привести к ощутимому торможению при обновлении некоторых запросов и порой даже к полной или частичной их неработоспособности.

Давайте разберемся в этом вопросе, прежде чем продолжать.

Зачем нужны уровни конфиденциальности

Для начала представьте себе, что при работе в Power Query вы столкнулись с одной из следующих задач:

- Вам нужно загрузить список друзей из Facebook (с личной страницы или страницы вашей компании) и сравнить его со списком клиентов, загруженным из корпоративной CRM-системы.
- Вы хотите взять из ячейки локального Excel-файла номер договора и затем использовать его в запросе к корпоративной базе данных SQL, чтобы отфильтровать всю информацию по этой сделке.
- У вас есть текстовый файл с датами, для каждой из которых нужно сделать веб-запрос курса валют на соответствующий день, обратившись к веб-сервису сайта ЦБ РФ.

Общим моментом для всех перечисленных ситуаций будет то, что информация здесь пересылается между источниками с разным уровнем конфиденциальности, что потенциально может привести к утечке личных или корпоративных данных или попаданию их в чужие руки. Согласитесь, одно дело – ваш личный файл на локальном диске C:\, и совсем другое – корпоративная база данных или публичный веб-сервис, куда вы передаёте свою персональную информацию.

Power Query различает три уровня конфиденциальности источников и по-разному относится к данным, передаваемой им или получаемой от них:

Уровень	Описание	Пример
Частный (Private)	Личная или конфиденциальная информация, доступная очень ограниченному кругу пользователей или только вам.	Данные из вашего аккаунта Facebook, личные дела сотрудников в файле на жёстком диске вашего ПК, содержимое вашего почтового ящика и т. п.
Организационный (Organizational)	Информация внутри организации или компании, доступная только её сотрудникам и авторизованным пользователям.	Файлы на сетевом диске в корпоративной сети, документы с узла Sharepoint с корпоративного портала и т. д.
Общий (Public)	Данные, доступные всем.	Данные с публичных веб-страниц, из открытых баз данных или веб-сервисов.

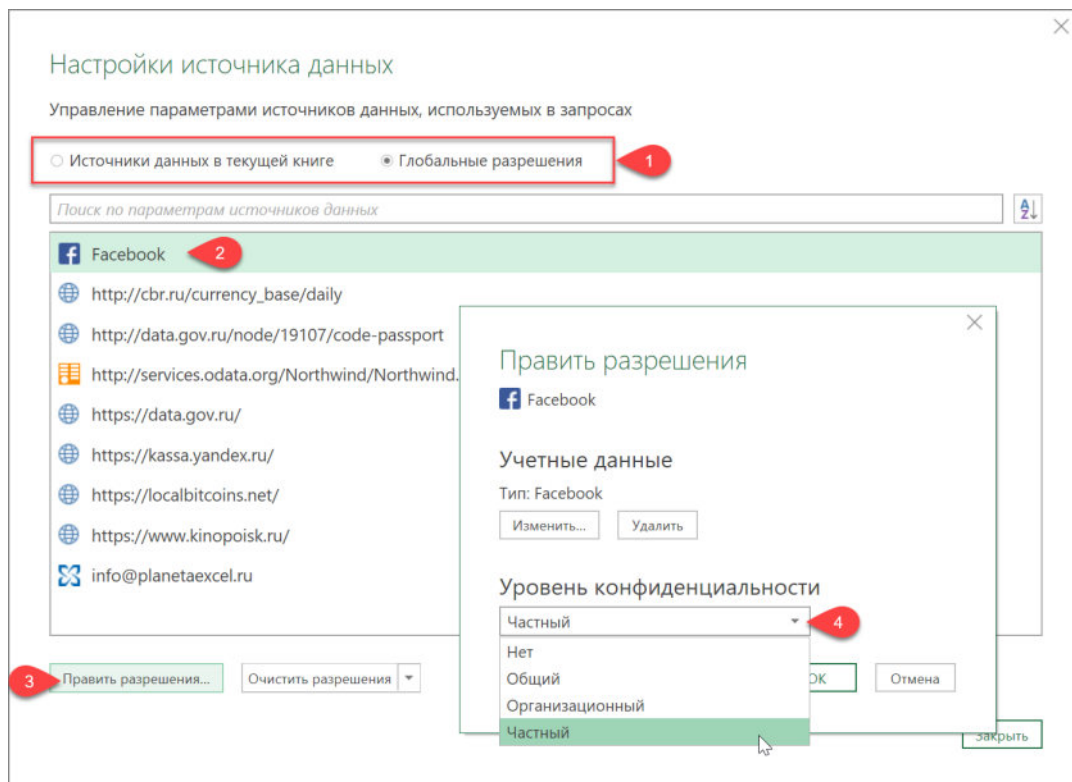
Когда вы соединяете в запросе данные из источников с разным уровнем, надо помнить следующее.

- Источники с **Частным** уровнем изолированы от всех остальных. Данные от частных источников никогда полностью не пересылаются другим получателям (в том числе и частным).
- Информация из источников с **Организационным** уровнем доступна только тем, у кого тоже **Организационный** или **Частный** уровень, и изолирована от источников с **Общим** уровнем.
- Данные из источников с **Общим** уровнем конфиденциальности доступны всем, включая все предыдущие уровни.

Настройка уровней и ошибка Formula.Firewall

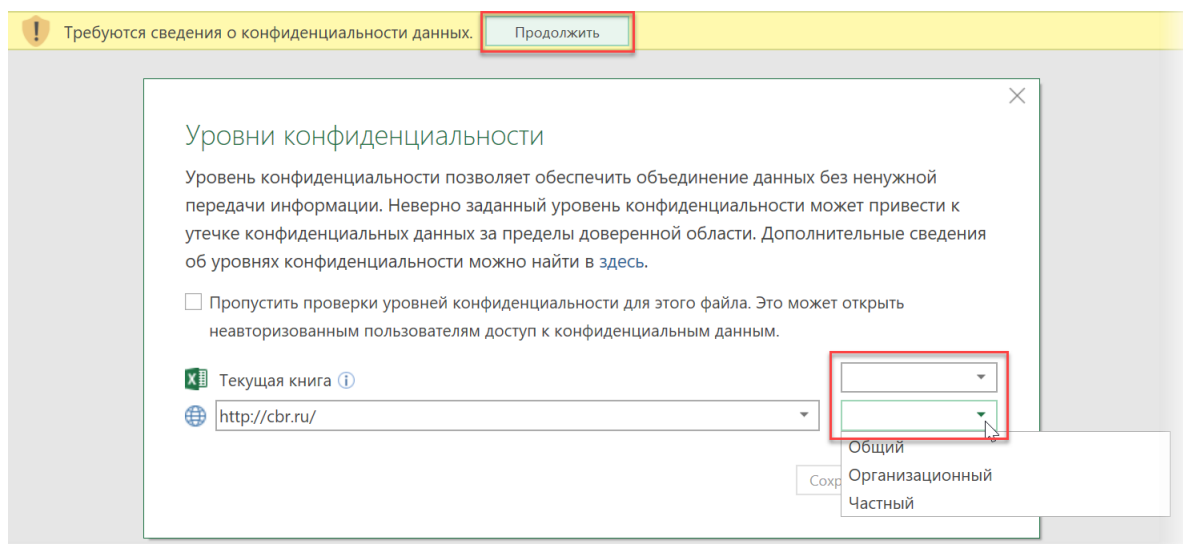
Настройка параметров уровней конфиденциальности в Power Query производится в Excel на вкладке **Данные** → **Получить данные** → **Параметры источника данных** (Data → Get Data → Data Source Settings) или в самом Power Query через меню **Файл** → **Параметры и настройки** → **Настройки источника данных** (File → Options and settings → Options):

В открывшемся окне нужно выбрать, какие источники нас интересуют (локальные или глобальные), выбрать источник (например **Facebook**) и затем, нажав кнопку **Править разрешения** (Edit Permissions), выбрать подходящий уровень конфиденциальности для данного поставщика информации:




Кроме перечисленных трёх уровней (**Частного**, **Организационного** и **Общего**) в выпадающем списке присутствует ещё и вариант **Нет** (None). Он означает, что этот источник наследует уровень от своего «родителя», например, файл будет наследовать уровень конфиденциальности своей папки или диска, где он расположен, а база данных унаследует уровень сервера и т. д.

Если мы смешиваем в запросе данные из разных источников, а уровни для них не были заранее заданы, то запрос не сработает. Мы получим сообщение об ошибке с предложением настроить недостающие уровни в диалоговом окне:

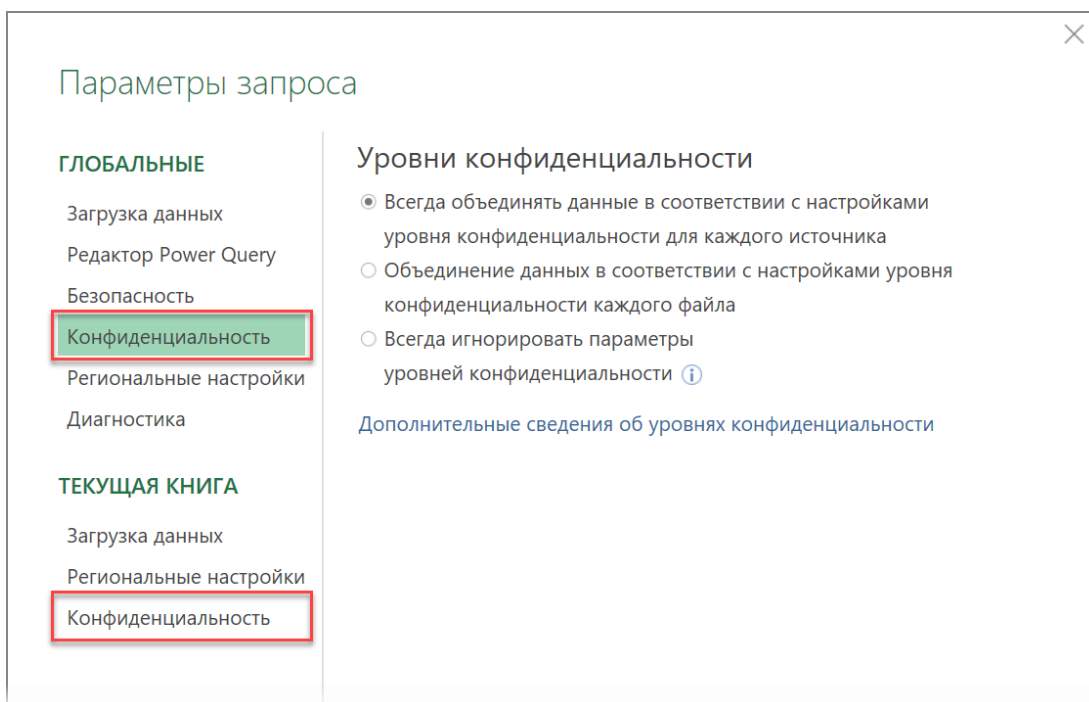


Если комбинация уровней конфиденциальности для источника и получателя данных потенциально может привести к утечке информации (например, мы пытаемся отправить данные из частной книги Excel в публичный веб-сервис), то запрос будет остановлен с ошибкой **Formula.Firewall**:

 Formula.Firewall: Запрос "Запрос курса с сайта ЦБ" (шаг "Измененный тип") запрашивает доступ к источникам данных, имеющим уровни конфиденциальности, которые не могут использоваться вместе. Измените эту комбинацию данных.

Чтобы такой запрос всё же сработал, нам придется вернуться в настройки уровня конфиденциальности для каждого из используемых источников и задать там для обоих **Организационный (Organizational)** или **Общий (Public)** уровень.

При желании мы также можем управлять тем, как Power Query реагирует на смешивание данных из источников с разными уровнями, выбрав в Excel на вкладке **Данные** команду **Получить данные → Параметры запроса (Data → Get Data → Query options)** или в самом Power Query в меню **Файл → Параметры и настройки → Параметры запроса (File → Options and settings → Options)**:



Обратите внимание, что в левой половине окна есть два раздела **Конфиденциальность (Privacy)**: один отвечает за глобальные настройки для всего Power Query, другой – за настройки для текущего файла. Если в глобальных настройках был выбран пункт 1 или 3, то настройки для текущей книги будут недоступны.

Проверять или нет?

В настройке уровней конфиденциальности всегда возникает соблазн выбрать «лёгкий путь» в виде третьей опции в предыдущем окне – **Всегда игнорировать параметры уровней конфиденциальности (Ignore the Privacy Levels)**. Во многих случаях это не лишено смысла, поскольку тогда:

- вам не нужно следить за совместимостью уровней и настраивать их для каждого источника;
- вас реже беспокоит ошибка **Formula.Firewall** (но она может возникать и по другим причинам);
- ваши запросы выполняются быстрее, т. к. проверка конфиденциальности занимает время;
- при запросах к базам данных часть работы запроса (например, первичная фильтрация ненужных данных) перекладывается на сервер, т. е. запросы обновляются заметно шустрее.

Минусом же может быть потенциальная утечка информации на сторону.

Смотрите по ситуации. Если вы не работаете с гостайной, миллиардными корпоративными секретами или персональными данными (или работаете, но только в пределах сети и компьютеров компании), то в подавляющем большинстве случаев подобную защиту можно спокойно отключить.

Массовая загрузка данных

Это тема, на которой проявляется в буквальном смысле вся мощь Power Query. Здесь мы:

- научимся загружать данные в Power Query из **всех файлов заданной папки** и её подпапок;
- разберем отличия при массовом импорте **CSV и Excel-файлов**;
- выясним, что делать, если нужно собрать **не один, а несколько или все листы** из каждой книги;
- освоим оптовую загрузку в Power Query **всех простых и «умных» таблиц** из текущего файла.

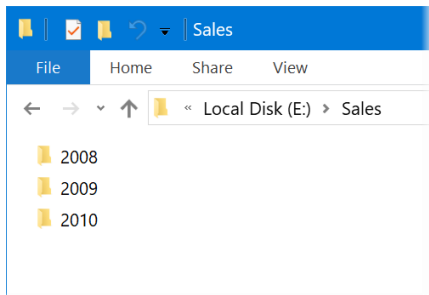


Импорт всех текстовых файлов из папки

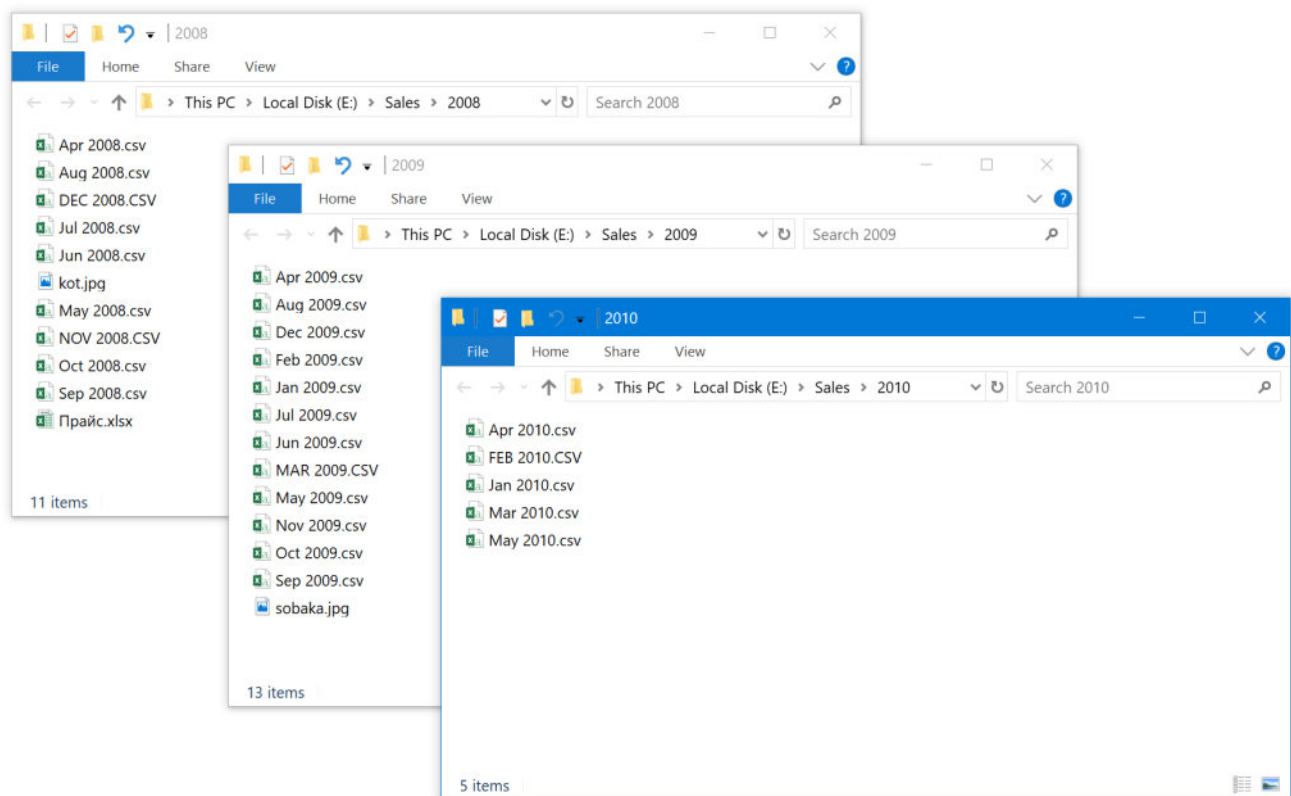
*Чтобы сделать что-то хорошее, надо сначала перестать делать ненужное.
(Максим Дорофеев)*

Постановка задачи

Допустим, что у нас есть на диске (локальном или сетевом) папка **Sales** с подпапками для каждого года:



В каждой папке находятся несколько текстовых файлов с названиями месяцев, откуда мы должны собрать данные:

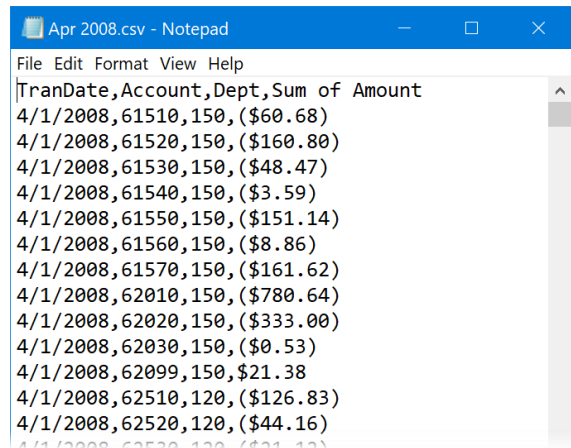


С ходу обратите внимание, что кроме нужных нам файлов с расширением **CSV** (Comma Separated Values = текст разделенный запятыми) в папках присутствуют также и ненужные (sobaka.jpg, kot.jpg, Прайс.xlsx). Кроме того, имена и расширения некоторых файлов введены заглавными буквами (FEB 2010.CSV и др.), что важно учесть в будущем, т. к. Power Query, если помните, строго относится к регистру.

Внутри каждый файл представляет собой текст, разделенный запятыми следующего вида:

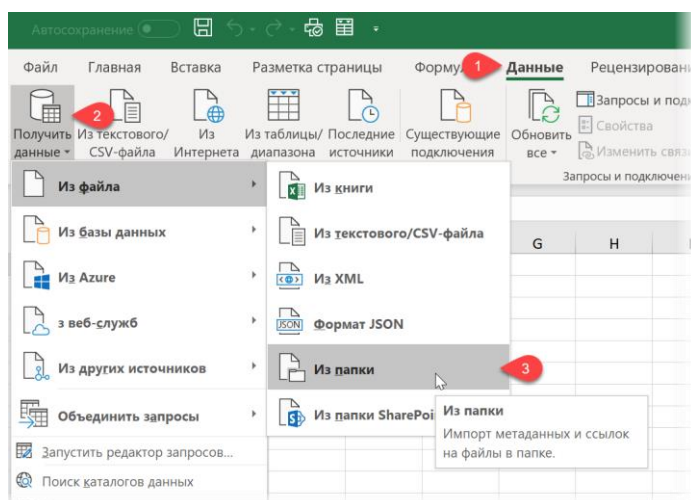
- первая строка – это шапка с названиями столбцов: дата транзакции (**TranDate**), номер счёта (**Account**), номер подразделения (**Dept**), сумма (**Sum of Amount**);
- даты введены через косую черту и в американском формате ММ/ДД/YYYY;
- отрицательные значения сумм введены в круглых скобках (стандарт записи расходов в МСФО);
- в каждом файле около 2000 строк.

Давайте попробуем собрать всё это в один файл Excel с помощью Power Query.

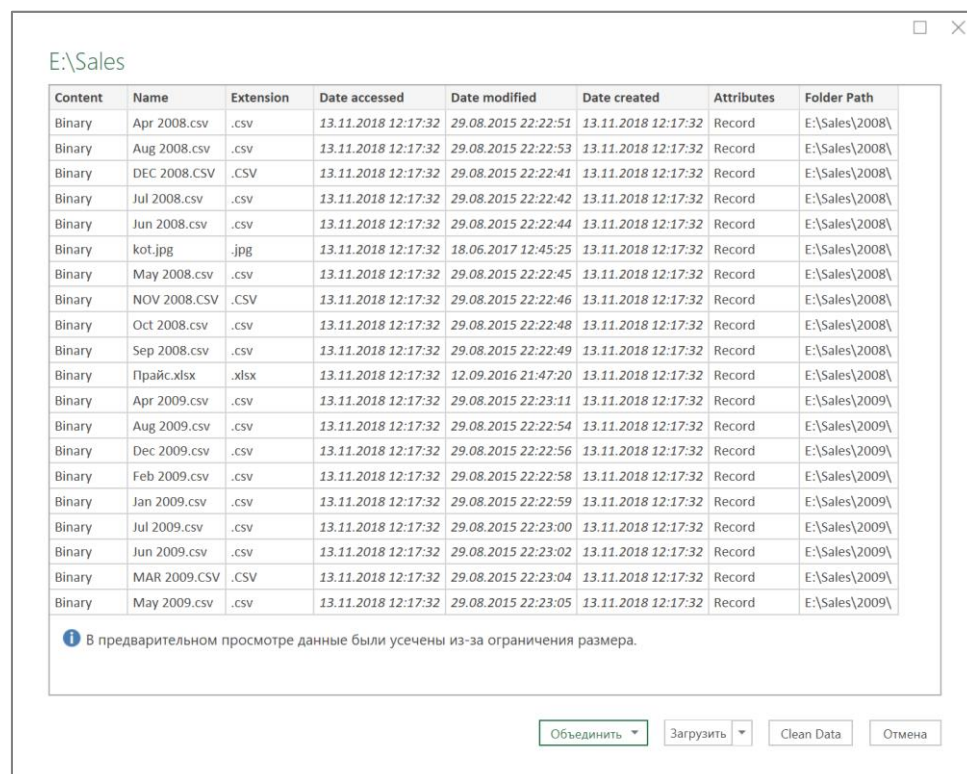


Отбираем нужные файлы

Выберем на вкладке **Данные** команду **Получить данные → Из файла → Из папки** (Data → Get Data → From File → From Folder):



Затем укажем в появившемся окне корневую папку **Sales** и после нажатия на **ОК** увидим в окне список файлов, которые у нас есть:



Поскольку нам нужно ещё дополнительно исключить ненужные файлы и обработать содержимое нужных, то жмем кнопку **Преобразовать данные (Transform Data)** или **Изменить (Edit)** в правом нижнем углу и попадаем в редактор запросов Power Query:

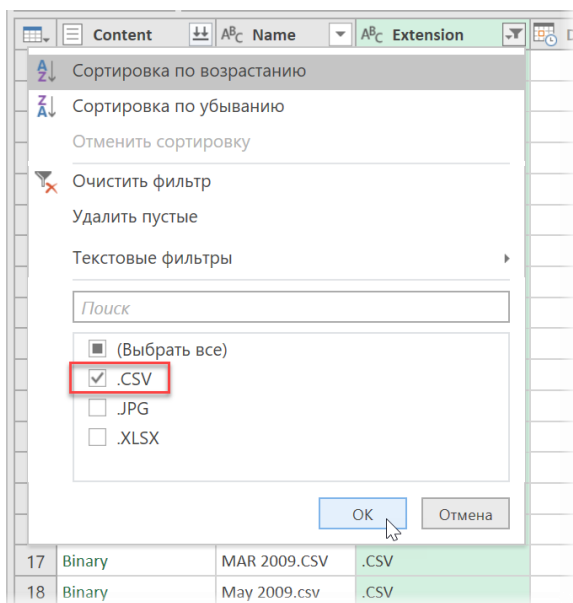
	Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
1	Binary	Apr 2008.csv	.csv	13.11.2018 12:17:32	29.08.2015 22:22:51	13.11.2018 12:17:32	Record	E:\Sales\2008\
2	Binary	Aug 2008.csv	.csv	13.11.2018 12:17:32	29.08.2015 22:22:53	13.11.2018 12:17:32	Record	E:\Sales\2008\
3	Binary	DEC 2008.CSV	.CSV	13.11.2018 12:17:32	29.08.2015 22:22:41	13.11.2018 12:17:32	Record	E:\Sales\2008\
4	Binary	Jul 2008.csv	.csv	13.11.2018 12:17:32	29.08.2015 22:22:42	13.11.2018 12:17:32	Record	E:\Sales\2008\
5	Binary	Jun 2008.csv	.csv	13.11.2018 12:17:32	29.08.2015 22:22:44	13.11.2018 12:17:32	Record	E:\Sales\2008\
6	Binary	kot.jpg	.jpg	13.11.2018 12:17:32	18.06.2017 12:45:25	13.11.2018 12:17:32	Record	E:\Sales\2008\
7	Binary	May 2008.csv	.csv	13.11.2018 12:17:32	29.08.2015 22:22:45	13.11.2018 12:17:32	Record	E:\Sales\2008\

Как видите, здесь представлен полный список всех найденных в папке Sales и её подпапках файлов с кучей подробностей и параметров по каждому из них, таких, как дата создания, изменения, атрибуты (только чтение и т. д.), полный путь.

На этом этапе наша задача состоит в том, чтобы отобрать только нужные нам файлы, исключив прочий информационный мусор в виде кошечек-собачек и т. д. Естественно, первая мысль, наверное, будет в данной ситуации о фильтре по расширению нужных нам файлов, т. к. все они CSV. Однако надо учитывать тот момент, что некоторые имена файлов и их расширения введены прописными буквами, а некоторые – строчными. Чисто теоретически можно даже допустить в будущем появление файлов с расширениями вида: Csv, cSv, csV, CsV и т.д.

Чтобы не перебирать все безумные вариации регистра в этом слове из трех букв, пойдем на тактическую хитрость: выделим столбец **Extension** и выберем на вкладке **Преобразование → Формат → ВЕРХНИЙ РЕГИСТР (Transform → Format → UPPER CASE)**, чтобы сконвертировать всё его содержимое в единый стиль.

Теперь можно смело отфильтровать только нужные нам файлы, просто развернув фильтр по столбцу **Extension** и поставив флажок только напротив **CSV**:

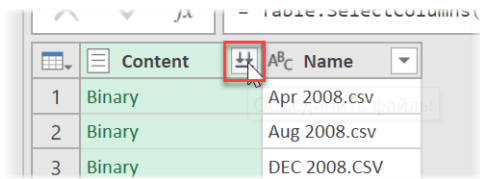


И давайте избавимся от балласта.

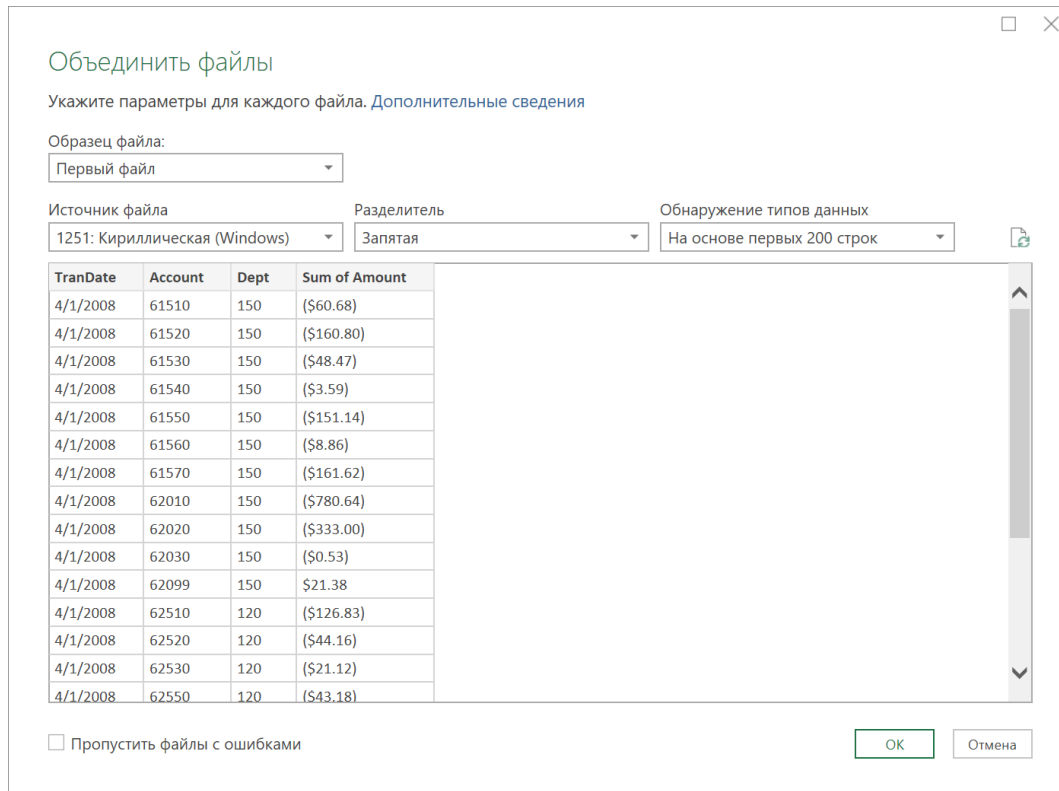
В столбце **Content**, собственно, и хранится содержимое каждого файла, так что он нам нужен точно. Кроме того, полезно потом бывает знать, из какого именно файла какая строка попала в нашу итоговую сборку, так что колонку **Name** тоже можно оставить. В будущем из нее можно будет сделать дату, если нужно. Остальные столбцы сейчас не содержат для нас полезной информации, поэтому их можно смело удалить. Для этого проще всего выделить (удерживая **Ctrl**) первые две колонки, а затем щелкнуть по их заголовку правой кнопкой мыши и выбрать команду **Удалить другие столбцы (Remove Other Columns)**.

Разворачиваем содержимое файлов

Теперь воспользуемся кнопкой с двойными стрелками в шапке столбца **Content**, чтобы развернуть содержимое каждого файла, спрятанное под словом **Binary**:



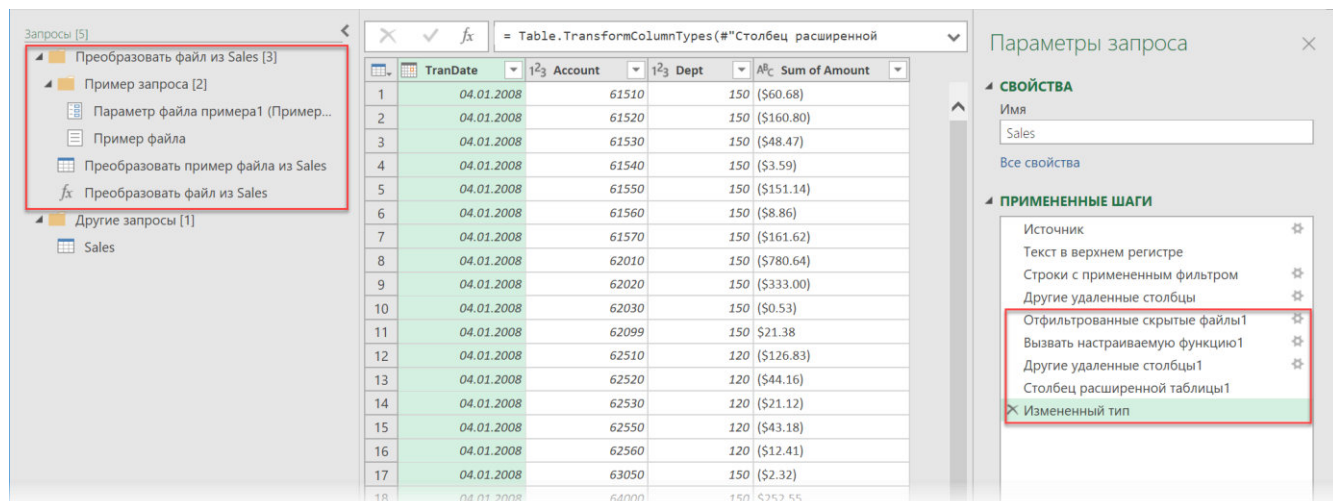
На экране появится окно с предварительным просмотром содержимого на примере первого файла из нашего списка:



В этом окне нужно:

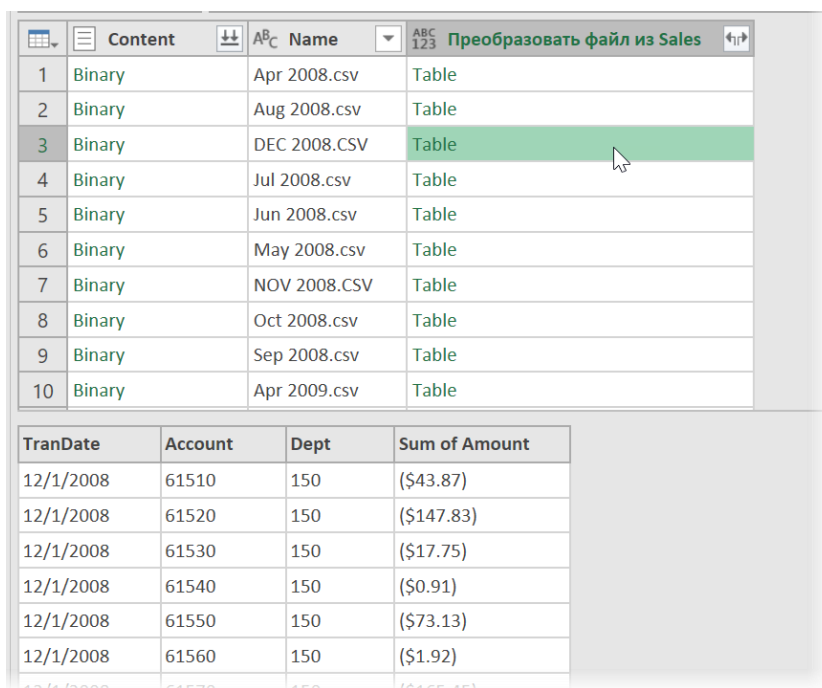
- проверить, правильно ли Power Query определил кодировку исходных файлов. Если вы видите вместо русского текста непонятные иероглифы или пустые квадратики, то значит, надо «поиграть» выпадающим списком **Источник файла (Source)**;
- проверить, правильно ли определился символ-разделитель (в нашем случае это запятая). Если неправильно, то выбрать нужный разделитель из списка или задать свой;
- решить, что делать с файлами, на которых будут возникать ошибки. Например, если в файле другая структура данных, то Power Query может либо игнорировать их и идти дальше (флажок **Пропустить файлы с ошибками**), либо останавливать запрос и выводить сообщение об ошибке.

После нажатия на **OK** через некоторое время мы увидим собранные из всех файлов данные в столбцах **TranDate**, **Account**, **Dept** и **Sum of Amount**. Плюс ко всему в левой половине экрана появится новая папка в панели **Запросы** и сразу несколько шагов в списке **Примененные шаги (Applied Steps)** в правой панели **Параметры запроса**:



Давайте-ка рассмотрим их повнимательнее и разберемся с тем, что тут случилось «под капотом» на самом деле. Когда мы нажали **ОК** в предыдущем окне, то произошла целая цепочка событий:

1. Power Query на примере первого файла сделал функцию для импорта данных – её можно увидеть в левой панели под именем **Преобразовать файл из Sales** (Transform Sample File from Sales) с характерным значком *fx*. В этой функции записаны параметры подключения и загрузки данных, в частности кодировка файла, символ-разделитель, то, что первая строка является шапкой, и т. д.
2. Чтобы созданную функцию можно было применить потом по очереди ко всем имеющимся файлам, имя файла было заведено как параметр – его можно увидеть в левой панели как **Параметр файла примера1** (Sample File Parameter).
3. Все скрытые файлы (если таковые были) были отфильтрованы – шаг **Отфильтрованные скрытые файлы1** (Filtered Hidden Files) в правой панели.
4. Созданная функция была применена по очереди ко всем оставшимся файлам, что добавило к нашей таблице столбец с названием **Преобразовать файл из Sales** (Transform File from Sales) с содержащимися в каждой ячейке вложенными таблицами, загруженными из каждого файла. Увидеть их можно, если выделить в правой панели шаг **Вызвать настраиваемую функцию1** (Invoke Custom Function1):



5. На шаге **Другие удаленные столбцы1** (Removed Other Columns1) были удалены все столбцы, кроме **Преобразовать файл из Sales** (и наш полезный столбец **Name** с именем файла тоже удалился).

6. Содержимое вложенных таблиц было развернуто (так же, как мы делали ранее с помощью кнопки с двойными стрелками в шапке таблицы).
7. К столбцам получившейся сборной таблицы Power Query попытался автоматически применить соответствующие форматы данных – шаг **Измененный тип (Changed Type)**.

Мощно, не правда ли? Столько всего случилось, оказывается!

Ну, и с ходу видны два момента, которые надо поправить:

- На шаге **Другие удаленные столбцы1 (Removed Other Columns1)** были удалены все столбцы, кроме **Преобразовать файл из Sales**, и наш полезный столбец **Name** с именем файла тоже удалился, что не радует. Исправить это можно легко, просто нажав на значок шестеренки справа от шага **Другие удаленные столбцы1** и добавив галочку рядом с **Name**.
- Power Query хотя и попытался, но не смог корректно определить форматы данных в столбцах (посмотрите хотя бы на даты в первом столбце: это должно было быть 1 апреля, а получилось 4 января), поэтому последний шаг **Измененный тип** можно смело удалить. Вместо этого лучше настроить форматы для каждого столбца вручную, учитывая американский формат даты и отрицательные числа в скобках. Для этого во втором и последнем столбцах из выпадающего списка форматов нужно выбрать вариант **Используя локаль (Use local)** → **Английский (США)**:

	ABC Name	ABC TranDate	ABC Account	ABC Dep
1	Apr 2008.csv	1.2 Десятичное число		150
2	Apr 2008.csv	\$ Валюта		150
3	Apr 2008.csv	123 Целое число		150
4	Apr 2008.csv	% Процент		150
5	Apr 2008.csv	📅 Дата и время		150
6	Apr 2008.csv	📅 Дата		150
7	Apr 2008.csv	🕒 Время		150
8	Apr 2008.csv	🌐 Дата, время и часовой пояс		150
9	Apr 2008.csv	🕒 Продолжительность		150
10	Apr 2008.csv	ABC Текст		150
11	Apr 2008.csv	☑ Истина/ложь		150
12	Apr 2008.csv	☑ Двоичный		120
13	Apr 2008.csv	Используя локаль...		120
14	Apr 2008.csv	4/1/2008	62530	120

Для пущей красоты можно ещё переименовать заголовки столбцов двойным щелчком, и после этого собранные данные примут, наконец, желаемый вид:

	ABC Файл	📅 Дата	123 Счет	123 Отдел	\$ Сумма
1	Apr 2008.csv	01.04.2008	61510	150	-60,68
2	Apr 2008.csv	01.04.2008	61520	150	-160,8
3	Apr 2008.csv	01.04.2008	61530	150	-48,47
4	Apr 2008.csv	01.04.2008	61540	150	-3,59
5	Apr 2008.csv	01.04.2008	61550	150	-151,14
6	Apr 2008.csv	01.04.2008	61560	150	-8,86
7	Apr 2008.csv	01.04.2008	61570	150	-161,62
8	Apr 2008.csv	01.04.2008	62010	150	-780,64
9	Apr 2008.csv	01.04.2008	62020	150	-333
10	Apr 2008.csv	01.04.2008	62030	150	-0,53
11	Apr 2008.csv	01.04.2008	62099	150	21,38
12	Apr 2008.csv	01.04.2008	62510	120	-126,83
13	Apr 2008.csv	01.04.2008	62520	120	-44,16

Выгружаем в Excel и ловим ошибки

Теперь давайте попробуем выгрузить получившуюся таблицу обратно на лист Excel. Для этого выберем команду **Главная → Закрывать и загрузить (Home → Close & Load)**. На первый взгляд всё хорошо, но потом в правом нижнем углу нас ждет неприятный сюрприз:

№	Файл	Дата	Счет	Отдел	Сумма
2	Apr 2008.csv	01.04.2008	61510	150	-60,68
3	Apr 2008.csv	01.04.2008	61520	150	-160,8
4	Apr 2008.csv	01.04.2008	61530	150	-48,47
5	Apr 2008.csv	01.04.2008	61540	150	-3,59
6	Apr 2008.csv	01.04.2008	61550	150	-151,14
7	Apr 2008.csv	01.04.2008	61560	150	-8,86
8	Apr 2008.csv	01.04.2008	61570	150	-161,62
9	Apr 2008.csv	01.04.2008	62010	150	-780,64
10	Apr 2008.csv	01.04.2008	62020	150	-333
11	Apr 2008.csv	01.04.2008	62030	150	-0,53
12	Apr 2008.csv	01.04.2008	62099	150	21,38
13	Apr 2008.csv	01.04.2008	62510	120	-126,83
14	Apr 2008.csv	01.04.2008	62520	120	-44,16
15	Apr 2008.csv	01.04.2008	62530	120	-21,12
16	Apr 2008.csv	01.04.2008	62550	120	-43,18
17	Apr 2008.csv	01.04.2008	62560	120	-12,41
18	Apr 2008.csv	01.04.2008	63050	150	-2,32
19	Apr 2008.csv	01.04.2008	64000	150	252,55
20	Apr 2008.csv	01.04.2008	64010	150	18,06
21	Apr 2008.csv	01.04.2008	64020	150	6,83
22	Apr 2008.csv	01.04.2008	65540	110	-513,77
23	Apr 2008.csv	01.04.2008	65550	110	-55,42

26 ошибок, Карл!

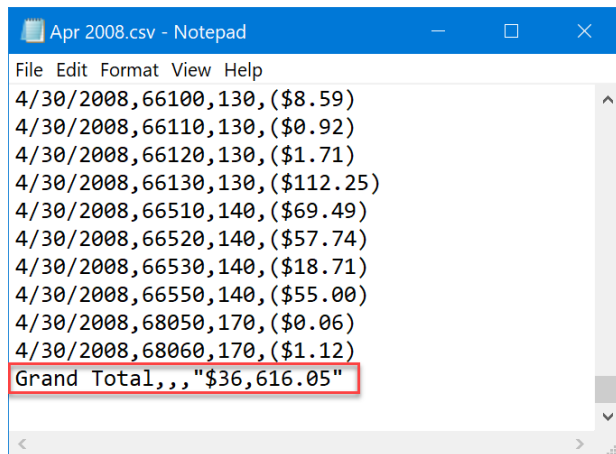
Как же так? Всё так хорошо начиналось!

Возьмем себя в руки попробуем разобраться в причинах неудачи. Для этого щелкнем мышкой прямо в эти злосчастные слова **«26 ошибок»**, т. к. они являются на самом деле гиперссылкой. Power Query сформирует для нас еще один запрос, цинично назвав его **Ошибки в Sales**, и отобразит в нём те самые 26 строк, где возникли проблемы:

№	Номер строки	Файл	Дата	Счет	Отдел	Сумма
1	2250	Apr 2008.csv	Error		null	36616,05
2	4010	Aug 2008.csv	Error		null	-124911,77
3	5934	DEC 2008.CSV	Error		null	100220,36
4	8364	Jul 2008.csv	Error		null	-36681,2
5	10607	Jun 2008.csv	Error		null	-56486,01
6	13008	May 2008.csv	Error		null	87985,65
7	15088	NOV 2008.CSV	Error		null	47399,19
8	17409	Oct 2008.csv	Error		null	90496,79
9	19625	Sep 2008.csv	Error		null	-58284,4
10	21967	Apr 2009.csv	Error		null	77193,37
11	24279	Aug 2009.csv	Error		null	-133457,17
12	26413	Dec 2009.csv	Error		null	169389,86
13	28334	Feb 2009.csv	Error		null	35577,84
14	30247	Jan 2009.csv	Error		null	89238,09

Если щелкнуть мышью в белый фон любой ячейки со словом Error, то в нижней части окна на желтом фоне мы увидим описание ошибки. Формулировка несколько мутная, но суть, я думаю, вы ухватите: Power Query на шаге преобразования первого столбца в дату не смог это сделать для слов «Grand Total».

«Какой ещё Grand Total?» - скажете вы, а потом ещё разок откроете проверить свои исходные файлы и, конечно же, обнаружите в конце каждого из них строчку с итогами, которую мы с вами не заметили ранее:

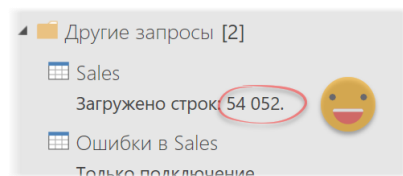


У нас в папках всего 26 файлов, в каждом из них по Grand Total – вот и причина 26 ошибок в запросе. Бинго!

Исправить ситуацию очень легко. Вернемся через левую панель в запрос **Sales**, выделим столбец с датой и выберем команду **Главная → Удалить строки → Удалить ошибки** (Home → Remove Rows → Remove Errors), которая просто-напросто удалит строки с ошибками в этом столбце. Или же можно сразу после сбора данных ещё до преобразования в дату просто отфильтровать в первом столбце строки со словами *Grand Total*.

После этого, обновив наш запрос ещё разок, увидим долгожданные результаты.

И ни одной ошибки!



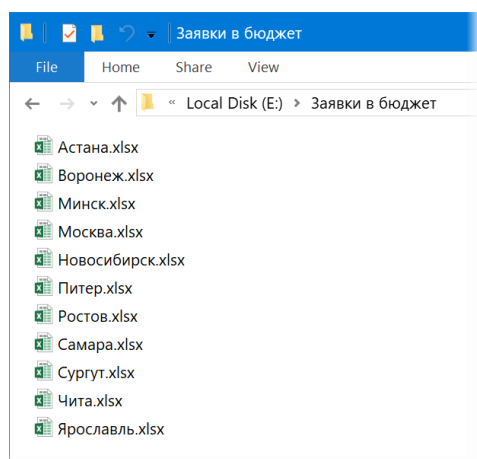
Сбор данных из всех Excel-файлов заданной папки

В отличие от сбора данных из текстовых файлов, который мы разбирали в прошлой главе, массовая загрузка информации из Excel-книг происходит немного иначе и имеет свои плюсы и минусы. Минусы состоят в том, что в Excel-файлах есть листы, которые могут быть видимыми или скрытыми, иметь различные названия и т. д., и это надо учитывать. Плюс же в том, что это привычный нам экселевский лист с ячейками с данными в определенном формате, а не слипшийся текст непонятного вида.

Давайте разберем весь процесс поэтапно от и до.

Постановка задачи

Предположим, что у нас есть локальная или сетевая папка, куда собираются заявки в бюджет от различных подразделений (филиалов, «дочек» компании):



В реальности количество файлов может быть существенно больше, или они могут быть рассортированы ещё и по вложенным папкам, это не важно.

В каждом файле может быть один или несколько листов, где в более-менее типовой форме расписан бюджет филиала или одного из офисов компании. Вот так, например, выглядит содержимое файла **Москва.xlsx**:

	A	B	C	D	E	F	G	H	I	J
1	Заявка в бюджет									
2										
3										
4	Статья	Подстатья	январь	февраль	март	апрель	май	июнь	июль	август
5	Управленческие	оплата труда	1 731	1 752	1 756	1 812	1 516	1 164	1 104	1 60
6		бонусы	535	483	431	760	1 010	114	656	58
7		аренда служ.авто.	119	500	616	328	748	569	882	49
8		социальные нужды	916	556	129	573	636	920	361	94
9		командировки	740	333	907	484	99	549	126	10
10		услуги связи	174	110	432	379	472	426	319	19
11		интернет	286	942	685	241	748	131	540	31
12		представительские	139	23	406	169	809	514	343	90
13		ИТОГО	4 640	4 699	5 362	4 746	6 038	4 387	4 331	5 15
14										
15	Хозяйственные	оплата труда	1 270	1 998	1 103	1 187	1 025	1 813	1 854	1 85
16		социальные нужды	86	461	424	768	353	944	871	72
17		аренда	657	810	300	680	474	844	322	3
18		ремонт и благоустройство	859	592	202	972	849	911	581	87
19		охрана	128	439	239	321	259	200	432	43
20		прочие	150	307	355	314	176	310	256	22
21		ИТОГО	3 150	4 607	2 623	4 242	3 136	5 022	4 316	4 15
22										
23	Коммерческие	упаковка	670	295	829	925	284	372	707	76
24		доставка	871	244	997	938	922	112	961	32
25		страхование	197	260	781	239	811	207	426	34

А вот так выглядит начинка файла **Астана.xlsx**:

	A	B	C	D	E	F	G	H	I	J
1	Заявка в бюджет									
2										
3										
4	Статья	Подстатья	январь	февраль	март	апрель	май	июнь	июль	август
5	Управленческие	командировки	877	820	380	343	318	276	705	91
6		услуги связи	192	123	203	295	373	335	250	10
7		оплата труда	1 545	1 118	1 366	1 170	1 711	1 081	1 278	1 51
8		интернет	788	875	137	557	421	495	408	96
9		социальные нужды	415	941	864	131	907	569	323	52
10		ИТОГО	3 817	3 877	2 950	2 496	3 730	2 756	2 964	4 02
11										
12	Хозяйственные	аренда	820	19	112	57	253	915	176	54
13		ремонт и благоустройство	510	29	147	21	907	146	931	25
14		социальные нужды	296	447	631	883	975	902	955	92
15		прочие	245	180	436	232	121	358	288	13
16		охрана	399	252	174	376	145	140	153	43
17		оплата труда	1 398	1 111	1 761	1 257	1 336	1 879	1 935	1 16
18		ИТОГО	3 668	2 038	3 261	2 826	3 737	4 340	4 438	3 44
19										
20	Коммерческие	упаковка	137	305	134	865	614	368	590	79
21		доставка	311	266	709	67	26	979	830	67
22		страхование	432	681	287	701	924	666	198	28
23		хранение	489	485	552	973	171	789	251	48
24		реклама	570	756	609	953	712	826	652	68
25		ИТОГО	1 939	2 493	2 291	3 559	2 447	3 628	2 521	2 92

Обратите внимание на следующие нюансы:

- в каждом файле может быть один или несколько листов с разными именами (названия офисов);
- одни и те же статьи расходов могут быть в разных файлах в разных строчках;
- набор статей расходов в файлах различается (например, в Астане нет подстатьи «Бонусы» в управленческих расходах, а в Москве они есть). Соответственно, общее число строк в файлах тоже не совпадает;
- после каждой статьи идет серая строка с итогом;
- по столбцам структура файлов похожа.

Наша конечная задача – собрать данные со всех (или только нужных) листов из всех файлов в одну таблицу и построить по ней сводную для анализа расходов по статьям-городам-месяцам.

Формируем список файлов

Как и в прошлой главе, всё начинается в новом пустом файле с выбора команды **Получить данные → Из файла → Из папки** на вкладке **Данные** (Data → Get Data → From file → From folder). Затем мы указываем положение исходной папки **Заявки в бюджет** и загружаем в Power Query список всех найденных там файлов:

	Content	APC Name	APC Extension	Date accessed	Date modified	Date created	Attributes	APC Folder Path
1	Binary	Астана.xlsx	.xlsx	16.11.2018 13:36:40	16.11.2018 13:30:22	16.11.2018 13:36:40	Record	E:\Заявки в бюджет\
2	Binary	Воронеж.xlsx	.xlsx	16.11.2018 13:36:40	05.09.2017 15:00:40	16.11.2018 13:36:40	Record	E:\Заявки в бюджет\
3	Binary	Минск.xlsx	.xlsx	16.11.2018 13:36:40	22.08.2017 14:42:03	16.11.2018 13:36:40	Record	E:\Заявки в бюджет\
4	Binary	Москва.xlsx	.xlsx	16.11.2018 13:36:40	16.11.2018 13:34:23	16.11.2018 13:36:40	Record	E:\Заявки в бюджет\
5	Binary	Новосибирск.xlsx	.xlsx	16.11.2018 13:36:40	05.09.2017 14:50:15	16.11.2018 13:36:40	Record	E:\Заявки в бюджет\
6	Binary	Питер.xlsx	.xlsx	16.11.2018 13:36:40	22.08.2017 14:22:50	16.11.2018 13:36:40	Record	E:\Заявки в бюджет\
7	Binary	Ростов.xlsx	.xlsx	16.11.2018 13:36:40	22.08.2017 14:22:49	16.11.2018 13:36:40	Record	E:\Заявки в бюджет\
8	Binary	Самара.xlsx	.xlsx	16.11.2018 13:36:41	22.08.2017 14:22:48	16.11.2018 13:36:41	Record	E:\Заявки в бюджет\
9	Binary	Сургут.xlsx	.xlsx	16.11.2018 13:36:41	22.08.2017 14:22:46	16.11.2018 13:36:41	Record	E:\Заявки в бюджет\
10	Binary	Чита.xlsx	.xlsx	16.11.2018 13:36:41	22.08.2017 14:22:43	16.11.2018 13:36:41	Record	E:\Заявки в бюджет\
11	Binary	Ярославль.xlsx	.xlsx	16.11.2018 13:36:41	22.08.2017 14:22:40	16.11.2018 13:36:41	Record	E:\Заявки в бюджет\

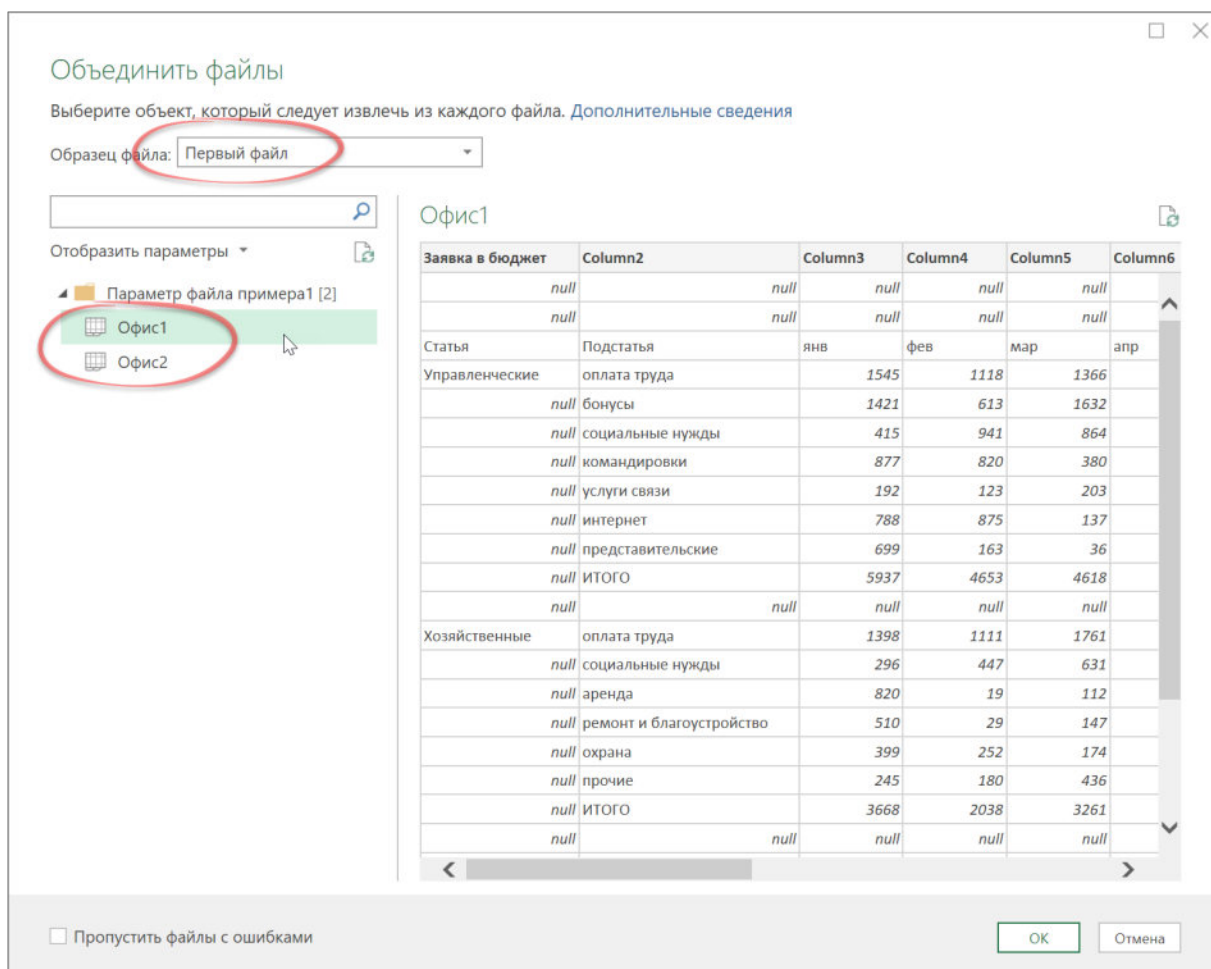
Если кроме файлов с данными в этом списке есть ещё какие-то ненужные файлы, случайно оказавшиеся в нашей папке или её подпапках, то их на этом шаге можно дополнительно отфильтровать, например, по расширению в столбце *Extension*, как мы делали в прошлой главе. Возможно также дополнительно отбирать нужные файлы по дате изменения или любым другим признакам.

После этого можно смело избавиться от всех столбцов, кроме первых двух. Имя файла в колонке *Name* пригодится нам позже для анализа расходов по городам, а в колонке *Content* под словом **Binary** скрывается собственно содержимое каждой книги, которое нам и нужно получить.

Извлекаем содержимое каждого файла

Чтобы вытащить данные из каждого файла, можно пойти двумя путями.

Первый – использовать уже знакомую нам кнопку с двойными стрелками в шапке столбца *Content*, чтобы развернуть содержимое каждой книги. В этом случае на экране появится окно **Навигатора**, где нужно будет на примере первого файла (Астана.xlsx) выбрать листы, которые мы хотим извлечь из каждой книги:

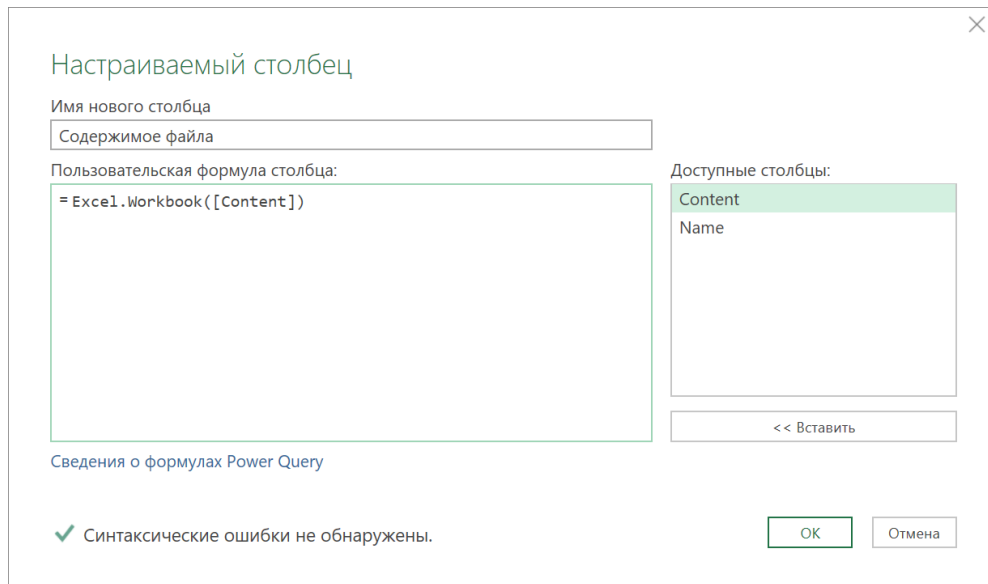


Очевидно, что тут кроется большая проблема: если мы выберем сейчас листы *Офис1* и *Офис2* для загрузки, то Power Query будет в дальнейшем импортировать данные только с них. А в других файлах листов может быть больше или меньше и называться они могут по-другому! Все это приведет к ошибкам при загрузке. Так что такой вариант можно посоветовать только в тех случаях, когда вы на 100% уверены, что во всех файлах у вас полное совпадение имён листов и их количества.

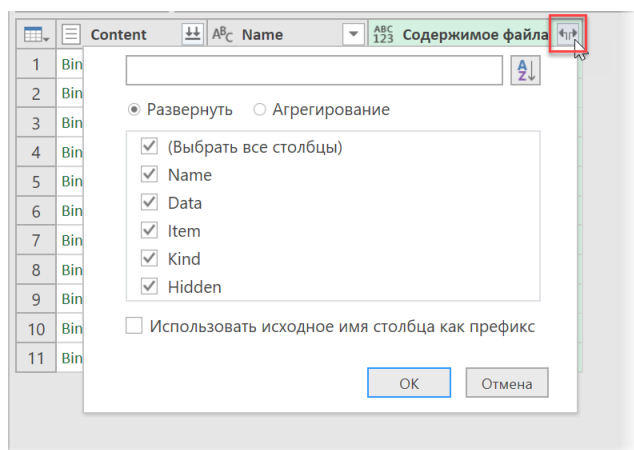
Если же рассматривать реальную жизненную ситуацию (разное количество листов с различающимися именами в каждом файле), то лучше пойти другим путём – использовать для сбора данных простую функцию из встроенного в Power Query языка M.

Для этого идём на вкладку **Добавление столбца** → **Настраиваемый столбец** (Add Column → Custom Column) и введём в открывшемся окне имя нового столбца и следующую формулу (соблюдая регистр!):

```
=Excel.Workbook([Content])
```



Эта функция извлекает из указанного в качестве аргумента файла (из столбца **[Content]**) всё его содержимое и отображает его в виде вложенной в ячейку таблицы. Развернуть все таблицы можно с помощью кнопки с двойными стрелками в шапке таблицы, выбрав затем все столбцы и сняв галочку **Использовать исходное имя столбца как префикс** (Use original column name as prefix):



После нажатия на **OK** к нашему исходному списку файлов добавится несколько новых столбцов и строк:

	Content	Name	Name.1	Data	Item	Kind	Hidden
1	Binary	Астана.xlsx	Офис1	Table	Офис1	Sheet	FALSE
2	Binary	Астана.xlsx	Офис2	Table	Офис2	Sheet	FALSE
3	Binary	Астана.xlsx	Лист2	Table	Лист2	Sheet	TRUE
4	Binary	Воронеж.xlsx	Офис на ул.Ленина	Table	Офис на ул.Ленина	Sheet	FALSE
5	Binary	Минск.xlsx	Офис	Table	Офис	Sheet	FALSE
6	Binary	Минск.xlsx	Лист1	Table	Лист1	Sheet	FALSE
7	Binary	Минск.xlsx	Таблица1	Table	Таблица1	Table	FALSE
8	Binary	Минск.xlsx	_xlnm.Print_Area	Table	Офис!_xlnm.Print_Area	DefinedName	FALSE
9	Binary	Москва.xlsx	Центральный Офис	Table	Центральный Офис	Sheet	FALSE
10	Binary	Москва.xlsx	Офис на Кутузовском	Table	Офис на Кутузовском	Sheet	FALSE
11	Binary	Москва.xlsx	Офис в Бутово	Table	Офис в Бутово	Sheet	FALSE
12	Binary	Новосибирск.xlsx	Главный офис	Table	Главный офис	Sheet	FALSE
13	Binary	Питер.xlsx	Офис на Невском	Table	Офис на Невском	Sheet	FALSE
14	Binary	Питер.xlsx	Второй офис на Петроградке	Table	Второй офис на Петроградке	Sheet	FALSE
15	Binary	Питер.xlsx	расходы	Table	расходы	DefinedName	FALSE
16	Binary	Ростов.xlsx	Ростовский офис	Table	Ростовский офис	Sheet	FALSE
17	Binary	Самара.xlsx	Офис на ул.Революции	Table	Офис на ул.Революции	Sheet	FALSE
18	Binary	Сургут.xlsx	офис	Table	офис	Sheet	FALSE
19	Binary	Чита.xlsx	Чита (офис)	Table	Чита (офис)	Sheet	FALSE
20	Binary	Ярославль.xlsx	Ярославский офис	Table	Ярославский офис	Sheet	FALSE

Отбираем нужные листы

Давайте поподробнее рассмотрим, что мы получили. Power Query видит в других файлах четыре типа объектов, обозначая каждый из них соответствующим словом в столбце **Kind** (тип):

- Листы (Sheet)
- Именованные диапазоны (Defined Name)
- Умные таблицы (Table)
- Области печати (Defined Name, где в имени есть Print_Area)

Соответственно, каждая строка, представлявшая собой файл, в новой таблице размножилась по количеству найденных в каждом файле подобных объектов. Так, например, в файле **Минск.xlsx** у нас два листа **Офис** и **Лист1**, «умная» таблица с именем **Таблица1** и область печати на листе **Офис**.

Конечно же, собирать информацию мы можем из всех вышеперечисленных объектов, но на практике чаще всего это будут листы, как в нашем примере. Давайте отберем только видимые листы, где в названии листа содержится слово «офис». Для этого отфильтруем:

- слово *Sheet* в столбце **Kind**;
- слово *FALSE* в столбце **Hidden**;
- все строки, содержащие слово «офис» в столбце **Item**. Чтобы учесть все возможные варианты с разным регистром (*офис*, *Офис*, *ОФИС* и т. д.), лучше сначала конвертировать весь текст в этом столбце, например, в нижний регистр командой **Преобразование → Формат → нижний регистр** (**Transform → Format → lower case**), а потом уже фильтровать по слову «офис» (маленькими буквами, соответственно).

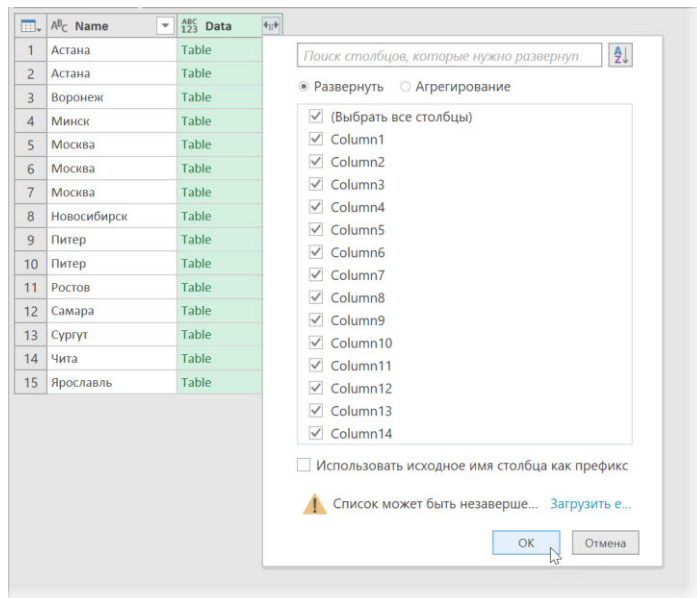
В итоге на экране должны остаться только нужные нам листы:

	Content	Name	Name.1	Data	Item	Kind	Hidden
1	Binary	Астана.xlsx	Офис1	Table	офис1	Sheet	FALSE
2	Binary	Астана.xlsx	Офис2	Table	офис2	Sheet	FALSE
3	Binary	Воронеж.xlsx	Офис на ул.Ленина	Table	офис на ул.ленина	Sheet	FALSE
4	Binary	Минск.xlsx	Офис	Table	офис	Sheet	FALSE
5	Binary	Москва.xlsx	Центральный Офис	Table	центральный офис	Sheet	FALSE
6	Binary	Москва.xlsx	Офис на Кутузовском	Table	офис на кутузовском	Sheet	FALSE
7	Binary	Москва.xlsx	Офис в Бутово	Table	офис в бутово	Sheet	FALSE
8	Binary	Новосибирск.xlsx	Главный офис	Table	главный офис	Sheet	FALSE
9	Binary	Питер.xlsx	Офис на Невском	Table	офис на невском	Sheet	FALSE
10	Binary	Питер.xlsx	Второй офис на Петроградке	Table	второй офис на петроградке	Sheet	FALSE
11	Binary	Ростов.xlsx	Ростовский офис	Table	ростовский офис	Sheet	FALSE
12	Binary	Самара.xlsx	Офис на ул.Революции	Table	офис на ул.революции	Sheet	FALSE
13	Binary	Сургут.xlsx	офис	Table	офис	Sheet	FALSE
14	Binary	Чита.xlsx	Чита (офис)	Table	чита (офис)	Sheet	FALSE
15	Binary	Ярославль.xlsx	Ярославский офис	Table	ярославский офис	Sheet	FALSE

Теперь можно смело удалить все ненужные столбцы, оставив только колонки **Name** (с именем файла) и **Data** (содержимое листа). В именах файлов можно удалить расширение, щелкнув по заголовку столбца правой кнопкой мыши и выбрав команду **Замена значений** (**Replace values**).

Разворачиваем таблицы и «причёсываем» результаты

Осталось совсем немного. Содержимое каждого листа хранится в столбце **Data** во вложенных таблицах, которые можно развернуть уже знакомым нам образом – щелчком по кнопке с двойными стрелками в шапке таблицы. В открывшемся окне оставляем все галочки для столбцов и снимаем флажок **Использовать исходное имя столбца как префикс** (**Use original column name as prefix**):



После нажатия на **OK** все наши таблицы со всех собранных листов из всех файлов соберутся в одну мегатаблицу вот такого вида:

ABC 123	ABC 123	ABC 123	ABC 123	ABC 123	ABC 123	ABC 123	ABC 123	ABC 123	ABC 123	ABC 123	ABC 123
1	Астана	Заявка в бюджет	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9
2	Астана		null	null	null	null	null	null	null	null	null
3	Астана		null	null	null	null	null	null	null	null	null
4	Астана	Статья	Подстатья	январь	февраль	март	апрель	май	июнь	июль	
5	Астана	Управленческие	оплата труда	1545	1118	1366	1170	1711	1081		
6	Астана		бонусы	1421	613	1632	564	842	1139		
7	Астана		социальные нужды	415	941	864	131	907	569		
8	Астана		командировки	877	820	380	343	318	276		
9	Астана		услуги связи	192	123	203	295	373	335		
10	Астана		интернет	788	875	137	557	421	495		
11	Астана		представительские	699	163	36	888	308	170		
12	Астана		ИТОГО	5937	4653	4618	3948	4880	4065		
13	Астана			null	null	null	null	null	null	null	null
14	Астана	Хозяйственные	оплата труда	1398	1111	1761	1257	1336	1879		
15	Астана		социальные нужды	296	447	631	883	975	902		
16	Астана		аренда	820	19	112	57	253	915		
17	Астана		ремонт и благоустройство	510	29	147	21	907	146		
18	Астана		охрана	399	252	174	376	145	140		
19	Астана		прочие	245	180	436	232	121	358		
20	Астана		ИТОГО	3668	2038	3261	2826	3737	4340		
21	Астана			null	null	null	null	null	null	null	null
22	Астана	Коммерческие	упаковка	137	305	134	865	614	368		
23	Астана		доставка	311	266	709	67	26	979		
24	Астана		страхование	432	681	287	701	924	666		
25	Астана		хранение	489	485	552	973	171	789		
26	Астана		реклама	570	756	609	953	712	826		
27	Астана		ИТОГО	1939	2493	2291	3559	2447	3628		
28	Астана	Заявка в бюджет		null	null	null	null	null	null	null	null
29	Астана			null	null	null	null	null	null	null	null
30	Астана			null	null	null	null	null	null	null	null
31	Астана	Статья	Подстатья	январь	февраль	март	апрель	май	июнь	июль	
32	Астана	Управленческие	оплата труда	1545	1118	1366	1170	1711	1081		
33	Астана		бонусы	1421	613	1632	564	842	1139		
34	Астана		социальные нужды	415	941	864	131	907	569		
35	Астана		командировки	877	820	380	343	318	276		
36	Астана		услуги связи	192	123	203	295	373	335		
37	Астана		интернет	788	875	137	557	421	495		
38	Астана		представительские	699	163	36	888	308	170		

Осталось немного навести красоту:

1. Удалим три верхние ненужные строки, используя команду Главная → Удалить строки → Удаление верхних строк (Home → Remove Rows → Remove top rows).
2. Поднимем первую строку в шапку кнопкой Использовать первую строку в качестве заголовка на вкладке Главная (Home → Use first row as headers).
3. Переименуем первый столбец из Астана в Филиал.

- Удалим пустые строки, ненужные итоги и задублированные из каждого листа шапки таблиц с помощью фильтра по столбцу **Подстатья**, сняв флажки *NULL*, *ИТОГО* и *Подстатья* соответственно.
- Выделим второй столбец **Статья** и заполним пустые ячейки (null) предыдущими значениями, используя команду **Заполнить** → **Вниз** на вкладке **Преобразование** (Transform → Fill → Down).

Дополнительные улучшения для сводной

В принципе, на этом моменте можно было бы и остановиться, т. к. мы выполнили нашу главную задачу – собрали данные из всех файлов и привели их в вид, с которым можно работать, но есть одно небольшое но. Поскольку мы планируем построить по объединенным данным сводную таблицу, то было бы гораздо удобнее иметь в нашем списке не 12 отдельных столбцов-месяцев, а всего два: один столбец с названием месяца или датой, а другой с числовым значением в этом месяце, т. е сделать вот так:

	АБС Филиал	АБС Статья	АБС Подстатья	АБС Месяц	АБС 123 Значение
1	Астана	Управленческие	оплата труда	январь	1545
2	Астана	Управленческие	оплата труда	февраль	1118
3	Астана	Управленческие	оплата труда	март	1366
4	Астана	Управленческие	оплата труда	апрель	1170
5	Астана	Управленческие	оплата труда	май	1711
6	Астана	Управленческие	оплата труда	июнь	1081
7	Астана	Управленческие	оплата труда	июль	1278
8	Астана	Управленческие	оплата труда	август	1514
9	Астана	Управленческие	оплата труда	сентябрь	1643
10	Астана	Управленческие	оплата труда	октябрь	1057
11	Астана	Управленческие	оплата труда	ноябрь	1253
12	Астана	Управленческие	оплата труда	декабрь	1008
13	Астана	Управленческие	бонусы	январь	1421
14	Астана	Управленческие	бонусы	февраль	613
15	Астана	Управленческие	бонусы	март	1632
16	Астана	Управленческие	бонусы	апрель	564
17	Астана	Управленческие	бонусы	май	842
18	Астана	Управленческие	бонусы	июнь	1139
19	Астана	Управленческие	бонусы	июль	596
20	Астана	Управленческие	бонусы	август	1130

В таком виде таблица станет, как вы понимаете, в 12 раз длиннее и труднее для человеческого восприятия, но вот сводную таблицу по ней строить станет существенно удобнее. В терминах баз данных такое преобразование иногда называют конвертацией *кросс-таблицы* (от слова cross – пересечение) в *плоскую* (flat table)¹.

Чтобы выполнить такой редизайн, выделим первых три столбца и, щёлкнув по их заголовкам правой кнопкой мыши, выберем команду **Отменить свертывание других столбцов** (Unpivot Other Columns).

Ну, и останется переименовать получивший столбец **Атрибут** в **Месяц** для наглядности.

И ещё один момент.

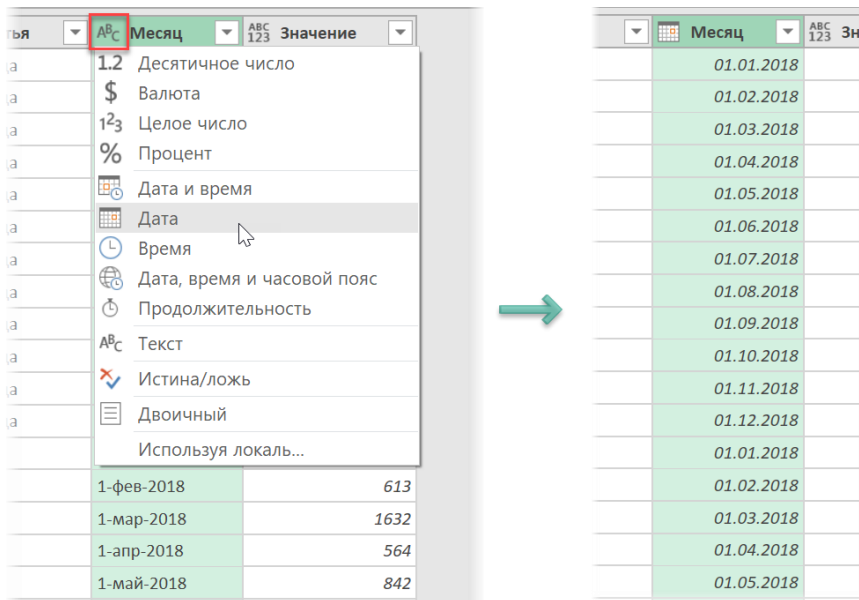
Было бы здорово превратить трехбуквенные названия месяцев в настоящую дату (например, в первое число каждого месяца 2018 года). Это даст возможность в будущем полноценно сортировать, фильтровать и группировать нашу сводную по месяцам, кварталам и т. д., поскольку сейчас даже простой сортировки месяцев сделать невозможно: Excel выстроит их по алфавиту: авг, апр, дек... вместо нормального календаря.

Чтобы это сделать, пойдём на небольшую тактическую хитрость. Выделим столбец **Месяц** и добавим к нему приставку «1-» в начало каждой ячейки, используя команду **Преобразование** → **Формат** → **Добавить префикс** (Transform → Format → Add Prefix).

Затем аналогичным образом добавим «-2018» в конец каждой даты, чтобы получить склейку вида 1-январь-2018.

И вот теперь можно совершенно спокойно поменять формат столбца на дату: такой текст Power Query уже сможет распознать и конвертировать:

¹ Подробнее техника такого преобразования разобрана в главе [Отмена свёртывания](#).



Останется выгрузить данные на лист или в режиме **Только создать подключение (Only create connection)**, используя команду **Главная → Закрывать и загрузить → Закрывать и загрузить в...** (Home → Close & Load → Close & Load to...) и построить по ним сводную таблицу, просуммировав бюджеты всех филиалов:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N		
1	Филиал	(Все)														
3	Сумма по полю	Значение	Назва													
4				+ янв	+ фев	+ мар	+ апр	+ май	+ июн	+ июл	+ авг	+ сен	+ окт	+ ноя	+ дек	Общий итог
5	Названия строк															
6	Коммерческие	36 513	35 097	42 251	37 346	37 330	35 637	37 526	40 528	41 505	36 185	40 633	39 980	460 531		
7	PR	540	1 020	1 355	648	585	1 402	221	584	1 649	367	826	1 251	10 448		
8	доставка	5 224	3 576	9 453	6 899	8 029	7 265	7 607	7 524	7 793	5 612	7 687	7 126	83 795		
9	реклама	8 528	9 621	6 673	7 581	8 565	8 344	8 184	9 277	9 328	8 911	8 269	7 988	101 269		
10	страхование	8 582	6 364	8 578	5 871	7 612	5 387	6 644	7 281	8 703	7 416	7 910	5 899	86 247		
11	упаковка	6 353	8 287	8 179	7 551	5 965	6 918	9 431	9 259	7 807	7 275	8 431	8 921	94 377		
12	хранение	7 286	6 229	8 013	8 796	6 574	6 321	5 439	6 603	6 225	6 604	7 510	8 795	84 395		
13	Управленческие	69 656	71 426	66 400	70 209	65 222	67 408	67 154	73 066	74 951	59 526	69 370	65 844	820 232		
14	аренда служ.авто.	940	1 525	1 069	2 251	1 377	1 600	1 520	1 754	2 452	934	740	1 186	17 348		
15	бонусы	14 825	11 504	12 525	12 754	10 718	12 639	10 285	12 413	13 102	8 004	13 280	11 494	143 543		
16	интернет	8 722	10 044	7 557	7 040	7 464	6 770	8 270	8 283	9 750	6 001	5 564	7 721	93 186		
17	командировки	8 496	8 908	7 439	7 080	5 348	7 125	6 884	10 234	6 797	6 836	8 049	8 700	91 896		
18	оплата труда	20 566	19 589	20 166	20 835	21 607	19 671	19 416	20 659	22 445	18 660	21 829	18 543	243 986		
19	представительские	6 696	6 403	7 068	9 364	6 512	9 173	7 933	7 953	7 192	7 773	7 529	7 901	91 497		
20	социальные нужды	6 323	9 344	6 205	5 614	8 464	6 666	7 976	8 404	8 317	7 129	7 917	6 698	89 057		
21	услуги связи	3 088	4 109	4 371	5 271	3 732	3 764	4 870	3 366	4 896	4 189	4 462	3 601	49 719		
22	Хозяйственные	50 054	48 274	51 450	49 420	50 282	53 015	54 581	52 750	50 218	53 845	50 785	48 847	613 521		
23	аренда	6 320	4 910	6 139	5 626	5 444	6 165	4 182	6 007	5 690	5 756	4 763	4 362	65 364		
24	оплата труда	18 988	19 601	20 357	19 858	19 779	21 501	22 946	21 806	21 724	21 861	21 799	22 454	252 674		
25	охрана	4 334	4 511	5 132	4 609	4 657	3 949	5 378	4 961	4 898	5 221	5 454	4 769	57 873		
26	прочие	5 201	5 269	5 026	4 853	4 895	5 550	4 597	4 293	5 021	6 110	4 969	5 755	61 539		
27	ремонт и благоустройство	7 357	6 945	5 527	6 062	8 009	6 603	8 965	6 202	6 783	6 684	5 398	5 776	80 311		
28	социальные нужды	7 854	7 038	9 269	8 412	7 498	9 247	8 513	9 481	6 102	8 213	8 402	5 731	95 760		
29	Общий итог	156 223	154 797	160 101	156 975	152 834	156 060	159 261	166 344	166 674	149 556	160 788	154 671	1 894 284		

Импорт всех «умных» таблиц из текущей книги

Постановка задачи

Давайте предположим, что нам с вами досталась для работы выгрузка из какой-либо программы с информацией по продажам автомобильных запчастей в разных городах и дилерских центрах (ДЦ) вот такого вида:

	A	B	C	D	E	F	G	H
1	Наименование	Категория	День	Город	ДЦ	Сумма	Менеджер	
2	Corolla	Toyota	15	Оренбург	Урал	1 256 800	Михайлов	
3	Corolla	Toyota	24	Оренбург	Урал	282 784	Михайлов	
4	Escape	Ford	8	Оренбург	Урал	957 472	Михайлов	
5	Escape	Ford	12	Оренбург	Урал	1 405 856	Михайлов	
6	Escape	Ford	14	Оренбург	Урал	398 176	Михайлов	
7	Fiesta	Ford	15	Оренбург	Урал	397 152	Дубинин	
8	Fiesta	Ford	25	Оренбург	Урал	1 526 976	Иванов	
9	Fiesta	Ford	27	Оренбург	Урал	1 449 376	Дубинин	
10	Focus	Ford	14	Оренбург	Урал	458 304	Михайлов	
11	i20	Hyundai	16	Оренбург	Восток	800 832	Дмитриенко	
12	i20	Hyundai	17	Оренбург	Восток	1 225 792	Тарасов	
13	i30	Hyundai	5	Оренбург	Восток	436 416	Дмитриенко	
14	i30	Hyundai	10	Оренбург	Восток	385 280	Тарасов	
15	i30	Hyundai	24	Оренбург	Восток	523 008	Волина	
16	iQ	Toyota	7	Оренбург	Восток	75 808	Дмитриенко	
17	iQ	Toyota	7	Оренбург	Восток	1 569 120	Тарасов	
18	iQ	Toyota	21	Оренбург	Восток	938 848	Волина	
19	Kuga	Ford	3	Оренбург	Восток	1 329 888	Волина	
20	Kuga	Ford	14	Оренбург	Восток	784 416	Волина	

Обратите внимание:

- на каждом листе данные находятся в виде «умной» таблицы;
- число строк на разных листах неодинаковое;
- положение и количество столбцов, а также шапка таблицы, наоборот, совершенно идентичны;
- количество листов может меняться.

Нам надо собрать все «умные» таблицы со всех имеющихся листов и объединить их в единую таблицу, с которой в дальнейшем можно будет работать (фильтровать, анализировать сводной и т. д.)

Формируем список таблиц

Если бы количество листов и, соответственно, таблиц было постоянным и небольшим (2–3 шт.), то можно было бы пойти простым путем: сделать к каждой из них отдельный запрос и объединить затем эти запросы в режиме добавления, как мы уже делали в главе [Добавление двух таблиц](#).

У нас же таблиц, во-первых, много, и делать к каждой из них отдельный запрос будет долго и скучно. А во-вторых, количество таблиц в нашем файле – величина переменная: сегодня их 5, а завтра может быть 10. Поэтому нам нужен другой механизм, который позволит загрузить в Power Query сразу все имеющиеся в книге «умные» таблицы одним махом.

Для этого на вкладке **Данные** выберем команду **Получить данные → Из других источников → Пустой запрос (Data → Get Data → From other sources → Blank query)** и в открывшемся окне в строку формул впишем функцию

```
=Excel.CurrentWorkbook()
```

Еще раз напомним, что встроенный в Power Query язык M регистрочувствительный, т. е. писать любые формулы и функции нужно с точностью до регистра. Если у вас в окне редактора не видно строки формул, то её можно включить на вкладке **Просмотр (View)**.

В окне должно отобразиться всё содержимое текущей книги (что-то похожее мы уже делали в главе [Загрузка данных из текущей книги Excel](#)). Мы увидим все «умные» таблицы, именованные диапазоны и области печати, содержащиеся в нашем файле. Если щелкнуть мышью в белый фон ячейки со словом Table, то в нижней части окна отобразится содержимое каждой таблицы.

The screenshot shows an Excel interface with a list of tables on the left and a data table on the right. The list of tables includes 'Table' (1-12), 'Январь|Область_печати', 'Прогноз', and 'Справочник'. The data table below has columns: Модель, Бренд, День, Город, ДЦ, Сумма, Менеджер.

Модель	Бренд	День	Город	ДЦ	Сумма	Менеджер
Escape	Ford	15	Оренбург	Урал	1040704	Михайлов
Explorer	Ford	15	Оренбург	Урал	2420000	Михайлов
Fiesta	Ford	13	Оренбург	Урал	1182816	Михайлов
Fiesta	Ford	22	Оренбург	Урал	673280	Тарасов
Fiesta	Ford	26	Оренбург	Урал	1995712	Михайлов
Focus	Ford	9	Оренбург	Урал	283776	Иванов

Очевидно, что Таблица1...12 – это как раз нужные нам данные по 12 месяцам, но что делать с лишними данными, представленными в конце списка в виде именованных диапазонов **Прогноз**, **Справочник** и области печати с листа **Январь**?

Ну, области печати можно отфильтровать по наличию фразы *Область_печати*, а что делать с именованными диапазонами? Отбирать таблицы, которые в имени содержат слово *Таблица*? А если у нас будут таблицы с другими, не столь явными признаками?

100% надежного способа отбора тут, наверное, нет и быть не может, но есть трюк, позволяющий отобразить нужные нам таблицы с очень высокой надежностью. Пощелкайте мышью в белый фон ячеек в первом столбце, глядя на внешний вид таблиц, появляющихся в нижней части экрана. У нужных нам «умных» таблиц будет отображаться шапка со словами **Модель**, **Бренд**, **День**, **Город** и т. д. (см. предыдущую иллюстрацию), а у «мусорных» таблиц эта шапка отсутствует, заменяясь стандартными **Column1,2,3...**

The close-up shows the list of tables with 'Table' (13-15) and 'Январь|Область_печати', 'Прогноз', 'Справочник'. Below, a data table has a red box around its header: 'Column1', 'Column2', 'Column3', 'Column4'.

Column1	Column2	Column3	Column4
Город	2018	2019	2020
Москва	9349023	4547344	9874623
Питер	5501062	671915	8383845
Оренбург	7449594	8022732	9867159

Таким образом, если мы сможем вытащить в отдельный столбец название, допустим, первого столбца из шапки, то по этой колонке затем будет легко отфильтровать нужные нам таблицы!

Чтобы это сделать, выберем на вкладке **Добавление столбца** команду **Настраиваемый столбец** (Add Column → Custom Column) и введем в открывшемся окне имя новой колонки и вот такую формулу на языке M:

```
=Table.ColumnNames([Content]){0}
```

ABC 123	Content	ABC 123	Name	ABC 123	Признак
1	Table		Таблица1		Модель
2	Table		Таблица2		Модель
3	Table		Таблица3		Модель
4	Table		Таблица4		Модель
5	Table		Таблица5		Модель
6	Table		Таблица6		Модель
7	Table		Таблица7		Модель
8	Table		Таблица8		Модель
9	Table		Таблица9		Модель
10	Table		Таблица10		Модель
11	Table		Таблица11		Модель
12	Table		Таблица12		Модель
13	Table		Январь!Область_печати		Column1
14	Table		Прогноз		Column1
15	Table		Справочник		Column1

Настраиваемый столбец

Имя нового столбца
Признак

Пользовательская формула столбца:
= Table.ColumnNames([Content]){0}

Доступные столбцы:
Content
Name

<< Вставить

Сведения о формулах Power Query

✓ Синтаксические ошибки не обнаружены.

OK Отмена

Давайте разберем её чуть подробнее.

1. Функция **Table.ColumnNames** требует в качестве аргумента таблицу (которые в нашем случае лежат в столбце **[Content]**) и выдает в качестве результата список названий столбцов из шапки, т. е. **{Модель, Бренд, День, Город, ДЦ, Сумма, Менеджер}**.
2. Поскольку нам для отбора достаточно названия первого столбца, то мы добавляем к нашей формуле **{0}**, чтобы извлечь первый элемент списка. **{0}**, а не **{1}** потому, что в Power Query нумерация почти всех объектов идёт с нуля, а не с единицы.

После ввода формулы и нажатия на **ОК** мы увидим столбец **Признак** с названием первого столбца в каждой таблице. По нему отфильтровать нужные нам данные не составит труда.

Разворачиваем таблицы

И остались сущие пустяки. Удалим все столбцы, кроме **Content**, и развернем содержимое всех вложенных таблиц **Table** с помощью кнопки с двойными стрелками в шапке этого столбца:

ABC 123	Content
1	Table
2	Table
3	Table
4	Table
5	Table
6	Table
7	Table
8	Table
9	Table
10	Table
11	Table
12	Table

Поиск столбцов, которые нужно развернуть

Развернуть Агрегирование

(Выбрать все столбцы)

Модель

Бренд

День

Город

ДЦ

Сумма

Менеджер

Использовать исходное имя столбца как префикс

⚠ Список может быть незавершенным... Загрузить е...

OK Отмена

Галочку **Использовать исходное имя столбца как префикс** (Use original column name as prefix) можно снять, и после нажатия на **ОК** мы увидим содержимое всех «умных» таблиц из книги, склеенное в единую целую длинную «простыню». Останется выгрузить собранные данные на лист с помощью команды **Главная → Заккрыть и загрузить** (Home → Close & Load):

	A	B	C	D	E	F	G	H
1	Модель	Бренд	День	Город	ДЦ	Сумма	Менеджер	
2	Corolla	Toyota	15	Оренбург	Урал	1256800	Михайлов	
3	Corolla	Toyota	24	Оренбург	Урал	282784	Михайлов	
4	Escape	Ford	8	Оренбург	Урал	957472	Михайлов	
5	Escape	Ford	12	Оренбург	Урал	1405856	Михайлов	
6	Escape	Ford	14	Оренбург	Урал	398176	Михайлов	
7	Fiesta	Ford	15	Оренбург	Урал	397152	Дубинин	
8	Fiesta	Ford	25	Оренбург	Урал	1526976	Иванов	
9	Fiesta	Ford	27	Оренбург	Урал	1449376	Дубинин	
10	Focus	Ford	14	Оренбург	Урал	458304	Михайлов	
11	i20	Hyundai	16	Оренбург	Восток	800832	Дмитриенко	
12	i20	Hyundai	17	Оренбург	Восток	1225792	Тарасов	
13	i30	Hyundai	5	Оренбург	Восток	436416	Дмитриенко	
14	i30	Hyundai	10	Оренбург	Восток	385280	Тарасов	
15	i30	Hyundai	24	Оренбург	Восток	523008	Волина	
16	iQ	Toyota	7	Оренбург	Восток	75808	Дмитриенко	
17	iQ	Toyota	7	Оренбург	Восток	1569120	Тарасов	
18	iQ	Toyota	21	Оренбург	Восток	938848	Волина	
19	Kuga	Ford	3	Оренбург	Восток	1329888	Волина	
20	Kuga	Ford	14	Оренбург	Восток	284416	Волина	
21	Mondeo	Ford	4	Оренбург	Восток	100416	Пушкарев	
22	Mondeo	Ford	13	Оренбург	Восток	951200	Дмитриенко	
23	Mondeo	Ford	14	Оренбург	Восток	412832	Тарасов	
24	Mondeo	Ford	17	Оренбург	Восток	1255200	Волина	

Запросы и п... x

Запросы | Подключения

1 запрос

Запрос1
Загружено строк: 1 399.

Красота?

Не совсем. Есть ещё one more thing, как любил говорить Стив Джобс.

Исключаем рекурсию

Налюбовавшись на полученные результаты, попробуем обновить наш запрос, щелкнув правой кнопкой мыши по нему в правой панели или прямо по таблице и выбрав команду **Обновить (Refresh)** или нажав сочетание клавиш **Ctrl+Alt+F5**. И после обновления увидим странную вещь: количество загруженных строк увеличится вдвое!

Запрос1
Загружено строк: 2 798.

Вот те раз! Как это? Откуда?

Если открыть наш запрос повторно, а затем перейти на первый шаг **Источник (Source)** и нажать сверху кнопку **Обновить предварительный просмотр (Update preview)**, то легко можно будет обнаружить причину такого поведения: полученная в результате нашего запроса «умная» таблица **Запрос1** тоже добавилась в сборку, удвоив таким образом объем загруженных данных:

	ABC Content	ABC Name
1	Table	Запрос1
2	Table	Таблица1
3	Table	Таблица2
4	Table	Таблица3
5	Table	Таблица4
6	Table	Таблица5
7	Table	Таблица6
8	Table	Таблица7

В программировании подобный трюк, когда функция вызывает саму себя, передавая себе же результаты работы, называется *рекурсией* и иногда используется во благо. В нашем же случае это явно лишнее.

Исправить проблему несложно. Находясь на шаге **Источник (Source)**, вставим в наш запрос ещё один дополнительный шаг, нажав на треугольник фильтра в шапке столбца **Name**, выбрав **Текстовые фильтры → Не содержит (Text filters → Does not contain)** и введя затем имя нашего запроса, чтобы исключить его из таблицы:

Фильтрация строк

Базовый Подробнее

Сохранять строки, в которых "Name"

не равно ABC Запрос1

И Или

ABC Введите или выберите з

OK Отмена

После нажатия на **OK** Power Query заботливо предупредит нас об опасности вставки шага в середину запроса, и после подтверждения и последующего обновления число строк вернётся к прежнему значению.

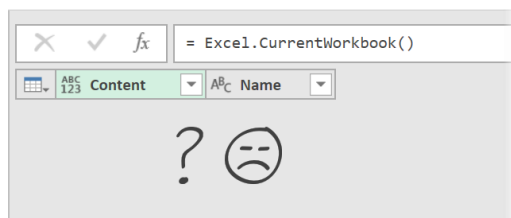
Загрузка всех простых таблиц с листов текущей книги

Но что делать, если нам нужно загрузить в Power Query не удобные для импорта «умные» таблицы, а обычные диапазоны, т. е. простые данные со всех (или избранных) листов текущей книги?

Разберем, как обычно, эту ситуацию на живом примере. Допустим, что у нас есть книга с большим количеством листов, где на каждом листе находится таблица с данными по сделкам в этом городе:

1	Товар	Дата	Стоимость	Менеджер
2	Плащ	31.05.2017	7882	Кирилл
3	Кардиган	07.02.2011	7207	Егор
4	Джинсы	13.10.2013	7567	Кристина
5	Комбинезон	24.01.2016	2382	Мирон
6	Лосины	17.11.2013	2083	Валерия
7	Поло	24.04.2013	3936	Артур
8	Шорты	06.01.2017	1268	Павел
9	Пиджак	09.01.2017	6031	Олеся
10	Бриджи	22.09.2013	8897	Есения
11	Пуховик	24.01.2015	948	Владимир
12	Свитер	25.10.2014	6854	Алиса
13	Майка	06.11.2013	8751	Илья
14	Пальто	26.03.2014	8220	Ксения
15	Кардиган	11.12.2015	4888	Никита
16	Пальто	10.07.2012	9525	Дарина
17	Юбка	17.10.2017	5325	Сергей
18	Туника	17.04.2013	4320	Кира
19	Плащ	27.08.2013	2258	Яна
20	Поло	17.12.2011	4828	Егор
21	Платье	19.06.2014	7131	Евгения

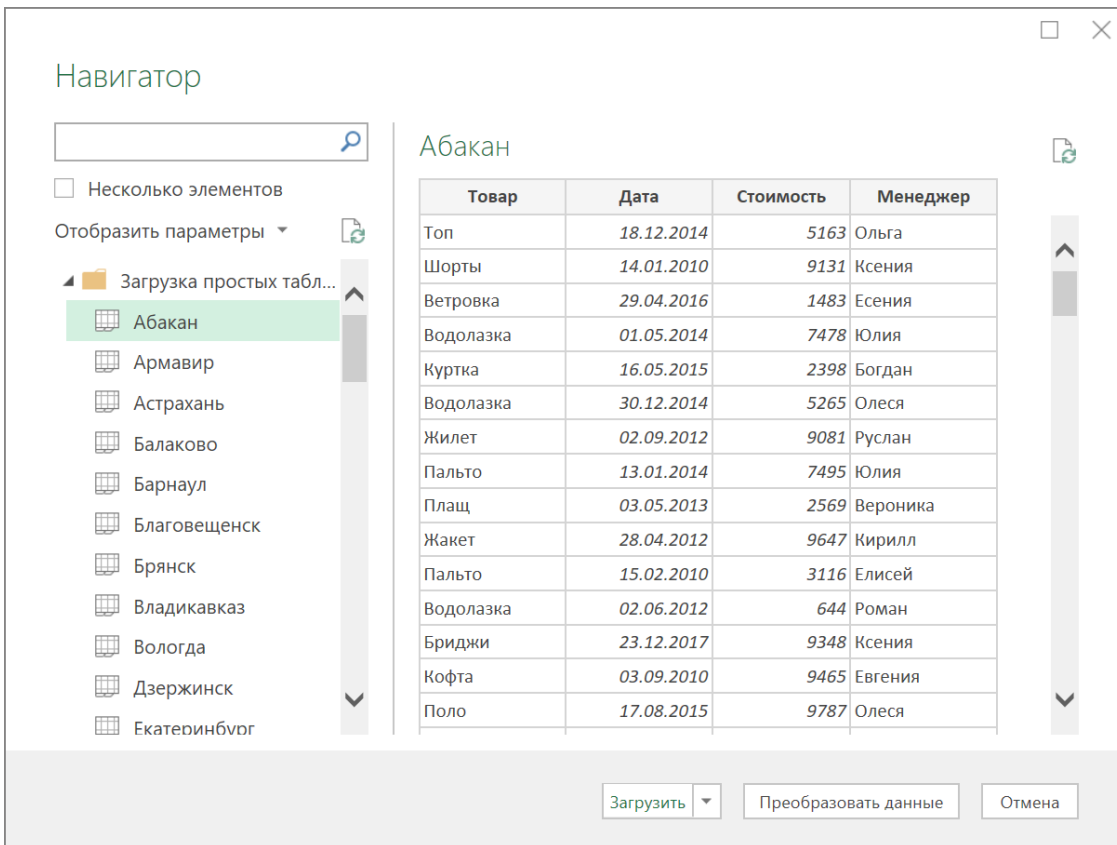
Проблема в том, что уже обкатанный нами в предыдущих главах универсальный способ, основанный на создании пустого запроса через **Данные → Получить данные → Из других источников → Пустой запрос (Data → Get Data → From Other Sources → Blank Query)** и последующее использование команды языка M **Excel.CurrentWorkbook**, выдает в результате пустую таблицу:



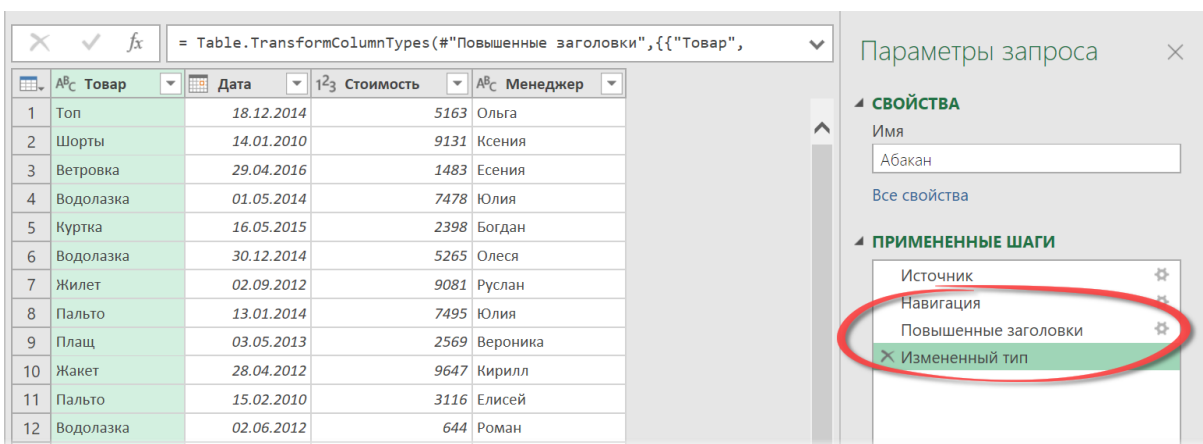
Это происходит потому, что функция **Excel.CurrentWorkbook** «видит» в текущей книге только «умные» таблицы, именованные диапазоны и области печати, но не листы. Что же делать, если в нашей книге десятки листов с обычными таблицами? Превращать вручную каждую из них в «умную»? Сколько времени на это уйдет? Да и не всегда можно это легко сделать, т. к. таблицы могут быть для этого непригодны (иметь многострочную шапку, промежуточные итоги и объединенные ячейки внутри и т. д.).

Выход заключается в использовании уже знакомой нам техники импорта из внешнего файла Excel через команду **Данные → Получить данные → Из файла → Excel (Data → Get Data → From file → Excel)**, но в качестве файла, из которого нужно загрузить данные, мы указываем эту же текущую книгу Excel, в которой создаем запрос!

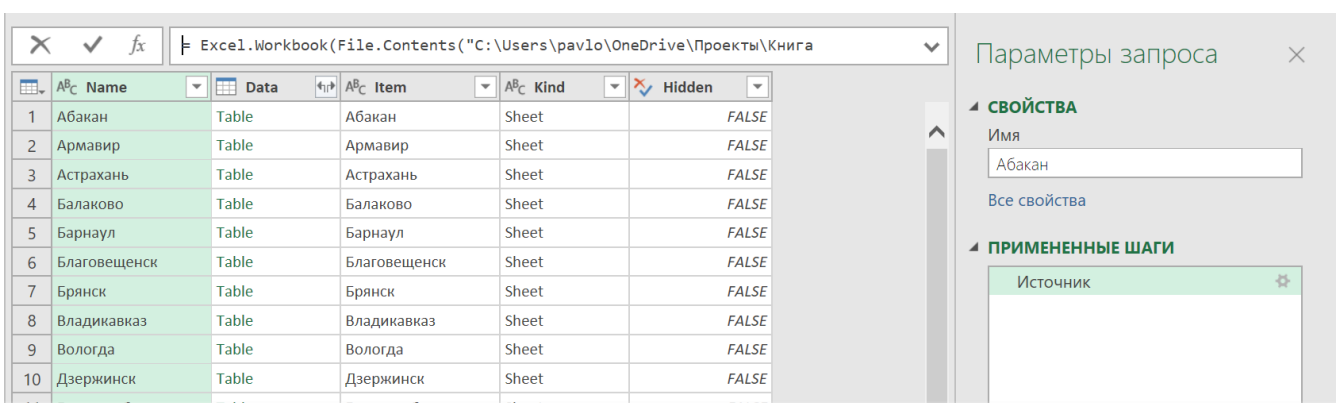
Тогда на экране мы увидим уже знакомое окно **Навигатора (Navigator)** со списком всех листов, которые есть в данном файле:



Выберем любой лист и нажмем кнопку **Преобразовать данные (Transform Data)** или **Изменить (Edit)** в правом нижнем углу, чтобы перейти в окно редактора запросов, где должны увидеть содержимое одного выбранного листа и 4 шага в панели справа:



А теперь удаляем все шаги, оставив только первый – **Источник (Source)**. Это приведёт к тому, что мы вернемся «к истокам» и увидим список всех листов, которые есть в книге. Их содержимое, свернутое в виде вложенных таблиц (Table), находится в столбце **Data**:



Ну, а дальше всё идёт по уже знакомой траектории:

1. Если нам нужны не все листы, то отбираем требуемые фильтрацией по столбцу **Name**.
2. Удаляем ненужные колонки, оставив только **Name** и **Data**.
3. Разворачиваем вложенные таблицы **Table** с содержимым каждого листа кнопкой с двойными стрелками в заголовке столбца **Data**.
4. Поднимаем первую строку данных в шапку, используя кнопку **Главная → Использовать первую строку в качестве заголовков (Use first row as headers)**.
5. Фильтруем задублированные в собранных данных шапки исходных таблиц, сняв флажок по слову **Товар** в фильтре столбца **Товар**.
6. Придумываем и вводим для нашего запроса наглядное имя, например *Сборщик*.

В итоге у нас на экране должна получиться вот такая картина:

Параметры запроса

Имя: Сборщик

Все свойства

ПРИМЕНЕННЫЕ ШАГИ

- Источник
- Другие удаленные столбцы
- Развернутый элемент Data
- Повышенные заголовки
- Измененный тип
- Переименованные столбцы
- Строки с примененным фильтром

Осталось последнее.

Если вы читали предыдущую главу про сбор данных из всех «умных» таблиц в книге, то должны помнить о рекурсии. Здесь у нас проблема аналогичная. Если выбрать в качестве финального места выгрузки новый лист, то он добавится к общему списку листов и после следующего обновления тоже попадет в сборку как *Лист1*. Также в сборку попадет результирующая таблица *Сборщик*, поскольку она «умная» и созданный на её основе именованный диапазон *External_Data_1*:

№	Name	Data	Item	Kind	Hidden
1	Лист1	Table	Лист1	Sheet	FALSE
2	Абакан	Table	Абакан	Sheet	FALSE
3	Армавир	Table	Армавир	Sheet	FALSE
4	Астрахань	Table	Астрахань	Sheet	FALSE
59	Элиста	Table	Элиста	Sheet	FALSE
60	Южно-Сахалинск	Table	Южно-Сахалинск	Sheet	FALSE
61	Ярославль	Table	Ярославль	Sheet	FALSE
62	Сборщик	Table	Сборщик	Table	FALSE
63	ExternalData_1	Table	Лист1!ExternalData...	DefinedName	TRUE

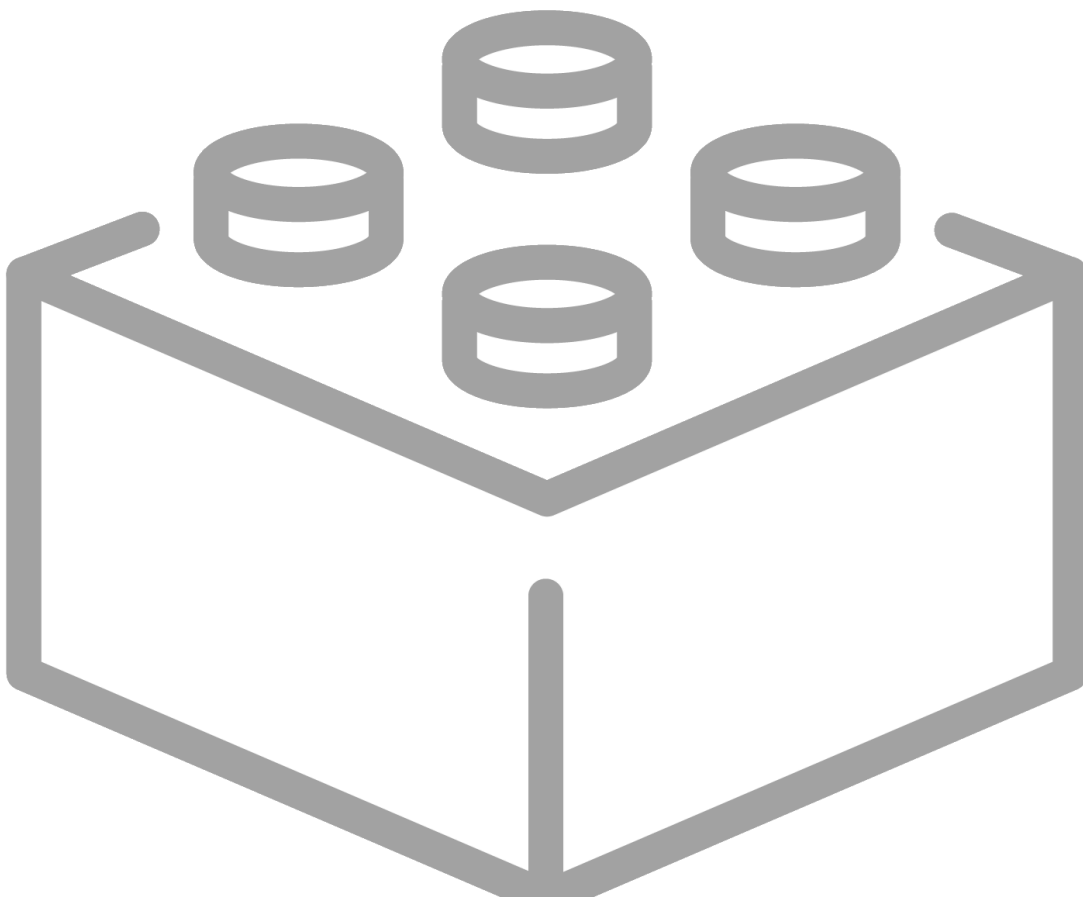
Чтобы этого не произошло, нужно выгружать результаты не на лист, а оставлять как подключение командой **Главная → Закреть и загрузить → Закреть и загрузить в → Только создать подключение (Home → Close & Load → Close & Load to... → Only Create Connection)**.

Другой вариант: вставить сразу после первого еще один дополнительный шаг, где уже собранные данные будут исключаться из результатов сборки. Для этого нужно отфильтровать только листы (Sheet) по столбцу **Kind** и снять флажок напротив *Лист1* в фильтре по столбцу **Name**.

Преобразования таблиц

После загрузки данных в Power Query их почти всегда нужно «довести до ума» и придать им нужный вид. Эта глава посвящена всевозможным трансформациям и преобразованиям, которые можно выполнить в Power Query, а именно:

- мы разберёмся, как **фильтровать** данные в Power Query, какие есть при этом подводные камни и в чём специфика фильтрации по сравнению с Excel;
- научимся быстро **транспонировать** таблицы и заполнять в них пустые ячейки;
- используем **группировку** для «схлопывания» наборов строк в промежуточные итоги;
- проясним вопрос **свёртывания и отмены свёртывания** таблиц: что это, зачем нужно и когда и как применяется;
- разберём на примере **сворачивание одномерного столбца в двумерную таблицу и подтягивание значений к краю** таблицы.



Фильтрация

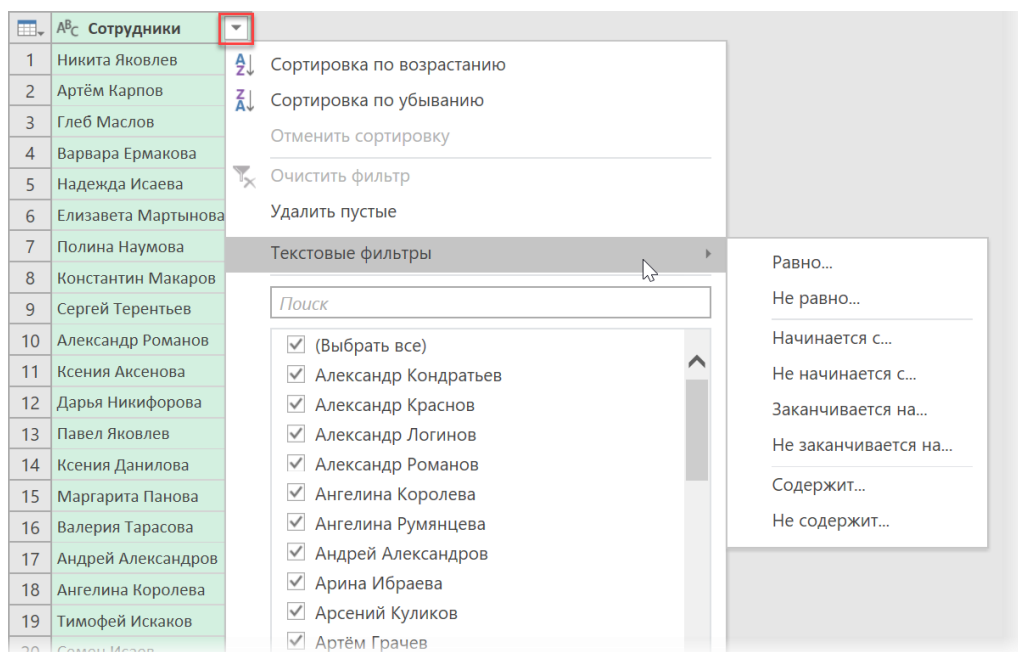
*Тот, кто передвигает горы, сначала убирает
мелкие камешки.
(Конфуций)*

Технически фильтрация ненужных данных в Power Query очень похожа на фильтрацию в Microsoft Excel, но имеет одно принципиальное отличие: если в Excel отфильтрованные ненужные строки скрываются, то в Power Query они удаляются. Таким образом, фильтрация здесь выступает как один из основных и самых удобных инструментов зачистки данных от информационного мусора и лишних строк.

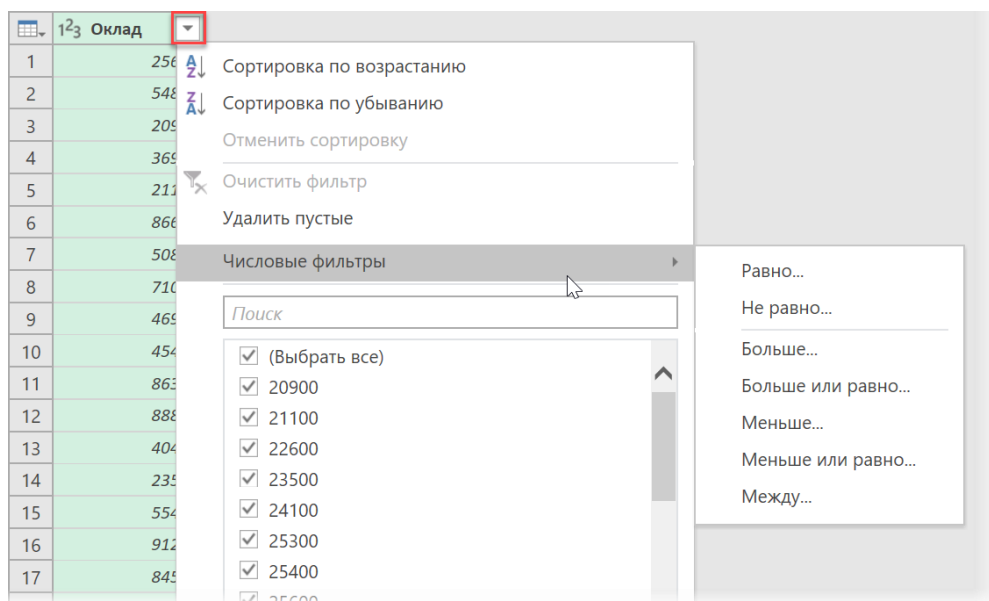
Фильтрация разных типов данных

Набор инструментов фильтрации, как и в Excel, зависит от типа данных в каждом столбце.

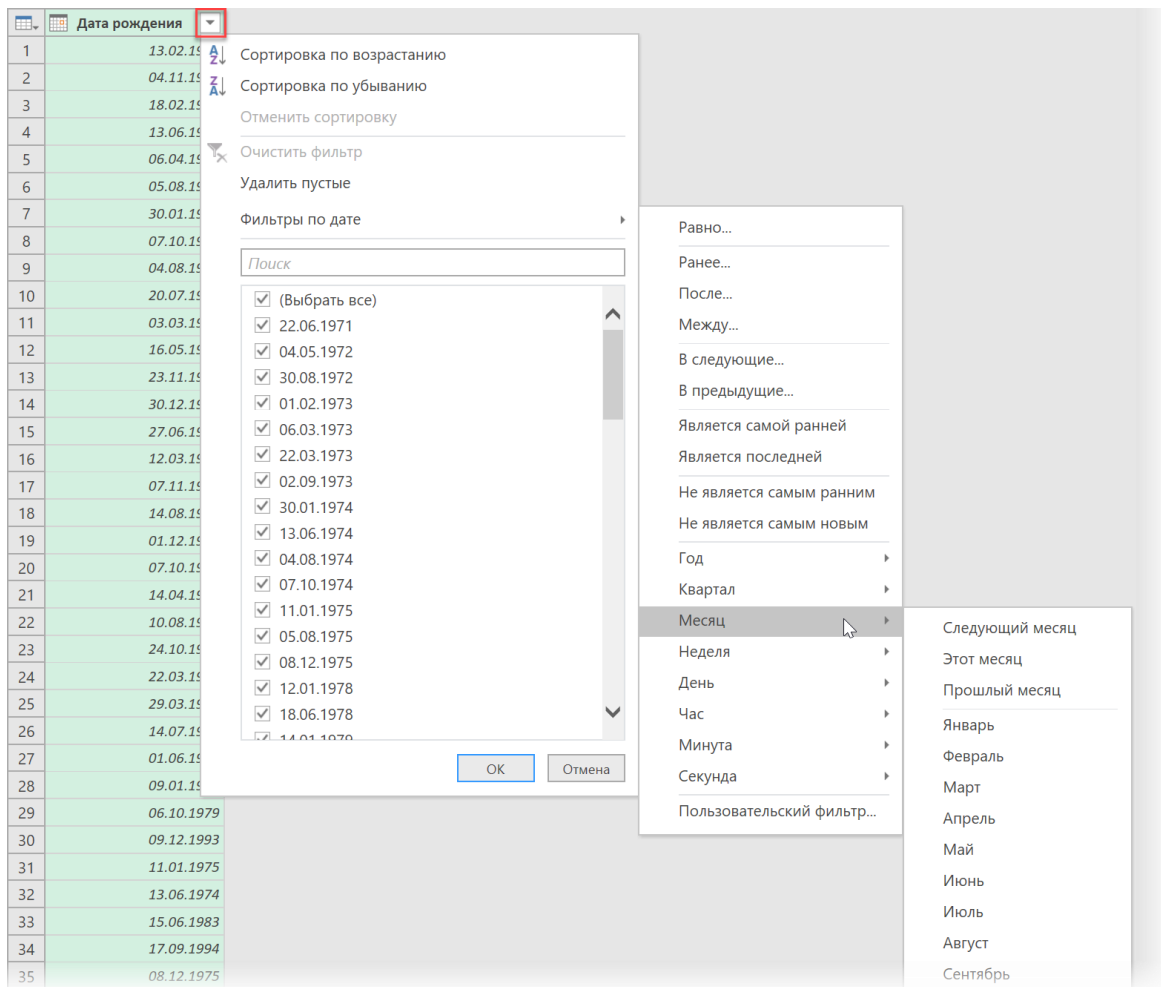
Если столбец с текстом, то кроме отбора галочками-флажками и поля **Поиск (Search)** нам доступны дополнительные возможности в разделе **Текстовые фильтры (Text filters)**:



Аналогичная ситуация будет, если мы фильтруем данные в числовом столбце, тогда в разделе **Числовые фильтры (Number filters)** будет возможность настроить более сложные критерии отбора:



И самое интересное нас ждёт, если мы будем фильтровать столбец с датами. В этом случае к нашим услугам очень большой набор различных возможностей:



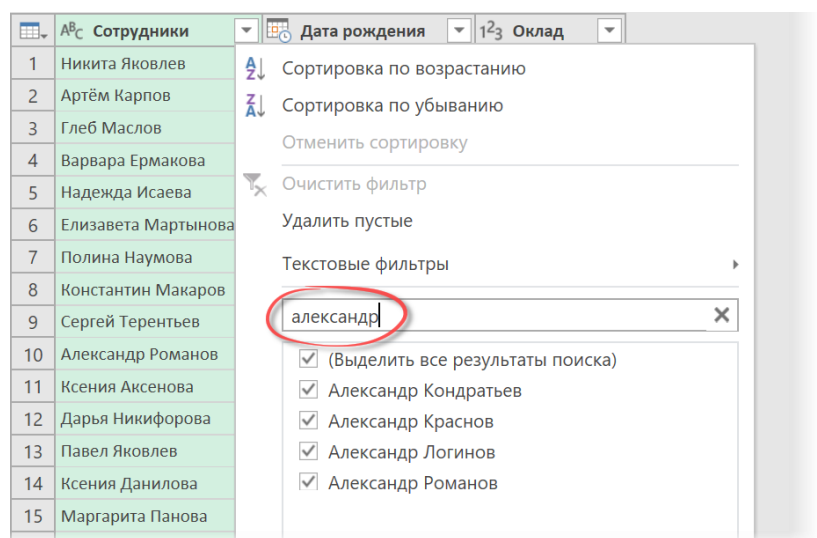
Разумеется, любые отфильтрованные строки удаляются не навсегда, а только из всех последующих шагов. Мы всегда можем вернуть отброшенную информацию, просто удалив шаг фильтрации или поменяв критерии отбора с помощью значка шестерёнки справа.

Опасная иллюзия с фильтрацией через поле «Поиск»

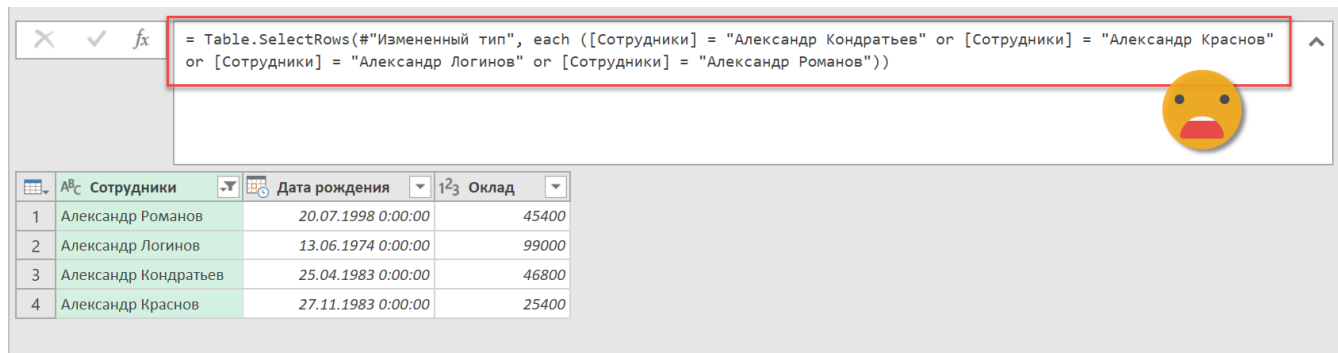
При всей внешней простоте и похожести на Excel у фильтрации в Power Query есть один подводный камень, незнание которого может привести в ваших запросах к очень печальным последствиям. Рассмотрим его на примере фильтрации имён сотрудников на предыдущей странице.

Предположим, что я хочу оставить в моих данных только всех *Александров*, чтобы потом работать с ними дальше. Первым рефлекторным желанием будет, скорее всего, ввести нужное имя в поле **Поиск** (оно, кстати, регистронечувствительно, во отличие от большинства функций Power Query).

После нажатия на **ОК** кажется, что мы получили именно то, что хотелось. — всех, у кого в имени встречается "александр".



Но если присмотреться к строке формул или открыть исходный код запроса на вкладке **Просмотр** → **Расширенный редактор** (View → Advanced Editor), то видно, что не всё так просто: Power Query прописал в команде фильтрации имена и фамилии этих конкретных четырёх *Александров*, вместо того чтобы сделать критерий вида «*имя содержит Александр*».

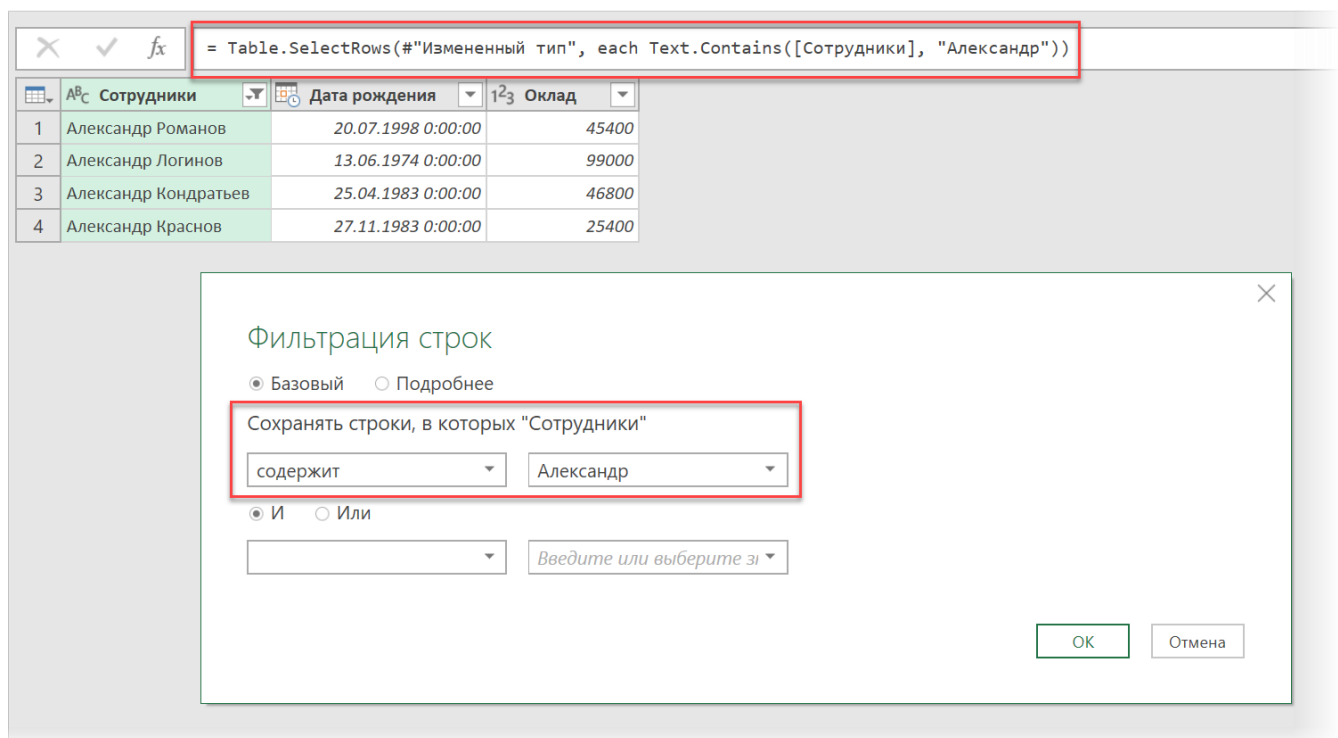


```
= Table.SelectRows("#Измененный тип", each ([Сотрудники] = "Александр Кондратьев" or [Сотрудники] = "Александр Краснов" or [Сотрудники] = "Александр Логинов" or [Сотрудники] = "Александр Романов"))
```

	АВС Сотрудники	Дата рождения	123 Оклад
1	Александр Романов	20.07.1998 0:00:00	45400
2	Александр Логинов	13.06.1974 0:00:00	99000
3	Александр Кондратьев	25.04.1983 0:00:00	46800
4	Александр Краснов	27.11.1983 0:00:00	25400

Проблема в том, что если в будущем к нашим данным добавится, например, «*Александр Пушкин*», то он не попадёт под действие такого фильтра и будет удален. Причем произойдет это молча и без каких-либо сообщений об ошибках! Последствия такого могут быть весьма печальными, а первопричину найти не всегда легко.

Чтобы выполнить фильтрацию правильно, нужно обязательно выбрать упомянутую выше опцию **Текстовые фильтры** → **Содержит** (Text filters → Contains) и ввести (уже с учётом регистра!) имя «*Александр*» в диалоговое окно поиска:



```
= Table.SelectRows("#Измененный тип", each Text.Contains([Сотрудники], "Александр"))
```

	АВС Сотрудники	Дата рождения	123 Оклад
1	Александр Романов	20.07.1998 0:00:00	45400
2	Александр Логинов	13.06.1974 0:00:00	99000
3	Александр Кондратьев	25.04.1983 0:00:00	46800
4	Александр Краснов	27.11.1983 0:00:00	25400

Фильтрация строк

Базовый Подробнее

Сохранять строки, в которых "Сотрудники"

содержит Александр

И Или

Введите или выберите из

OK Отмена

После нажатия на **OK** в строке формул будет хорошо видна разница: теперь строки отбираются с использованием специальной функции **Text.Contains**, которая проверяет наличие заданной подстроки в данных, как мы и хотели изначально.

Вот такой неочевидный нюанс. Не попадитесь.

Транспонирование

В математике *транспонированием* называют такое преобразование матрицы (таблицы), при котором строки и столбцы в ней меняются местами:

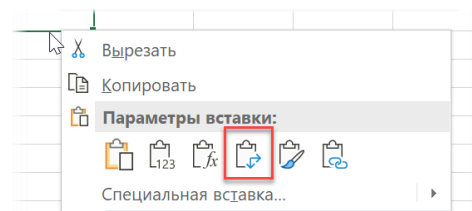
	Лиссабон	Дели	Джакарта	Ханой	Багдад
Елисей	67,59	42,87	22,46	11,2	31,2
Кирилл	6,28	94,54	32,97	76,84	61,48
Артём	77,61	62,78	75,34	27,21	21,66
Ксения	31,91	89,06	47,71	0,21	1,08
Алексей	92,13	31,79	93,12	11,93	15,83
Софья	16,85	94,41	63,65	18,9	57,98
Максим	45,08	14,5	82,79	41,99	90,91



	Елисей	Кирилл	Артём	Ксения	Алексей	Софья	Максим
Лиссабон	67,59	6,28	77,61	31,91	92,13	16,85	45,08
Дели	42,87	94,54	62,78	89,06	31,79	94,41	14,5
Джакарта	22,46	32,97	75,34	47,71	93,12	63,65	82,79
Ханой	11,2	76,84	27,21	0,21	11,93	18,9	41,99
Багдад	31,2	61,48	21,66	1,08	15,83	57,98	90,91

При повседневной работе в Microsoft Excel весьма часто возникает подобная потребность. Решить её можно несколькими способами¹.

Обычно используют копирование и потом специальную вставку в режиме **Транспонировать (Transpose)**, но такой подход не создаст связи между исходным и транспонированным диапазонами. Так что при изменении входных данных всю процедуру придется повторить заново.



Можно ещё воспользоваться функцией **ТРАНСП (TRANSPOSE)**, которая пересчитывается «на лету», но у неё есть свои минусы. Во-первых, её надо вводить как формулу массива сочетанием клавиш **Ctrl+Shift+Enter**, что очень печально влияет на общую скорость пересчета книги. Во-вторых, эта функция не умеет отслеживать изменения размеров исходного диапазона и работает строго в заданных пределах.

Все эти сложности можно обойти, если выполнить транспонирование с помощью запроса в Power Query.

Сначала, конечно же, превратим нашу исходную таблицу в «умную» (например, сочетанием клавиш **Ctrl+T**) или в именованный диапазон и загрузим её в редактор с помощью кнопки **Из таблицы/диапазона** на вкладке **Данные (Data → From Table/Range)**:

	Менеджер	1.2 Лиссабон	1.2 Дели	1.2 Джакарта	1.2 Ханой	1.2 Багдад
1	Елисей	67,59	42,87	22,46	11,2	31,2
2	Кирилл	6,28	94,54	32,97	76,84	61,48
3	Артём	77,61	62,78	75,34	27,21	21,66
4	Ксения	31,91	89,06	47,71	0,21	1,08
5	Алексей	92,13	31,79	93,12	11,93	15,83
6	Софья	16,85	94,41	63,65	18,9	57,98
7	Максим	45,08	14,5	82,79	41,99	90,91

Затем обычно возникает желание сразу же воспользоваться кнопкой **Транспонировать** на вкладке **Преобразование (Transform → Transpose)**, но это приведет к не совсем желательному побочному эффекту – пропадут названия городов:

	Column1	Column2	Column3	Column4	Column5	Column6	Column7
1	Елисей	Кирилл	Артём	Ксения	Алексей	Софья	Максим
2	67,59	6,28	77,61	31,91	92,13	16,85	45,08
3	42,87	94,54	62,78	89,06	31,79	94,41	14,5
4	22,46	32,97	75,34	47,71	93,12	63,65	82,79
5	11,2	76,84	27,21	0,21	11,93	18,9	41,99
6	31,2	61,48	21,66	1,08	15,83	57,98	90,91

Чтобы этого не произошло, нужно сначала опустить города из шапки в первую строку, воспользовавшись кнопкой **Использовать заголовки как первую строку (Use Headers As First Row)** на вкладке **Главная (Home)**, а потом уже транспонировать, и тогда города останутся на месте:

¹ См. несколько способов решения этой задачи на <https://www.planetaexcel.ru/techniques/2/87/>.

	ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5	ABC 123 Column6	ABC 123 Column7
1	Менеджер	Елисей	Кирилл	Артём	Ксения	Алексей	Софья
2	Лиссабон	67,59	6,28	77,61	31,91	92,13	16
3	Дели	42,87	94,54	62,78	89,06	31,79	94
4	Джакарта	22,46	32,97	75,34	47,71	93,12	63
5	Ханой	11,2	76,84	27,21	0,21	11,93	11
6	Багдад	31,2	61,48	21,66	1,08	15,83	57

Останется обратно поднять первую строку с именами менеджеров в заголовки столбцов с помощью кнопки **Использовать первую строку в качестве заголовков** (Use First Row As Headers) и выгрузить готовую таблицу обратно в Excel с помощью команды **Главная → Заккрыть и загрузить** (Home → Close&Load).

Одним из главных преимуществ такого подхода является то, что при любых изменениях в исходной таблице в будущем (добавлении/удалении строк или столбцов, изменении подписей или числовых значений) нам достаточно будет лишь обновить наш запрос правой кнопкой мыши или сочетанием клавиш **Ctrl+Alt+F5**. Ни формулы, ни специальная вставка такого не умеют.

Заполнение пустых ячеек

Нет ничего более сильного и созидательного, чем пустота, которую люди стремятся заполнить.
(Лао-Цзы)

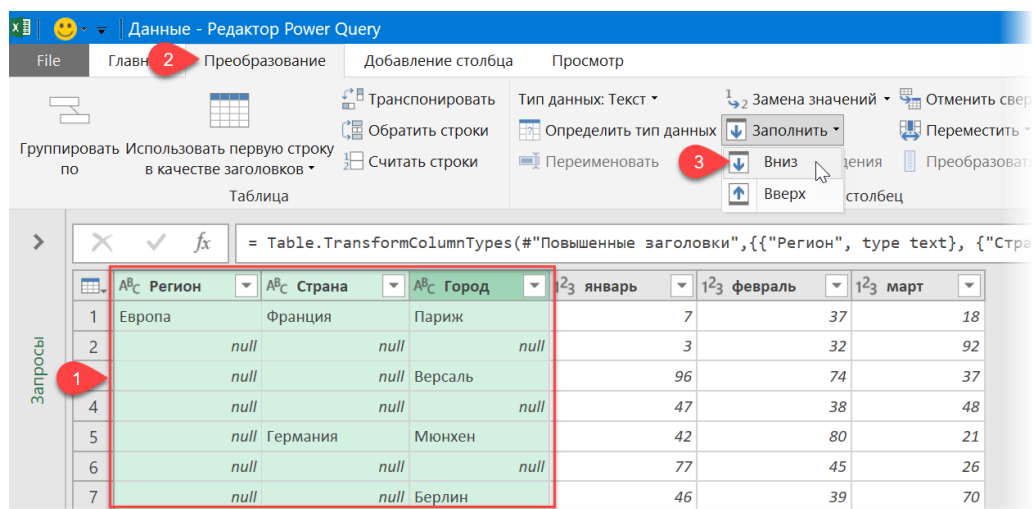
Это очень простой, но приятный инструмент, которого многим, к сожалению, не хватает в самом Excel, поэтому приходится постоянно изобретать какие-то «костыли» в виде формул¹ или макросов². С Power Query всё гораздо проще. Посмотрите, например, на первых три столбца вот такой таблицы:

	A	B	C	D	E	F
1	Регион	Страна	Город	январь	февраль	март
2	Европа	Франция	Париж	7	37	18
3				3	32	92
4			Версаль	96	74	37
5				47	38	48
6		Германия	Мюнхен	42	80	21
7				77	45	26
8			Берлин	46	39	70
9				41	48	12
10				39	96	27
11			Пекин	77	64	83
12		Китай	Шанхай	6	46	69
13				62	14	9
14				35	19	97
15	Азия		Бангкок	16	71	62
16				45	76	9
17		Таиланд		25	53	82
18			Паттайя	22	16	6
19				65	75	79
20						

Знакомая картина, правда?

Такую таблицу невозможно нормально фильтровать или сортировать, т. к. мы будем постоянно спотыкаться о пустые ячейки. Построить по ней сводную тоже проблема. Большинство функций поиска а-ля **ВПР (VLOOKUP)** или вычисления итогов (**СУММЕСЛИ** и т. д.) в ней тоже нормально работать не будут. Причем пустые ячейки могут быть как отдельными (как в регионе *Европа*), так ещё и объединенными (как в *Азии*), что добавляет немало сложностей.

В Power Query эта проблема решается очень легко. После загрузки данных в редактор нужно выделить те столбцы, в которых мы хотим заполнить пустые ячейки, и на вкладке **Преобразование** выбрать команду **Заполнить** → **Заполнить вниз** (Transform → Fill → Fill Down):



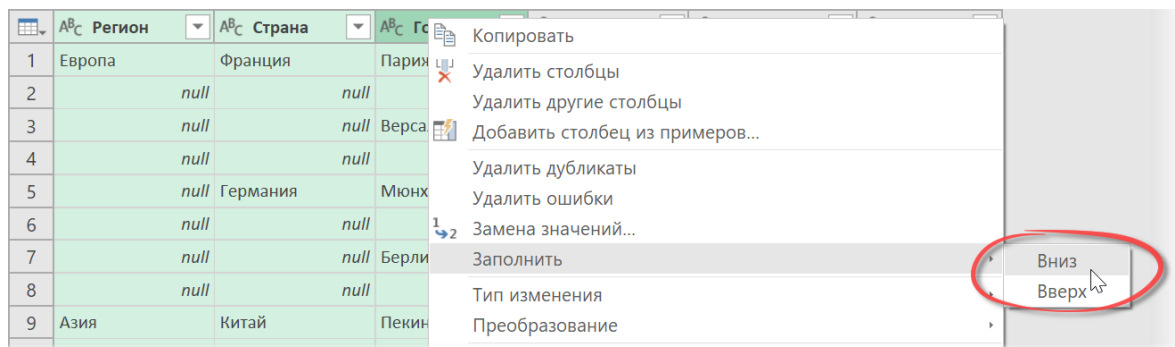
¹ См. статью на моём сайте на эту тему <https://www.planetaexcel.ru/techniques/2/96/>.

² См. макрос из надстройки PLEX для Microsoft Excel <https://www.planetaexcel.ru/plex/features/17/2009/>.

И все пустые (*null*) ячейки будут заполнены вышестоящими значениями:

	АВс Регион	АВс Страна	АВс Город	1 ² 3 январь	1 ² 3 февраль	1 ² 3 март
1	Европа	Франция	Париж	7	37	18
2	Европа	Франция	Париж	3	32	92
3	Европа	Франция	Версаль	96	74	37
4	Европа	Франция	Версаль	47	38	48
5	Европа	Германия	Мюнхен	42	80	21
6	Европа	Германия	Мюнхен	77	45	26
7	Европа	Германия	Берлин	46	39	70
8	Европа	Германия	Берлин	41	48	12
9	Азия	Китай	Пекин	39	96	27
10	Азия	Китай	Пекин	77	64	83

Аналогичная команда есть и в контекстном меню, если щелкнуть правой кнопкой мыши по заголовкам выделенных столбцов:



Признаюсь, что ни разу ещё не сталкивался с ситуацией, когда нужно было выполнить заполнение вверх, но такая возможна, конечно, тоже имеется.

Если же вам вдруг потребуется заполнение вправо или влево, то таблицу можно сначала просто транспонировать, как мы это делали в предыдущей главе.

Группировка строк

Если вы не новичок в Excel, то наверняка знакомы с инструментом **Промежуточные итоги** с вкладки **Данные** (Data → Subtotals). Эта функция позволяет быстро «схлопнуть» строки в группы по значениям заданного столбца и вычислить для них любую нужную функцию подитогов:

А	В	С	Д	Е
Категория	Наименование	Дата сделки	Город	Стоимость
Газир.напитки	Fanta	08.04.2013	Москва	14 900
Газир.напитки	Sprite	05.09.2013	Челябинск	63 100
Газир.напитки	Coca-Cola	30.08.2013	Челябинск	53 200
Соки	Добрый	02.07.2013	Санкт-Петербург	33 100
Соки	Caprice	25.04.2013	Самара	79 600
Вода и чай	Valser	25.10.2013	Екатеринбург	20 400
Соки	Моя семья	29.08.2013	Новосибирск	18 900
Вода и чай	BonAqua	14.07.2013	Казань	15 700
Соки	Добрый	23.05.2013	Омск	65 200
Соки	Добрый	22.04.2013	Нижний Новгород	15 900
Вода и чай	BonAqua	10.04.2013	Ростов-на-Дону	17 900
Соки	Rich	09.09.2013	Нижний Новгород	84 400
Соки	Да!	07.03.2013	Казань	44 600
Газир.напитки	Fanta	01.06.2013	Казань	84 000
Газир.напитки	Coca-Cola	06.10.2013	Нижний Новгород	87 800
Газир.напитки	Coca-Cola Light	29.12.2013	Самара	64 400
Газир.напитки	Фруктайм	03.03.2013	Москва	97 700
Газир.напитки	Fanta	30.12.2013	Омск	57 700
Газир.напитки	Schwennes	25.08.2013	Екатеринбург	60 300
Вода и чай Итого				345 100
Газир.напитки Итого				2 811 800
Соки Итого				1 995 900
Энергетики	Burn	07.08.2013	Челябинск	69 400
Энергетики	Burn	16.10.2013	Санкт-Петербург	19 300
Энергетики	Gladiator	12.10.2013	Омск	72 800
Энергетики	Burn	03.03.2013	Омск	65 000
Энергетики	Gladiator	26.08.2013	Омск	62 800
Энергетики	Powerade	20.02.2013	Казань	11 400
Энергетики	Powerade	01.10.2013	Омск	52 900
Энергетики	Gladiator	23.12.2013	Ростов-на-Дону	64 900
Энергетики	Gladiator	04.10.2013	Санкт-Петербург	99 400
Энергетики Итого				517 900
Общий итог				5 670 700

По сути, этот инструмент просто группирует строки (появляются плюсики слева) и вставляет в таблицу после каждой группы новую строку с функцией **ПРОМЕЖУТОЧНЫЕ.ИТОГИ (SUBTOTALS)**, которая и выполняет роль агрегатора, т. е. суммирует, подсчитывает среднее арифметическое, количество и т. д.

Бонусом можно добавить вставку разрыва при печати (Page Break) между группами, размещение итогов не снизу, а сверху групп и т. д. Простой и приятный инструмент, своего рода «младший брат» сводных таблиц.

Минусы у **Промежуточных итогов** тоже есть.

- Промежуточные итоги нельзя включить в «умных» таблицах
- Перед вставкой итогов таблицу сначала обязательно нужно отсортировать по столбцу, который будет использоваться как главный признак для группировки (в приведенном выше примере это столбец **Категория**), иначе получится каша из данных и итогов.
- При копировании результатов в свернутом виде приходится выделять их с помощью функции **Главная → Найти и выделить → Выделение группы ячеек → Только видимые ячейки (Home → Find & Select → Go to Special → Visible cells only)**, иначе потом вставляются все ячейки, включая и скрытые.
- Ну, и главная стратегическая проблема, на мой взгляд, состоит в том, что при таком подходе нарушается один из базовых принципов обработки данных: исходные данные смешиваются в одной таблице с результатами их же анализа. Мухи и котлеты должны быть отдельно (как в сводных таблицах, например).

Все эти моменты легко обойти, если вычислять промежуточные итоги в Excel через Power Query, где аналогичный инструмент называется **Группировкой (Group By)**.

Простая группировка

А	В	С	Д	Е	Ф	Г
Категория	Наименование	Дата сделки	Город	Стоимость		
Газир.напитки	Fanta	08.04.2013	Москва	14 900		
Газир.напитки	Sprite	05.09.2013	Челябинск	63 100		
Газир.напитки	Coca-Cola	30.08.2013	Челябинск	53 200		
Соки	Добрый	02.07.2013	Санкт-Петербург	33 100		
Соки	Caprice	25.04.2013	Самара	79 600		
Вода и чай	Valser	25.10.2013	Екатеринбург	20 400		
Соки	Моя семья	29.08.2013	Новосибирск	18 900		
Вода и чай	BonAqua	14.07.2013	Казань	15 700		
Соки	Добрый	23.05.2013	Омск	65 200		
Соки	Добрый	22.04.2013	Нижний Новгород	15 900		
Вода и чай	BonAqua	10.04.2013	Ростов-на-Дону	17 900		
Соки	Rich	09.09.2013	Нижний Новгород	84 400		
Соки	Да!	07.03.2013	Казань	44 600		

Предположим, что у нас есть большая таблица (первая на предыдущей иллюстрации) со списком сделок по продаже различных напитков по разным городам. Для загрузки её в Power Query превратим таблицу сначала в «умную» сочетанием клавиш **Ctrl+T** или через команду **Главная → Форматировать как таблицу (Home → Format as Table)**.

Затем загрузим таблицу в Power Query с помощью кнопки **Из таблицы / диапазона** с вкладки **Данные** (Data → From Table / Range).

	АВС Категория	АВС Наименование	Дата сделки	АВС Город	123 Стоимость
1	Газир.напитки	Fanta	08.04.2013	Москва	14900
2	Газир.напитки	Sprite	05.09.2013	Челябинск	63100
3	Газир.напитки	Coca-Cola	30.08.2013	Челябинск	53200
4	Соки	Добрый	02.07.2013	Санкт-Петербург	33100
5	Соки	Caprice	25.04.2013	Самара	79600
6	Вода и чай	Valser	25.10.2013	Екатеринбург	20400

Затем выделим столбец **Категория** и на вкладке **Преобразование** нажмем кнопку **Группировать по** (Transform → Group By):

В открывшемся окне нужно выбрать из выпадающего списка столбец, по которому мы подводим итоги (в нашем случае это **Категория**), ввести имя нового столбца и задать какую математическую операцию (**Сумма**) к какому столбцу (**Стоимость**) мы хотим применить после группировки. После нажатия на **ОК** получится миниатюрная таблица с итогами по каждой категории товаров:

	АВС Категория	1.2 Итоговая стоимость
1	Газир.напитки	2811800
2	Соки	1995900
3	Вода и чай	345100
4	Энергетики	517900

Сложная группировка

Если хочется чего-то более интересного, то можно нажать на значок шестерёнки справа от получившегося шага **Сгруппированные строки** (Grouped Rows) и, вернувшись в предыдущее окно, переключиться в более мощный вариант этого инструмента, выбрав в верхней части окна **Подробнее** (Advanced) вместо **Базовый** (Basic). Тогда мы сможем добавлять сразу несколько критериев группировки (**Категория-Наименование**) и создавать на выходе сразу несколько столбцов с разными видами итогов (сумма, среднее, количество и т. д.) по разным колонкам, например:

Группировать по

Базовый Подробнее

Укажите столбцы для группировки и желаемые выходные данные.

Группировка

Категория

Наименование

Добавление группирования

Имя нового столбца Операция Столбец

Итоговая стоимость Сумма Стоимость

Средний чек Среднее Стоимость

Число сделок Считать строки

Добавление агрегирования

Тогда после нажатия на **OK** нас ждёт другая, более подробная картина (я для наглядности отсортировал таблицу по столбцу **Категория** дополнительно):

	АВС Категория	АВС Наименование	1.2 Итоговая стоимость	1.2 Средний чек	1.2 Число сделок
1	Вода и чай	BonAqua	121400	30350	4
2	Вода и чай	Valser	142900	35725	4
3	Вода и чай	Nestea	80800	80800	1
4	Газир.напитки	Фруктайм	505000	56111,11111	9
5	Газир.напитки	Coca-Cola	807600	57685,71429	14
6	Газир.напитки	Fanta	311100	51850	6
7	Газир.напитки	Квас	424500	70750	6
8	Газир.напитки	Coca-Cola Light	283100	56620	5
9	Газир.напитки	Schweppes	213000	71000	3
10	Газир.напитки	Sprite	267500	44583,33333	6
11	Соки	Моя семья	176500	35300	5
12	Соки	Дал	539700	67462,5	8
13	Соки	Добрый	558900	62100	9
14	Соки	Rich	380500	76100	5
15	Соки	Caprice	340300	68060	5
16	Энергетики	Powerade	64300	32150	2
17	Энергетики	Gladiator	299900	74975	4
18	Энергетики	Burn	153700	51233,33333	3

Управлять уровнями группировки и добавленными итогами можно с помощью значка с многоточием, который появляется при наведении на правую часть любой строки. Он позволяет удалить ненужную группировку или сдвинуть её вверх или вниз при необходимости.

Группировка

Категория

Наименование

Добавление группирования

Удалить

Вверх

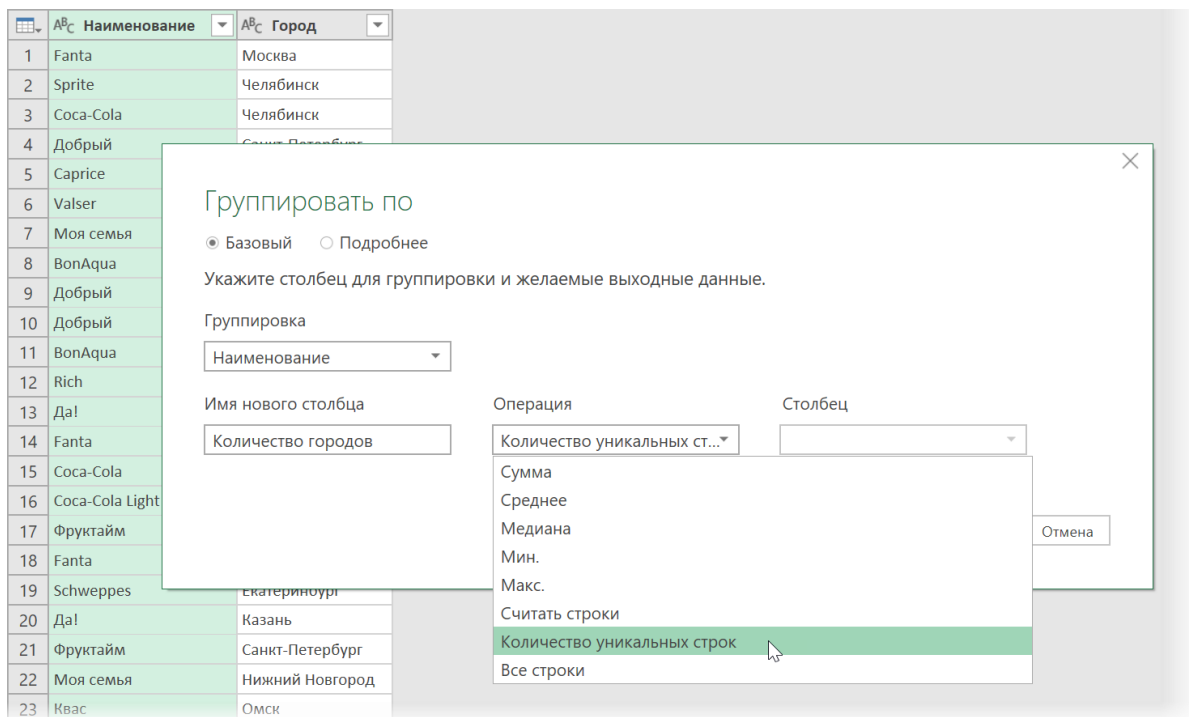
Вниз

Подсчет количества уникальных значений

Предположим, что мы хотим подсчитать количество городов, в которых продается тот или иной товар. Алгоритм действий в такой ситуации будет следующий:

1. Создадим новый запрос, загрузив еще раз предыдущую таблицу в Power Query кнопкой **Из таблицы/диапазона (From Table / Range)**.

- Удалим из данных все столбцы, кроме **Наименования** и **Города**, выделив их и щелкнув правой кнопкой по их заголовку → **Удалить другие столбцы** (Remove Other Columns).
- Выделим столбец **Наименование** и нажмем кнопку **Группировать по** на вкладке **Преобразование** (Transform → Group By). В открывшемся окне выберем **Наименование** как поле для группировки и функцию **Количество уникальных строк** (Count Distinct Rows) для расчета итога:



Для наглядности можно выбрать положение селектора **Подробнее** (Advanced) и добавить для сравнения ещё общее количество сделок по каждому товару:

Имя нового столбца	Операция	Столбец
Количество городов	Количество уникальных ст...	
Количество сделок	Считать строки	

- После нажатия на **ОК** мы получим желаемое – вычисленное количество уникальных (т. е. неповторяющихся) названий городов по каждому товару и общее количество сделок:

№	Наименование	1.2 Количество городов	1.2 Количество сделок
1	Fanta	4	6
2	Sprite	6	6
3	Coca-Cola	8	14
4	Добрый	6	9
5	Caprice	4	5
6	Valser	4	4

То есть «Фанту», например, купили шесть раз в четырех различных городах. Вот только вопрос, в каких именно? А вот для этого нам сначала надо разобраться со следующим пунктом.

Группировка с выводом всех значений

Одной из примечательных возможностей инструмента группировки в Power Query, по сравнению с его аналогом в Excel, является возможность вообще не применять агрегирующих функций типа суммы, среднего, количества и т. д. к сгруппированным данным, а просто выводить их «как есть» в виде вложенных таблиц.

Допустим, что в приведённом ранее примере нам нужно отобразить (не просуммировать, а показать по отдельности!) стоимости всех сделок по каждому товару.

Загрузим нашу таблицу в Power Query и нажмем кнопку **Группировать по** на вкладке **Преобразование** (Transform → Group By). В открывшемся окне выберем **Наименование** как критерий группировки, введём любое имя нового столбца и – главное! – выберем опцию **Все строки** (All rows):

Категория	Наименование	Дата сделки	Город	Стоимость
Газир.напитки	Fanta	08.04.2013	Москва	14900
Газир.напитки	Sprite	05.09.2013	Челябинск	63100
Газир.напитки	Coca-Cola	30.08.2013	Челябинск	53200
Соки				
Соки				
Вода и чай				
Соки				
Соки				
Вода и чай				
Соки				
Соки				
Газир.напитки				
Газир.напитки				
Газир.напитки				
Газир.напитки				
Газир.напитки				
Газир.напитки	Schweppes	25.08.2013	Екатеринбург	60300

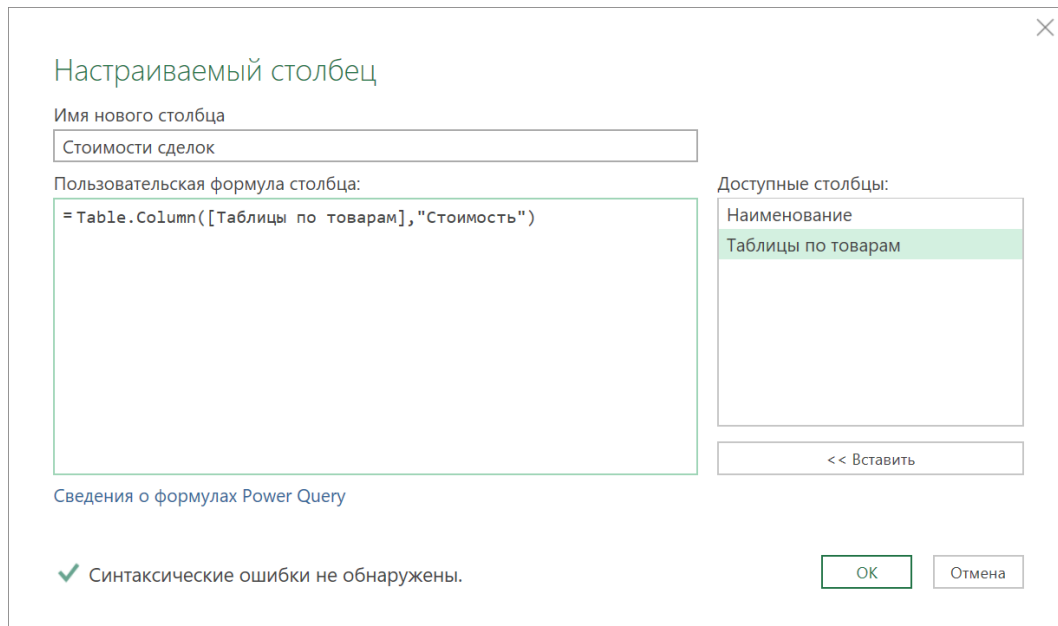
После нажатия на **ОК** мы увидим таблицу со списком товаров и вложенными таблицами, содержащими стоимости сделок по каждому из них. Увидеть содержимое этих таблиц можно в нижней части окна, если щелкнуть мышью в белый фон ячейки рядом со словом **Table** (но не в слово **Table!**):

Категория	Наименование	Дата сделки	Город	Стоимость
Газир.напитки	Sprite	05.09.2013	Челябинск	63100
Газир.напитки	Sprite	30.12.2013	Санкт-Петербург	38200
Газир.напитки	Sprite	20.09.2013	Омск	81700
Газир.напитки	Sprite	30.12.2013	Екатеринбург	38900
Газир.напитки	Sprite	09.08.2013	Самара	20000
Газир.напитки	Sprite	27.09.2013	Москва	25600

Теперь нам нужно извлечь из этих таблиц только столбец **Стоимость**. Перейдем на вкладку **Добавление столбца** и нажмем на кнопку **Настраиваемый столбец** (Add Column → Custom Column). В открывшемся окне введём любое имя столбца и формулу (с соблюдением регистра):

```
=Table.Column([Таблицы по товарам], "Стоимость")
```

где первый аргумент – это имя таблицы, из которой мы выдергиваем колонку (у нас вложенные таблицы лежат в **Таблицы по товарам**), а второй – имя столбца, который мы хотим извлечь.



После нажатия на **ОК** к нашей таблице должен добавиться столбец со списками (List), где каждый список представляет собой, по сути, соответствующий столбец исходной таблицы:

	AB C Наименование	Таблицы по товарам	ABC 123 Стоимости сделок
1	Fanta	Table	List
2	Sprite	Table	List
3	Coca-Cola	Table	List
4	Добрый	Table	List
5	Caprice	Table	List
6	Valser	Table	List
7	Моя семья	Table	List
8	BonAqua	Table	List

List
63100
38200
81700
38900
20000
25600

Чтобы развернуть содержимое списков, щёлкнем по значку с двойными стрелками в шапке столбца **Стоимости сделок** и выберем команду **Извлечь значения** (Extract Values). После выбора символа-разделителя (я использовал знак плюс) мы наконец увидим то, что хотелось, – отдельные суммы сделок по каждому товару:

	AB C Наименование	AB C Стоимости сделок
1	Fanta	14900 + 84000 + 57700 + 45700 + 61200 + 47600
2	Sprite	63100 + 38200 + 81700 + 38900 + 20000 + 25600
3	Coca-Cola	53200 + 87800 + 52000 + 47800 + 40400 + 53500 + 47900 + 96100 + 43200 + 76000 + 58800 + 22100 + 46100 + 82700
4	Добрый	33100 + 65200 + 15900 + 77900 + 72600 + 95000 + 94100 + 49100 + 56000
5	Caprice	79600 + 47900 + 72300 + 84800 + 55700
6	Valser	20400 + 36500 + 12100 + 73900
7	Моя семья	18900 + 19900 + 78300 + 10800 + 48600
8	BonAqua	15700 + 17900 + 38200 + 49600
9	Rich	84400 + 93500 + 17400 + 88100 + 97100
10	Дал	44600 + 38700 + 75000 + 75900 + 89000 + 100000 + 93600 + 22900
11	Coca-Cola Light	64400 + 31300 + 94000 + 38000 + 55400
12	Фруктайм	97700 + 71300 + 57600 + 17200 + 78200 + 15900 + 41900 + 82900 + 42300
13	Schweppes	60300 + 90700 + 62000
14	Квас	97100 + 44800 + 91100 + 97000 + 24500 + 70000
15	Nestea	80800
16	Burn	69400 + 19300 + 65000
17	Gladiator	72800 + 62800 + 64900 + 99400
18	Powerade	11400 + 52900

Извлечение уникальных значений при группировке

Положим, что ассортимент продаваемых в каждом городе товаров не одинаков и мы хотим вывести список наименований для каждого города в нашей таблице.

Первым делом, загрузив данные в Power Query, удалим все лишние столбцы, кроме **Наименования** и **Города**. Затем сгруппируем строки **Городам**, выбрав в качестве операции **Все строки (All rows)**:

На выходе получится уже знакомая нам картина: рядом с каждым городом образуется вложенная таблица со всеми сделками по данному городу. Проблема только в том, что наименования товаров в этой таблице не уникальны, т. к. один и тот же товар запросто может продаваться в этом городе несколько раз:

№	Город	Таблицы по городам
1	Москва	Table
2	Челябинск	Table
3	Санкт-Петербург	Table
4	Самара	Table
5	Екатеринбург	Table
6	Новосибирск	Table
7	Казань	Table
8	Омск	Table
9	Нижний Новгород	Table
10	Ростов-на-Дону	Table

Наименование	Город
Fanta	Москва
Фруктайм	Москва
Фруктайм	Москва
Coca-Cola	Москва
Sprite	Москва

В предыдущей главе мы использовали функцию **Table.Column** языка M, чтобы извлечь содержимое заданного столбца в виде списка (List). Теперь же придется обернуть её ещё одной функцией – **List.Distinct**, которая убирает в списке повторы, оставляя только уникальные значения.

Так что жмём ещё раз **Добавление столбца** → **Настраиваемый столбец (Add Column → Custom Column)** и в открывшемся окне вводим нашу конструкцию из двух вложенных друг в друга функций:

```
=List.Distinct(Table.Column([Таблицы по городам], "Наименование"))
```

После нажатия на **OK** мы увидим столбец со списками, где повторов в наименованиях уже нет:

Город	Таблицы по городам	Уникальные товары
1 Москва	Table	List
2 Челябинск	Table	List
3 Санкт-Петербург	Table	List
4 Самара	Table	List
5 Екатеринбург	Table	List
6 Новосибирск	Table	List
7 Казань	Table	List
8 Омск	Table	List
9 Нижний Новгород	Table	List
10 Ростов-на-Дону	Table	List

List
Fanta
Фруктайм
Coca-Cola
Sprite

Останется, как мы уже делали ранее, развернуть списки через, например, запятую, щёлкнув по кнопке с двойными стрелками в шапке столбца **Уникальные товары** и выбрав опцию **Извлечь значения (Extract Values)** – и задача решена:

Город	Уникальные товары
1 Москва	Fanta, Фруктайм, Coca-Cola, Sprite
2 Челябинск	Sprite, Coca-Cola, Burn, Квас, Да!, Фруктайм, Caprice
3 Санкт-Петербург	Добрый, Фруктайм, Sprite, BonAqua, Burn, Gladiator
4 Самара	Caprice, Coca-Cola Light, Фруктайм, Fanta, Да!, Coca-Cola, Sprite, Rich, Моя семья
5 Екатеринбург	Valser, Schweppes, Да!, Coca-Cola, Caprice, Добрый, Sprite, Coca-Cola Light
6 Новосибирск	Моя семья, Coca-Cola Light, Coca-Cola, Rich, Добрый
7 Казань	BonAqua, Да!, Fanta, Добрый, Coca-Cola, Powerade, Coca-Cola Light, Фруктайм, Valser, Schweppes
8 Омск	Добрый, Fanta, Квас, Nestea, Caprice, Gladiator, Burn, Sprite, Powerade, Coca-Cola, BonAqua
9 Нижний Новгород	Добрый, Rich, Coca-Cola, Моя семья, Квас, Фруктайм, Valser
10 Ростов-на-Дону	BonAqua, Schweppes, Квас, Valser, Моя семья, Gladiator

И для полноты картины стоит добавить, что совершенно аналогичный результат можно было бы достичь чуть быстрее, если использовать вместо нашей формулы другую, но посложнее:

```
=Text.Combine(Table.Distinct([Таблицы по городам],"Наименование")[Наименование],", ")
```

Давайте разберем её по частям.

1. Функция **Table.Distinct([Таблицы по городам],"Наименование")** удаляет во вложенных таблицах повторы по столбцу **Наименование**.
2. Добавление к ней на конце имени столбца в квадратных скобках: **Table.Distinct([Таблицы по городам],"Наименование")[Наименование]** выделяет из таблицы только столбец **Наименование**, который нам и нужен.
3. И наконец, функция **Text.Combine** склеивает содержимое этого столбца через заданный символ-разделитель (запятая с пробелом).

Как обычно, всегда есть несколько способов решения любой задачи. Выбор всегда хорошо!

Первый/последний элемент в каждой группе

Ещё одним весьма частым сценарием является поиск крайних элементов (строк) в каждой группе – последней сделки по каждому клиенту/товару по данным из CRM, последнего платежа от каждого контрагента в банковской выписке, первой продажи заданного товара и т. п.

Предположим, что в качестве входных данных мы имеем вот такую таблицу со сделками по товарам:

	A	B	C	I
1	Товар	Сумма	Дата	
2	Яблоко	6904	10.01.2018	
3	Лосось	2280	06.07.2018	
4	Лосось	4850	11.12.2018	
5	Просо	7921	29.11.2018	
6	Брокколи	5742	21.05.2018	
7	Просо	3052	15.04.2018	
8	Лосось	3170	24.04.2018	
9	Земляника	3549	27.12.2018	
10	Земляника	9593	15.11.2018	
11	Просо	1796	22.11.2018	
12	Просо	8552	07.07.2018	
13	Брокколи	5061	25.07.2018	
14	Земляника	4579	13.11.2018	
15	Земляника	766	07.01.2018	
16	Брокколи	2359	22.01.2018	

Загрузим эту таблицу в Power Query, а затем сделаем следующие действия.

1. Отсортируем данные по возрастанию даты от старых к новым, используя кнопку фильтра в шапке столбца **Дата**.
2. Выполним группировку, выбрав команду **Преобразование → Группировать по** (Transform → Group by) и настроив затем в диалоговом окне следующие параметры:

The screenshot shows the Power Query interface with a table of transactions. The 'Group By' dialog box is open, showing the following configuration:

- Группировка:** Товар
- Имя нового столбца:**
 - Количество сделок
 - Подробности
- Операция:**
 - Считать строки
 - Все строки
- Столбец:** (empty)

После нажатия на **ОК** мы получим таблицу, состоящую из трех столбцов:

- уникальные названия всех товаров;
- количество сделок (строк) по каждому товару;
- вложенные таблицы (Table) с подробностями по всем сделкам для каждого товара.

Товар	Сумма	Дата
Брокколи	2359	22.01.2018
Брокколи	4415	27.01.2018
Брокколи	5742	21.05.2018
Брокколи	5061	25.07.2018
Брокколи	3829	28.09.2018
Брокколи	9126	16.10.2018
Брокколи	7105	28.11.2018

Осталось извлечь из каждой такой вложенной таблицы содержимое первой и/или последней строки, чтобы получить данные по самой ранней или поздней сделке. Это можно легко сделать с помощью добавления к нашей таблице вычисляемого столбца с формулой на языке M.

Нажмём на вкладке **Добавление столбца** кнопку **Настраиваемый столбец** (Add Column → Custom Column) и введём туда формулу:

Настраиваемый столбец

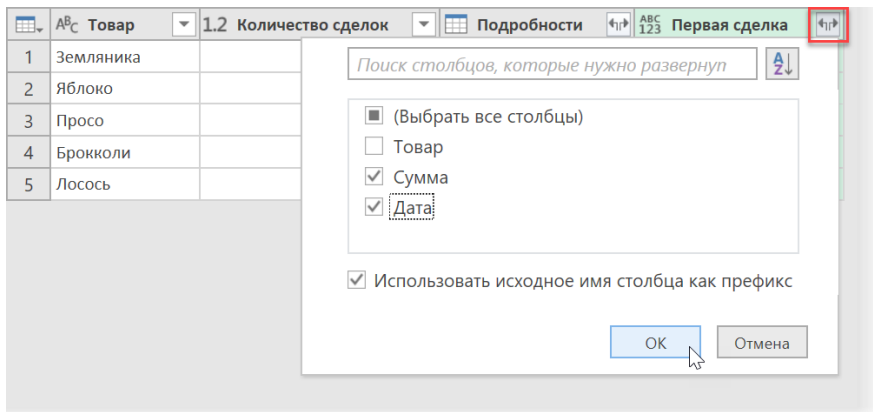
Имя нового столбца

Пользовательская формула столбца:

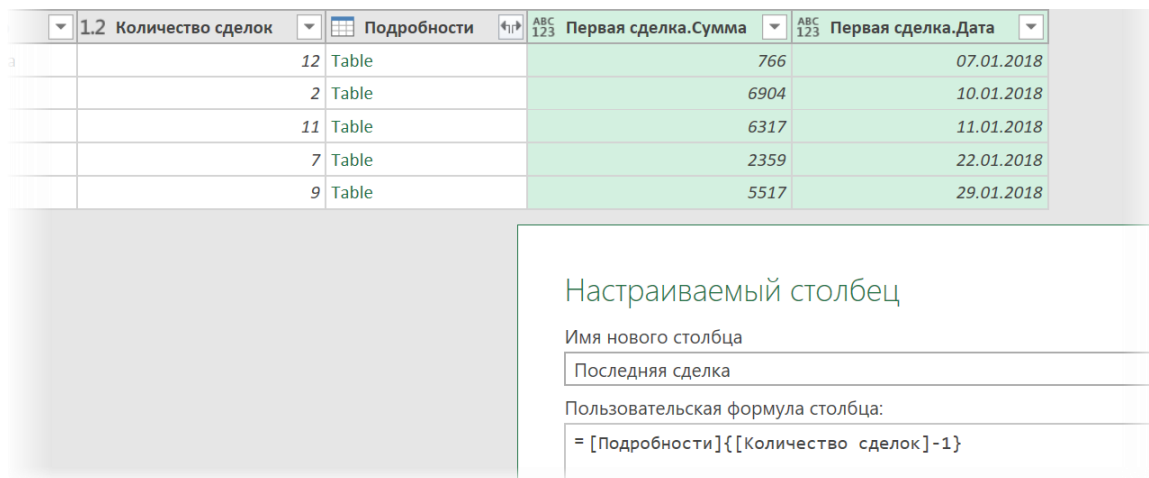
Технически это означает, что мы хотим взять первую строку из каждой вложенной таблицы столбца **Подробности** (нумерация строк в Power Query начинается с нуля). После нажатия на **ОК** мы получим столбец с **записями** (Records) – первыми строками из вложенных таблиц по каждому товару:

Товар	Сумма	Дата
Брокколи	2359	22.01.2018

Как и списки (Lists) ранее, записи (Records) можно развернуть в новые столбцы, используя значок с двойными стрелками в шапке столбца и выбрав затем колонки, которые хотим получить:



Для получения дат и сумм по последним сделкам можно поступить аналогично, но выдергивать данные не из первой (нулевой), а из последней строки. Для этого придется задействовать в формуле значение из столбца **Количество сделок**, убавив его на 1 (нумерация строк с нуля, помните?):



После разворачивания полученных записей кнопкой с двойными стрелками и последующего удаления лишних столбцов мы, наконец, получим желаемый результат:

	Товар	Первая сделка.Сумма	Первая сделка.Дата	Последняя сделка.Сумма	Последняя сделка.Дата
1	Земляника	766	07.01.2018	3549	27.12.2018
2	Яблоко	6904	10.01.2018	3994	03.06.2018
3	Просо	6317	11.01.2018	7921	29.11.2018
4	Брокколи	2359	22.01.2018	7105	28.11.2018
5	Лосось	5517	29.01.2018	9588	16.12.2018

В качестве альтернативного способа, позволяющего сэкономить пару лишних шагов, можно использовать и другие формулы. Для получения суммы первой сделки по каждому товару подойдет конструкция:

```
=List.First(Table.Column([Подробности], "Сумма"))
```

А для получения, например, даты последней сделки:

```
=List.Last(Table.Column([Подробности], "Дата"))
```

Как легко сообразить, здесь вложенная функция **Table.Column** выдергивает из вложенных таблиц колонки Подробности нужный столбец и превращает его в список, а затем функции **List.First** и **List.Last** извлекают первый или соответственно последний элементы этого списка. Тогда столбец с количеством сделок по каждому товару становится уже не нужен, как и необходимость разворачивать вложенные записи.

Свёртывание таблиц

Под свёртыванием (pivoting) в Power Query понимается особый тип трансформации таблиц, при котором **уникальные значения из заданного столбца превращаются в заголовки новых столбцов**. Визуально это выглядит как сворачивание из одномерной таблицы в двумерную и чем-то напоминает построение классической сводной таблицы, только без итогов:

Город	Товар	Стоимость
Новосибирск	Апельсины	23
Екатеринбург	Киви	785
Новосибирск	Яблоки	457
Санкт-Петербург	Киви	3
Челябинск	Апельсины	367
Нижний Новгород	Киви	210
Новосибирск	Яблоки	64
Санкт-Петербург	Яблоки	790
Санкт-Петербург	Яблоки	436
Новосибирск	Киви	961
Казань	Яблоки	179
Новосибирск	Киви	586
Челябинск	Киви	872
Екатеринбург	Яблоки	432
Челябинск	Апельсины	687
Нижний Новгород	Яблоки	551
Екатеринбург	Апельсины	759
Новосибирск	Апельсины	675
Новосибирск	Апельсины	490
Екатеринбург	Киви	793
Москва	Апельсины	334
Челябинск	Апельсины	298
Нижний Новгород	Яблоки	832
Новосибирск	Киви	387
Нижний Новгород	Яблоки	935
Новосибирск	Киви	596
Казань	Яблоки	884
Казань	Апельсины	639
Санкт-Петербург	Апельсины	205

Классическая сводная таблица

Сумма по полю Ст	Названия с	Апельсины	Киви	Яблоки	Общий итог	
Екатеринбург		759		1578	432	2769
Казань		639			1063	1702
Москва		334				334
Нижний Новгород			210		2318	2528
Новосибирск		1188	2530		521	4239
Санкт-Петербург		205	3		1226	1434
Челябинск		1352	872			2224
Общий итог		4477	5193	5560		15230

Результат свёртывания запросом Power Query

Город	Апельсин	Киви	Яблоки
Екатеринбург	759	1578	432
Казань	639		1063
Москва	334		
Нижний Новгород		210	2318
Новосибирск	1188	2530	521
Санкт-Петербург	205	3	1226
Челябинск	1352	872	

Тут же возникает логичный вопрос: «Если это так похоже на обычную сводную таблицу в Excel, то зачем это делать в Power Query?» Причин тут может быть несколько.

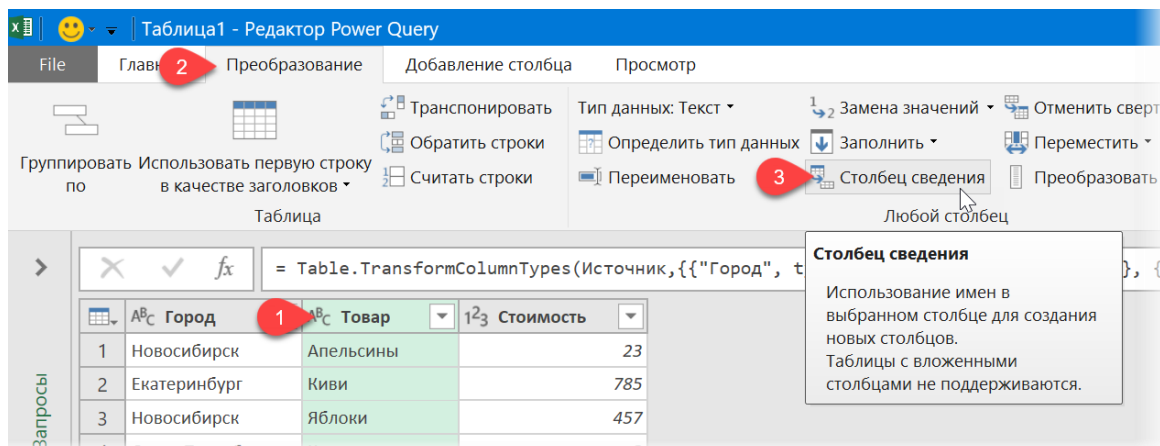
- Таблица, получаемая в результате свёртывания, может быть только одним из **промежуточных этапов** в процессе обработки данных, а не финальной картинкой. Соответственно, в Power Query мы можем продолжать с ней работать в запросе, а со сводной особо не поработаешь, она уже готовый продукт.
- На выходе запроса Power Query мы получаем стандартно «умную» таблицу, с которой во многих случаях **удобнее дальше работать**, чем со сводной. В сводную нельзя вносить изменения, легко добавлять столбцы (если только это не вычисляемые поля), нельзя построить по сводной некоторые типы диаграмм и т. д. «Умные» таблицы всем этим не страдают.
- Классическая сводная таблица умеет считать ограниченный набор функций в области значений (сумму, количество, среднее, минимум, максимум и т. д.). При сворачивании таблицы в Power Query **набор возможных функций гораздо шире**, к нему добавляются: медиана, количество уникальных элементов, возможность вывести текст и т. д.

В общем и целом свёртывание в Power Query является, конечно, не заменой, но удачной альтернативой сводным таблицам в большом количестве случаев. Давайте разберёмся с тем, как всё это работает.

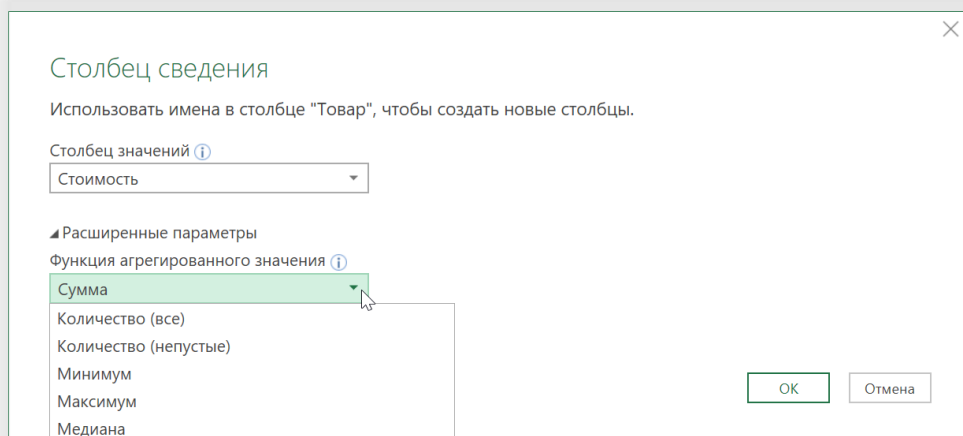
Простое свёртывание

Начнём с простого примера, приведённого на предыдущей иллюстрации. Допустим, что у нас есть таблица со столбцами **Город**, **Товар**, **Стоимость**. Чтобы получить из неё свернутую по товарам и городам таблицу, выполним следующие действия.

1. Загрузим исходную таблицу в Power Query через **Данные → Из таблицы/диапазона** (Data → From Table/Range).
2. Выделим столбец **Товар**, значения которого должны уйти в шапку, т. е. превратиться в заголовки новых столбцов.
3. На вкладке **Преобразование** нажмем кнопку **Столбец сведения** (Transform → Pivot Column):



4. Должно появиться диалоговое окно, где можно выбрать столбец, значения которого пойдут на пересечение строк и столбцов (область значений, если проводить аналогию со сводной) и функцию расчета:



И после нажатия на **ОК** мы увидим желаемый результат:

	Город	Апельсины	Киви	Яблоки
1	Екатеринбург	759	1578	432
2	Казань	639	null	1063
3	Москва	334	null	null
4	Нижний Новгород	null	210	2318
5	Новосибирск	1188	2530	521
6	Санкт-Петербург	205	3	1226
7	Челябинск	1352	872	null

На первый взгляд всё не очень сложно, правда? Но что будет, если в исходной таблице больше трёх столбцов? Тогда при сворачивании действует следующий принцип:

- тот столбец, который выделен, уходит заголовками в новые колонки свернутой таблицы (т. е. область столбцов в сводной таблице);

- столбец, выбранный в качестве столбца значений в диалоговом окне, идёт в середину (область значений сводной таблицы);
- все остальные столбцы попадают в область строк.

Таким образом, если к нашим исходным данным добавится ещё один столбец, например, с названием региона:

	A	B	C	D	E
1	Регион	Город	Товар	Стоимость	
2	Восток	Новосибирск	Апельсины	23	
3	Восток	Екатеринбург	Киви	785	
4	Восток	Новосибирск	Яблоки	457	
5	Север	Санкт-Петербург	Киви	3	
6	Восток	Челябинск	Апельсины	367	
7	Центр	Нижний Новгород	Киви	210	
8	Восток	Новосибирск	Яблоки	64	

то после выполнения всех ранее описанных действий мы получим уже другую таблицу, где строки схлопнулись по связке **Регион-Город**, а товары ушли, как и раньше, в столбцы:

	A ^В Регион	B ^В Город	1 ² Апельсины	1 ² Киви	1 ² Яблоки
1	Восток	Екатеринбург	759	1578	432
2	Восток	Новосибирск	1188	2530	521
3	Восток	Челябинск	1352	872	null
4	Север	Санкт-Петербург	205	3	1226
5	Центр	Казань	639	null	1063
6	Центр	Москва	334	null	null
7	Центр	Нижний Новгород	null	210	2318

Имитация сводной с текстом в значениях

Одним из важных преимуществ свёртывания в Power Query по сравнению с классической сводной таблицей является возможность помещать в область значений не числа, а текст, т. е. делать что-то очень похожее на сводную, но с текстом в значениях (**Values**).

Допустим, наша компания возит в несколько городов России и Казахстана свою продукцию в контейнерах. Контейнеры могут отправляться один или несколько раз в месяц. Каждый контейнер имеет буквенно-цифровой номер-код. В качестве исходных данных имеется стандартная таблица с перечислением поставок, из которой нужно сделать некое подобие сводной, чтобы наглядно видеть номера контейнеров, отправленных в каждый город и каждом месяце:

Страна	Город	Месяц	Контейнер
Россия	Москва	янв	3GQ-794
Россия	Москва	янв	CPK-333
Россия	Москва	май	D7H-481
Россия	Москва	май	SOL-304
Россия	Москва	мар	ONB-121
Россия	Москва	мар	ABC-777
Россия	Самара	янв	AEV-111
Россия	Самара	янв	YTY-323
Россия	Самара	апр	QZI-330
Россия	Москва	июн	HIW-465
Россия	Самара	фев	70U-145
Россия	Питер	апр	XBI-514
Россия	Питер	июн	N48-604
Россия	Питер	май	5GE-150
Казахстан	Алматы	май	7Y0-614
Казахстан	Алматы	янв	L6Y-960
Казахстан	Алматы	фев	BS8-273
Казахстан	Алматы	мар	13J-691



Страна	Город	янв	фев	мар	апр	май	июн
Казахстан	Алматы	L6Y-960	BS8-273	13J-691		7Y0-614	
Казахстан	Астана						LCH-184
Казахстан	Павлодар	H8Q-523		1ME-839	NHE-646	OEC-235 YIN-872 LSE-090	
Россия	Москва	3GQ-794 CPK-333		ONB-121 ABC-777		D7H-481 SOL-304	HIW-465
Россия	Питер				XBI-514	5GE-150	N48-604
Россия	Самара	AEV-111 YTY-323	70U-145		QZI-330		

Сначала, конечно же, превратим нашу исходную таблицу в «умную» и загрузим в Power Query:

	АВС Страна	АВС Город	АВС Месяц	АВС Контейнер
1	Россия	Москва	янв	3GQ-794
2	Россия	Москва	янв	CPK-333
3	Россия	Москва	май	D7H-481
4	Россия	Москва	май	SOL-304
5	Россия	Москва	май	VSL-920
6	Россия	Москва	мар	OHB-121

Для наглядности я предварительно отсортировал таблицу по странам-городам-месяцам через **Данные → Сортировка (Data → Sort)**, но по факту делать это совершенно необязательно.

Чтобы собрать все заказы по каждому городу и месяцу в единое целое, сначала сгруппируем наши данные по связке сразу трёх параметров: страны, города и месяца, используя уже знакомую нам команду **Группировать по** с вкладки **Преобразование (Transform Group by)**. В качестве операции выберем **Все строки (All rows)**:

Группировать по

Базовый Подробнее

Укажите столбцы для группировки и желаемые выходные данные.

Группировка

Страна

Город

Месяц

Добавление группирования

Имя нового столбца Операция Столбец

Поставки Все строки

Добавление агрегирования

OK Отмена

На выходе после группировки мы получим дополнительный столбец, где в каждой ячейке будет лежать вложенная таблица со всеми поставками:

	АВС Страна	АВС Город	АВС Месяц	Поставки
1	Россия	Москва	янв	Table
2	Россия	Москва	май	Table
3	Россия	Москва	мар	Table
4	Россия	Самара	янв	Table
5	Россия	Самара	апр	Table
6	Россия	Москва	июн	Table
7	Россия	Самара	фев	Table
8	Россия	Питер	апр	Table
9	Россия	Питер	июн	Table

Страна	Город	Месяц	Контейнер
Россия	Москва	май	D7H-481
Россия	Москва	май	SOL-304

Чтобы извлечь из вложенных таблиц только номера контейнеров, добавим ещё один вычисляемый столбец через **Добавление столбца** → **Настраиваемый столбец** (Add Column → Custom Column) и введём туда формулу, которую мы уже использовали в главе [Группировка с выводом всех значений](#):

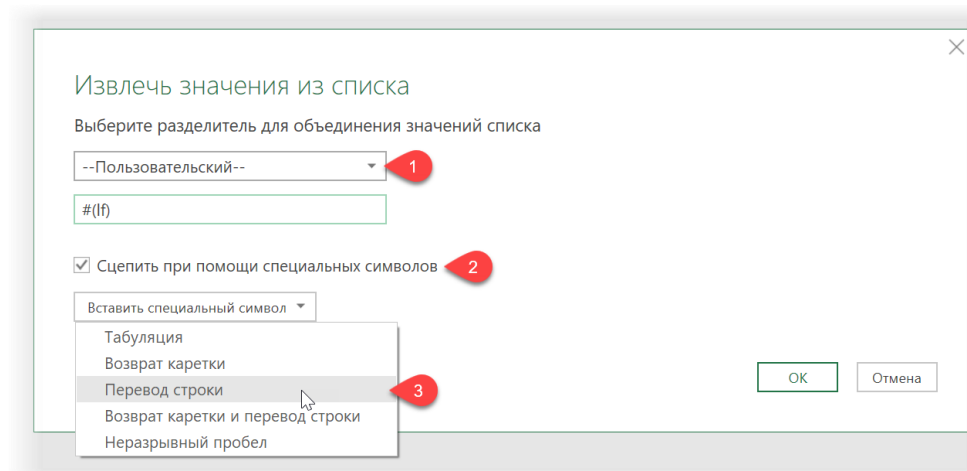
```
=Table.Column([Поставки], "Контейнер")
```

У нас получится новый столбец со списками (Lists) – извлеченными из последнего столбца каждой таблицы номерами контейнеров:

	Страна	Город	Месяц	Поставки	Контейнеры
1	Россия	Москва	янв	Table	List
2	Россия	Москва	май	Table	List
3	Россия	Москва	мар	Table	List
4	Россия	Самара	янв	Table	List
5	Россия	Самара	апр	Table	List
6	Россия	Москва	июн	Table	List
7	Россия	Самара	фев	Table	List
8	Россия	Питер	апр	Table	List
9	Россия	Питер	июн	Table	List

List
D7H-481
SOL-304

Теперь развернём списки с помощью кнопки с двойными стрелками в шапке таблицы, выбрав опцию **Извлечь значения** (Extract Values). В качестве разделителя можно использовать любой подходящий символ (запятую, пробел, точку с запятой и т. д.) по желанию. Если хочется, чтобы каждый номер контейнера был в отдельной строке, то нужно включить флажок **Сцепить при помощи специальных символов** (Concatenate using special characters) и выбрать затем **Перевод строки** (Line feed):



После нажатия на **OK** и удаления ненужного больше столбца **Поставки** с вложенными таблицами мы получим очень похожий на то, что нам нужно, результат:

	Страна	Город	Месяц	Контейнеры
1	Россия	Москва	янв	3GQ-794 CPK-333
2	Россия	Москва	май	D7H-481 SOL-304
3	Россия	Москва	мар	ONB-121 ABC-777
4	Россия	Самара	янв	AEV-111 YTY-323
5	Россия	Самара	апр	QZI-330
6	Россия	Москва	июн	HIW-465
7	Россия	Самара	фев	70U-145
8	Россия	Питер	апр	XBI-514

Осталось выделить столбец **Месяц** и произвести по нему свёртывание кнопкой **Столбец сведения (Pivot Column)** на вкладке **Преобразование (Transform)**. В открывшемся диалоговом окне выберем в качестве столбца значений колонку с номерами контейнеров, а в расширенных параметрах – операцию **Не агрегировать (Don't aggregate)**.

И вот наша «сводная» с текстом в значениях:

	АВС Страна	АВС Город	АВС янв	АВС май	АВС мар	АВС апр	АВС июн	АВС фев
1	Казахстан	Алматы	L6Y-960	7Y0-614	13J-691	null	null	BS8-273
2	Казахстан	Астана	null	null	null	null	LCH-184	null
3	Казахстан	Павлодар	H8Q-523	0EC-235 YIN-872 LSE-090	1ME-839	NHE-646	null	null
4	Россия	Москва	3GQ-794 CPK-333	D7H-481 SOL-304	OHV-121 ABC-777	null	HIW-465	null
5	Россия	Питер	null	5GE-150	null	XBI-514	N48-604	null
6	Россия	Самара	AEV-111 YTY-323	null	null	QZI-330	null	70U-145

Останется отсортировать её в любом желаемом порядке (по городам или странам) и расположить столбцы в нужной последовательности, перетаскивая их за заголовок. После этого можно выгрузить то, что получилось, на лист, навести красоту и наслаждаться результатом:

	А Страна	В Город	С янв	Д фев	Е мар	Ф апр	Г май	Н июн
1	Казахстан	Алматы	L6Y-960	BS8-273	13J-691		7Y0-614	
2	Казахстан	Астана					LCH-184	
3	Казахстан	Павлодар	H8Q-523		1ME-839	NHE-646	0EC-235 YIN-872 LSE-090	
4	Россия	Москва	3GQ-794 CPK-333		OHV-121 ABC-777		D7H-481 SOL-304	HIW-465
5	Россия	Питер				XBI-514	5GE-150	N48-604
6	Россия	Самара	AEV-111 YTY-323	70U-145		QZI-330		
7								
8								

Трансформация столбца в двумерную таблицу

Весьма полезным практическим применением инструмента **Столбец сведения (Pivot Column)** из предыдущей главы является задача трансформации одномерного столбца с данными в двумерную таблицу.

Представьте, что после выгрузки из какой-либо корпоративной ERP-системы, базы данных или программы мы получили на выходе информацию в виде длинного столбца, куда попали все значения друг за другом. Разумеется, для работы нам нужно трансформировать этот одномерный массив в нормальную таблицу, разложив содержимое по нескольким столбцам и строкам соответственно.

Прежде чем выполнять такую свёртку с помощью рассмотренного нами в прошлой главе инструмента **Столбец сведения (Pivot Column)**, потребуется сделать некоторые подготовительные манипуляции. Их сложность зависит от того, какого вида столбец нам достался – с одинаковым или разным шагом в данных.

Постоянный шаг в данных

Если в исходном столбце прослеживается чёткая зависимость и данные повторяются с регулярной периодичностью (например, каждые 7 ячеек, как на картинке ниже), то нам повезло: эта задача решается достаточно легко:

	A	B	C	D	E	F	G	H	I	J
1	10.04.2017									
2	Максим									
3	Коломна									
4	UPS									
5	9991988									
6	10%									
7										
8	13.04.2017									
9	Кристина									
10	Новокузнецк									
11	Alibaba Group									
12	3319133									
13	13%									
14										
15	15.07.2017									
16	Богдан									
17	Новороссийск									
18	Heineken									
19	8957205									
20	8%									
21										
22	16.02.2017									
23	Мария									
24	Воронеж									
25	Nike									

Дата	Менеджер	Город	Компания	Сумма	Скидка
10.04.2017	Максим	Коломна	UPS	9 991 988	10%
13.04.2017	Кристина	Новокузнецк	Alibaba Group	3 319 133	13%
15.07.2017	Богдан	Новороссийск	Heineken	8 957 205	8%
16.02.2017	Мария	Воронеж	Nike	9 559 469	2%
20.09.2017	Злата	Чита	SoftBank	2 078 239	46%

Сначала загрузим наш столбец в Power Query любым подходящим способом. Если будете делать это через «умную» таблицу, то не забудьте снять галочку **Таблица с заголовками (My table has headers)**, т. к. шапки у нашей таблицы в этом примере нет.

После загрузки обычным образом через **Данные → Из таблицы/диапазона (Data → From table/range)** делаем следующие действия:

1. Добавляем к нашей таблице столбец с номерами строк через вкладку **Добавление столбца → Столбец индекса → От 0 (Add Column → Index Column → From 0)**.
2. К созданному столбцу индекса применяем команду **Добавление столбца → Стандартный → Разделить (целое число) (Add Column → Standard → Divide (integer))** и в открывшемся окне вводим шаг повторения наших данных – число 7. Получим ещё одну колонку, где будет результат целочисленного деления индекса на 7.
3. Затем, выделив столбец индекса, делим все числа в нём на тот же шаг 7 и берём остаток от полученного результата деления. Это легко сделать командой **Преобразование → Стандартный → Остаток от деления (Transform → Standard → Modulo)**.

Если все описанные выше действия вы проделали правильно, то у вас должна получиться следующая картина:

	ABC 123 Столбец1	1.2 Индекс	1 ² 3 Целочисленное деление
1	10.04.2017 0:00:00	0	0
2	Максим	1	0
3	Коломна	2	0
4	UPS	3	0
5	9991988	4	0
6	0,1	5	0
7	null	6	0
8	13.04.2017 0:00:00	0	1
9	Кристина	1	1
10	Новокузнецк	2	1
11	Alibaba Group	3	1
12	3319133	4	1
13	0,13	5	1
14	null	6	1
15	15.07.2017 0:00:00	0	2
16	Богдан	1	2
17	Новороссийск	2	2
18	Heineken	3	2
19	8957205	4	2
20	0,08	5	2
21	null	6	2
22	16.02.2017 0:00:00	0	3
23	Мария	1	3
24	Воронеж	2	3
25	Nike	3	3
26	9559469	4	3

Числа в столбце **Целочисленное деление** – это, по сути, уникальный признак каждого набора данных, т. е. номер строки в будущей таблице. Числа же из колонки **Индекс** – это фактически готовые номера наших будущих столбцов.

Останется только выполнить уже знакомую нам операцию сворачивания по столбцу **Индекс**. Для этого выделим колонку **Индекс** и выберем **Преобразование** → **Столбец сведения** (Transform → Pivot Column). В открывшемся окне в качестве столбца значений задаём колонку с исходными данными **Столбец1**, а в расширенных параметрах выбираем функцию **Не агрегировать** (Don't Aggregate):

	ABC 123 Столбец1	1.2 Индекс	1 ² 3 Целочисленное деление
1	10.04.2017 0:00:00	1	0
2	Максим	2	0
3	Коломна	3	0
4	UPS	4	0
5		5	0
6		6	0
7		7	0
8	13.04.2017 0:00:00	0	1
9	Кристина	1	1
10	Новокузнецк	2	1
11	Alibaba Group	3	1
12		4	1
13		5	1
14		6	1
15	15.07.2017 0:00:00	0	2
16	Богдан	1	2
17	Новороссийск	2	2
18	Heineken	3	2
19	8957205	4	2

Столбец сведения

Использовать имена в столбце "Индекс", чтобы создать новые столбцы.

Столбец значений ⓘ

▲ Расширенные параметры
 Функция агрегированного значения ⓘ

Дополнительные сведения о столбце сведения

После нажатия на **ОК** таблица свернётся к нужному нам двумерному виду, разместив каждый блок из 7 ячеек построчно:

	Целочисленное деление	ABC 123 0	ABC 123 1	ABC 123 2	ABC 123 3	ABC 123 4	ABC 123 5	ABC 123 6	
1		0	10.04.2017 0:00:00	Максим	Коломна	UPS	9991988	0,1	null
2		1	13.04.2017 0:00:00	Кристина	Новокузнецк	Alibaba Group	3319133	0,13	null
3		2	15.07.2017 0:00:00	Богдан	Новороссийск	Heineken	8957205	0,08	null
4		3	16.02.2017 0:00:00	Мария	Воронеж	Nike	9559469	0,02	null
5		4	20.09.2017 0:00:00	Злата	Чита	SoftBank	2078239	0,46	null

Останется только удалить лишние столбцы и переименовать оставшиеся.

Изячно, не правда ли?

На самом деле все можно сделать ещё быстрее, буквально одной командой, если не побояться заглянуть «под капот» и использовать прямое программирование действий с помощью М-кода, а не ограничиваться только стандартными инструментами с ленты интерфейса.

1. Загрузите исходную таблицу-столбец в Power Query как подключение. Назовите его, допустим, **Исходник**.
2. Создайте новый пустой запрос через **Данные → Получить данные → Из других источников → Пустой запрос (Data → Get Data → From Other Sources → Blank Query)**.
3. Введите в строку формул следующую конструкцию:

```
=Table.FromRows(List.Split(Исходник[Столбец1],7))
```

и нажмите на **Enter** – на экране появится почти то, что нам нужно!

	Column1	Column2	Column3	Column4	Column5	Column6	Column7
1	10.04.2017 0:00:00	Максим	Коломна	UPS	9991988	0,1	null
2	13.04.2017 0:00:00	Кристина	Новокузнецк	Alibaba Group	3319133	0,13	null
3	15.07.2017 0:00:00	Богдан	Новороссийск	Heineken	8957205	0,08	null
4	16.02.2017 0:00:00	Мария	Воронеж	Nike	9559469	0,02	null
5	20.09.2017 0:00:00	Злата	Чита	SoftBank	2078239	0,46	null

Останется только удалить ненужный последний столбец и дать колонкам имена – и всё! Магия!

Как это работает?

Всё предельно просто.

Функция **List.Split** берёт **Столбец1** из нашей таблицы и разбивает его на блоки-списки по 7 ячеек в каждом. Затем функция **Table.FromRows** формирует таблицу, используя нарезанные блоки по 7 ячеек как строки.

Проникнитесь важной мыслью: зачастую то, что делается в Power Query стандартными инструментами за множество шагов, на самом деле можно сделать одной-двумя строками кода на языке М. Power Query – мощная штука, но эта мощь кратно умножается, если вы хоть немного знаете, как писать М-код. Поэтому в конце этой книги мы посвятим несколько глав подробному разбору синтаксиса, основных программных конструкций и самых полезных функций этого языка.

И ещё одно: практически любой запрос можно улучшить, сделать короче, быстрее, изящнее. Обработка (особенно нестандартная) данных в Power Query – это всегда немного импровизация и вызов. Найдя один способ решения проблемы, никогда не останавливайтесь. Скорее всего, найдется более короткая и быстрая цепочка шагов-действий, приводящих вас к тому же результату.

Переменный шаг в данных

Настоящая тренировка начинается, когда вы хотя бы на шаг превышаете свой прошлый результат.

(Из книги «Бизнес как игра»)

Теперь давайте ещё усложним задачу и предположим, что исходные данные не имеют чёткой повторяющейся структуры. Например, после каждого имени менеджера может идти любое произвольное количество его сделок. Нам же нужно преобразовать эту последовательность опять в двумерную таблицу:

	A	B	C	D	E	F	G	H	I	J
1	Иван									
2		30 500								
3		92 200								
4		85 300								
5	Сергей									
6		15 800								
7		63 500								
8	Елена				Иван	Сергей	Елена	Олег	Никола	Даниил
9		96 300			30500	15800	96300	56700	42800	75300
10		88 800			92200	63500	88800	53800	28100	43600
11		82 700			85300		82700		33900	55000
12		55 300					55300			90600
13		68 100					68100			18100
14	Олег									1000
15		56 700								
16		53 800								
17	Николай									
18		42 800								
19		28 100								
20		33 900								
21	Даниил									
22		75 300								
23		43 600								
24		55 000								
25		90 600								
26		18 100								
27		1 000								
28										

Как вы понимаете, если шаг повторения данных в исходном столбце не постоянный, то описанные в предыдущем пункте подходы не работают, нам нужна другая тактика.

Сначала загружаем исходные данные в Power Query и добавляем к ним столбец индекса через **Добавление столбца → Столбец индекса → От 1** (Add Column → Index Column → From 1), как мы уже делали ранее. Должно получиться следующее:

	ABC 123	Столбец1	1.2	Индекс
1		Иван		1
2		30500		2
3		92200		3
4		85300		4
5		Сергей		5
6		15800		6
7		63500		7
8		Елена		8

Теперь нам нужно понять, по какому принципу группируются исходные данные и по какому признаку можно понять, что начался новый блок. В приведенном выше примере можно использовать различие между типами данных: каждый блок начинается с текста (имени менеджера), за которым следуют числа (суммы его продаж).

Добавим новый вычисляемый столбец через **Добавление столбца** → **Настраиваемый столбец** (Add Column → Custom Column) и в открывшемся окне введём его имя (например, **Столбец**) и формулу:

```
=if Value.Is([Столбец1], type text) then [Индекс] else null
```

Здесь встроенная М-функция **Value.FromText** проверяет, является ли содержимое **Столбца1** текстом. Если это текст, то выводится порядковый номер строки из колонки **Индекс**, в противном случае оставляем ячейку пустой (**null**). На выходе должно получиться так:

1.2 Индекс	Столбец1	Столбец
1	Иван	1
2	30500	null
3	92200	null
4	85300	null
5	Сергей	5
6	15800	null
7	63500	null
8	Елена	8
9		
10		
11		
12		
13		
14	Олег	
15		
16		
17	Николай	

Настраиваемый столбец

Имя нового столбца:

Пользовательская формула столбца: `=if Value.Is([Столбец1], type text) then [Индекс] else null`

Доступные столбцы: Столбец1 Индекс

Дальше – проще. Выбираем созданный столбец **Столбец** и заполняем в нём пустые (null) ячейки предыдущими значениями, используя команду **Заполнить вниз** с вкладки **Преобразование** (Transform → Fill → Fill Down).

Добавляем ещё один вычисляемый столбец (назовём его, например, **Строка**), где вычислим разность между значениями в столбцах **Индекс** и **Столбец** формулой:

```
=[Индекс] - [Столбец]
```

Ненужный более столбец **Индекс** можно удалить.

Столбец1	Столбец	Строка
Иван	1	0
30500	1	1
92200	1	2
85300	1	3
Сергей	5	0
15800	5	1
63500	5	2
Елена	8	0
96300	8	1
88800	8	2
82700	8	3
55300	8	4
68100	8	5
Олег	14	0
56700	14	1
53800	14	2
Николай	17	0
42800	17	1

Как вы уже, наверное, догадались, таблица готова к свёртке. В колонке **Столбец** сейчас содержится номер столбца для свёртывания, а в колонке **Строка** – порядковый номер каждой строки будущей таблицы.

Выполняем сворачивание по колонке **Столбец** командой **Преобразование → Столбец сведения (Transform → Pivot Column)**, не забыв в расширенных параметрах выбрать **Не агрегировать (Don't Aggregate)**:

Столбец	Столбец	Столбец	Столбец	Столбец	Столбец	Столбец	Столбец
1	0	Иван	Сергей	Елена	Олег	Николай	Даниил
2	1	30500	15800	96300	56700	42800	75300
3	2	92200	63500	88800	53800	28100	43600
4	3	85300	null	82700	null	33900	55000
5	4	null	null	55300	null	null	90600
6	5	null	null	68100	null	null	18100
7	6	null	null	null	null	null	1000

Останется удалить ненужный первый столбец и поднять имена менеджеров из первой строки в шапку кнопкой **Использовать первую строку в качестве заголовков (Use First Row as Headers)** с вкладки **Главная (Home)** – и наша задача решена!

Вдогон хочется отметить, что в ваших данных вполне может быть другой принцип деления на блоки в отличие от рассмотренного примера с типами данных «текст-число». Тогда формула для проверки на начало нового блока, которую мы использовали:

```
=if Value.Is([Столбец1], type text) then [Индекс] else null
```

будет иметь другой вид.

Вот несколько жизненных примеров, которые помогут вам сориентироваться в нужном направлении.

Проверка на число:

```
=if Value.Is([Столбец1], type number) then [Индекс] else null
```

Проверка на дату:

```
=if Value.Is([Столбец1], type datetime) then [Индекс] else null
```

Проверка на пустую ячейку:

```
=if [Столбец1]=null then [Индекс] else null
```

Проверка на то, что текст в столбце начинается с заданной подстроки, например, название компании начинается с "ООО":

```
=if Text.StartsWith([Столбец1],"ООО") then [Индекс] else null
```

Проверка на то, что в конце текста есть заданные символы, например, "руб":

```
=if Text.EndsWith([Столбец1],"руб") then [Индекс] else null
```

Проверка на то, что текст содержит заданную подстроку, например, "ИНН":

```
=if Text.Contains([Столбец1],3),"ИНН") then [Индекс] else null
```

Проверка на то, что текст в ячейке длиннее 5 символов:

```
=if Text.Length([Столбец1])>5 then [Индекс] else null
```

Отмена свёртывания

Обратной процедурой к свертыванию столбцов (Pivot Column) из предыдущей главы является **Отмена свертывания** (Unpivot Column), представленная на вкладке **Преобразование** (Transform) одноименной кнопкой. И если свертывание в некоторых случаях ещё можно заменить сводной таблицей в Excel, то для отмены свертывания альтернативы можно сказать, что нет. Если без Power Query, то реализовать это можно лишь макросом (не самым простым) или формулой (совсем страшной).

Но для начала давайте разберёмся, зачем оно нужно в принципе.

Зачем нужна отмена свёртывания

Представим, что нам досталась двумерная таблица по продажам наших менеджеров по городам вот такого вида:

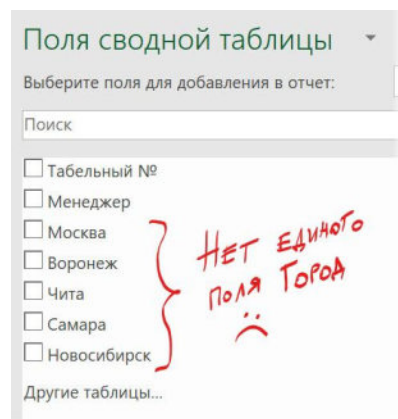
	A	B	C	D	E	F	G
1	Табельный №	Менеджер	Москва	Воронеж	Чита	Самара	Новосибирск
2	857738	Карина	214	594	382	670	544
3	102529	Александр	647	347	653	202	94
4	244665	Савелий	993	26	521	248	444
5	732482	Иван	167	751	676	60	867
6	295098	Яна	256	591	265	648	976
7	722796	Мargarита	493	624	627	123	57
8							

При попытке построить по ней сводную таблицу стандартным образом через **Вставка → Сводная таблица** (Insert Pivot → Table) мы тут же столкнемся с большой проблемой: в списке полей сводной не будет поля **Город**, которое можно было бы поместить в область строк, столбцов или фильтра. Вместо этого там будут присутствовать поля для каждого города по отдельности.

Это невероятно усложняет создание даже простой сводной таблицы и большинство вычислений в ней. Придется закидывать в сводную каждый город по отдельности и для каждого же настраивать формат чисел и функцию расчёта. Посчитать простейшую долю в процентах для каждого города или отличие продаж всех городов от, например, Москвы стандартными средствами тоже уже не получится.

Для решения всех этих проблем нам придется преобразовать нашу исходную двумерную таблицу (такие таблицы иногда ещё называют *кросс-таблицами*, от английского слова *cross = перекрёсток*) в одномерную или, если выражаться в терминах баз данных, *плоскую (flat)*:

Табельный №	Менеджер	Город	Значение
857738	Карина	Москва	214
857738	Карина	Воронеж	594
857738	Карина	Чита	382
857738	Карина	Самара	670
857738	Карина	Новосибирск	544
102529	Александр	Москва	647
102529	Александр	Воронеж	347
102529	Александр	Чита	653
102529	Александр	Самара	202
102529	Александр	Новосибирск	94
244665	Савелий	Москва	993
244665	Савелий	Воронеж	26
244665	Савелий	Чита	521
244665	Савелий	Самара	248
244665	Савелий	Новосибирск	444
732482	Иван	Москва	167
732482	Иван	Воронеж	751
732482	Иван	Чита	676

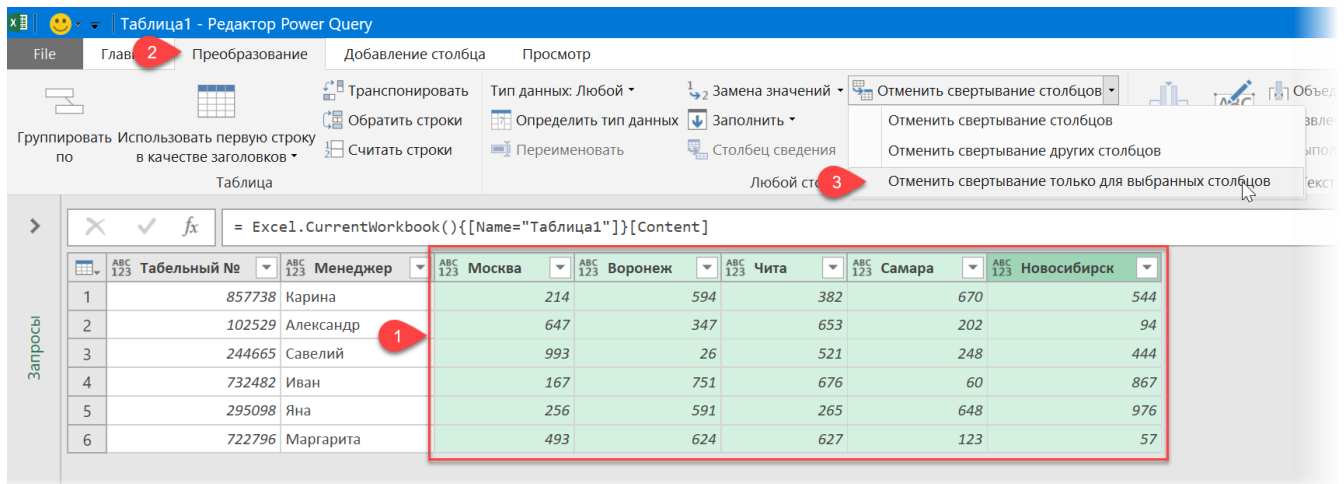


Давайте рассмотрим несколько практических сценариев использования этого крайне полезного инструмента в Power Query.

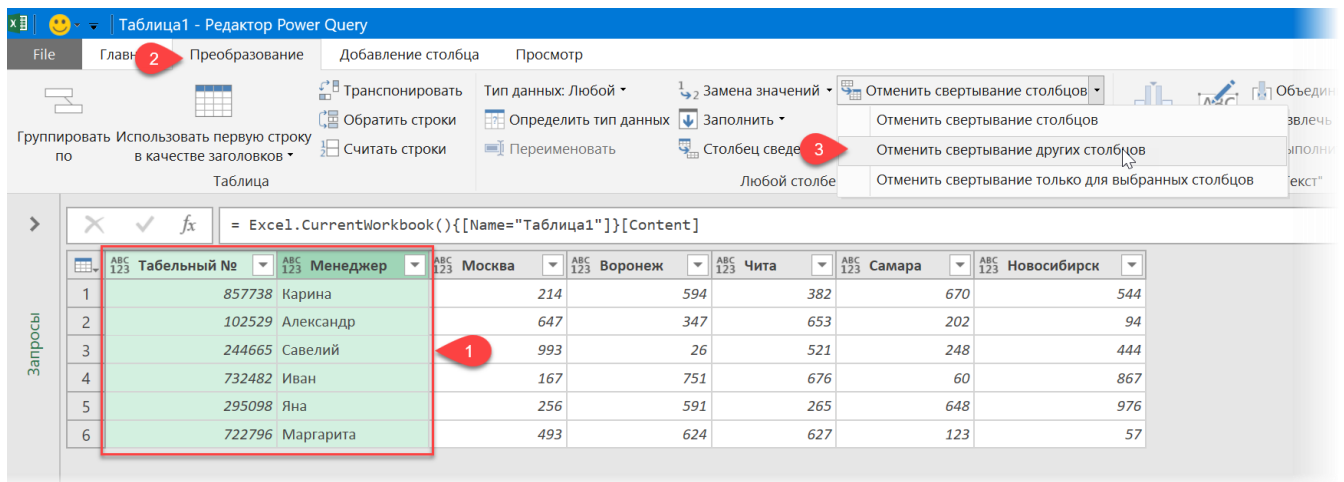
Отмена свёртывания простой таблицы

Начнем с простого случая, описанного на предыдущей странице. Превратим нашу исходную таблицу с продажами по городам и менеджерам в «умную» и загрузим в редактор Power Query. Дальше можно будет пойти двумя путями.

Можно выделить те столбцы, которые нужно превратить в пару **Город-Сумма**, и выбрать на вкладке **Преобразование** команду **Отменить свертывание только для выделенных столбцов** (Transform → Unpivot Only Selected Columns):



Этот вариант, однако, имеет один недостаток: если в будущем к нашей таблице добавятся новые столбцы-города (что весьма вероятно!), то они уже не будут конвертироваться. Поэтому в большинстве случаев лучше «пойти от противного» и выделить те столбцы, которые не должны попадать под развертывание, а затем выбрать соседнюю команду **Отменить свертывание других столбцов** (Unpivot Other Columns):



Power Query тут же выдаст нам желаемый результат:

Табельный №	Менеджер	Атрибут	Значение
1	Карина	Москва	214
2	Карина	Воронеж	594
3	Карина	Чита	382
4	Карина	Самара	670
5	Карина	Новосибирск	544
6	Александр	Москва	647
7	Александр	Воронеж	347
8	Александр	Чита	653
9	Александр	Самара	202
10	Александр	Новосибирск	94
11	Савелий	Москва	993

Останется только переименовать заголовки столбцов из стандартных *Атрибут* (*Attribute*) и *Значение* (*Value*) во что-то более наглядное и выгрузить полученные данные обратно в Excel с помощью кнопки **Заккрыть и загрузить** на вкладке **Главная** (**Home** → **Close & Load**).

Теперь при построении сводной таблицы по полученным данным всё будет гораздо проще и удобнее, т. к. теперь у нас есть отдельное поле **Город**, которое можно помещать в любую нужную нам область сводной таблицы:

Отмена свёртывания таблицы с многоуровневыми подписями

Теперь давайте попробуем победить более сложный случай – таблицу с многоуровневыми подписями строк и столбцов (многоэтажной шапкой), где у каждого числового значения есть не два параметра **Менеджер-Город**, как было в прошлом примере, а существенно больше:

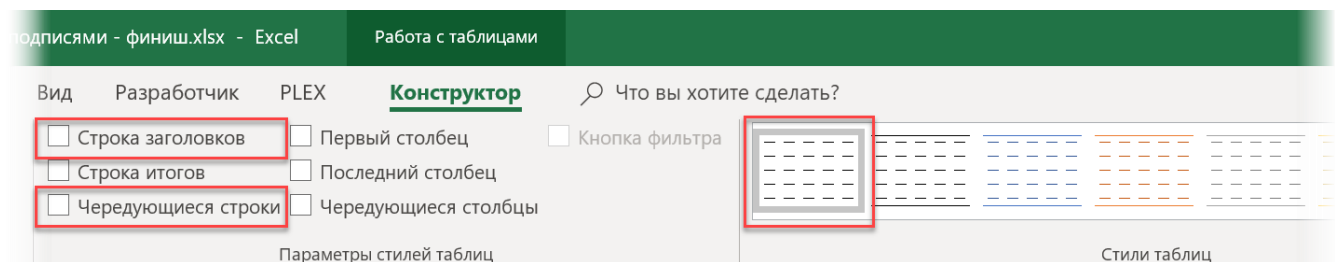
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1				2015								2016							
2				Q1		Q2		Q3		Q4		Q1		Q2		Q3		Q4	
3	Город	Менеджер	Компания	План	Факт	План	Факт	План	Факт	План	Факт	План	Факт	План	Факт	План	Факт	План	Факт
4	Москва	Виктория	Skoda	81	40	92	13	79	24	51	69	50	84	67	83	15	94	74	62
5			Volkswagen	12	59	61	19	45	85	81	51	59	95	99	64	26	22	45	70
6		Владислав	Fiat	50	30	70	51	53	25	99	32	13	98	46	79	53	87	41	53
7			Saab	10	63	56	18	12	27	95	39	95	80	40	94	17	77	63	96
8	СПБ	Артём	Hyundai	87	20	74	33	34	26	83	33	64	50	54	98	41	75	47	37
9			Mercedes	49	56	16	52	73	71	96	89	58	39	85	14	89	94	38	96
10		Татьяна	Mini	48	78	72	53	15	52	15	31	59	11	61	55	59	82	16	64
11			Mitsubishi	76	18	91	36	39	62	98	68	73	41	23	98	52	98	73	26
12			Rover	60	97	35	24	76	37	90	72	36	49	35	42	10	45	53	89
13																			

Для начала нужно загрузить нашего «монстра» в Power Query. Выделим исходные данные и преобразуем их в «умную» таблицу с помощью сочетания клавиш **Ctrl+T** или команды **Форматировать как таблицу** (**Format as Table**) на вкладке **Главная** (**Home**). В открывшемся окне обязательно проверьте, чтобы флажок **Таблица с заголовками** (**My table has headers**) был выключен, т. к. нормальной шапки (в понимании Excel) у нас нет.

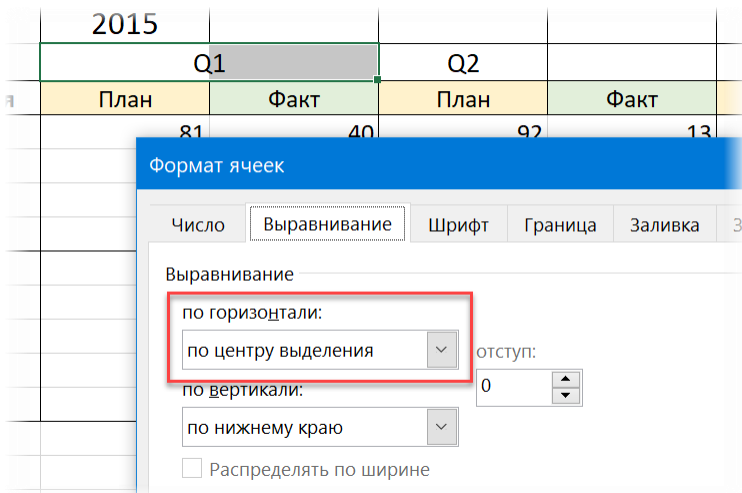
После нажатия на **ОК** Excel сам добавит к таблице шапку с фильтрами, разъединит объединенные ячейки с годами и кварталами и применит к таблице полосатый стиль оформления:

	A	B	C	D	E	F	G	H	I	J	K
1	Столбец1	Столбец2	Столбец3	Столбец4	Столбец5	Столбец6	Столбец7	Столбец8	Столбец9	Столбец10	Столбец11
2				2015							
3				Q1		Q2		Q3		Q4	
4	Город	Менеджер	Компания	План	Факт	План	Факт	План	Факт	План	Факт
5	Москва	Виктория	Skoda	81	40	92	13	79	24	51	
6			Volkswagen	12	59	61	19	45	85	81	
7		Владислав	Fiat	50	30	70	51	53	25	99	
8			Saab	10	63	56	18	12	27	95	
9	СПБ	Артём	Hyundai	87	20	74	33	34	26	83	
10			Mercedes	49	56	16	52	73	71	96	
11		Татьяна	Mini	48	78	72	53	15	52	15	
12			Mitsubishi	76	18	91	36	39	62	98	
13			Rover	60	97	35	24	76	37	90	
14											
15											

Если эти внешние изменения вас напрягают, то от большинства из них можно отказаться на вкладке **Конструктор (Design)**, сняв флажки **Строка заголовка (Headers)**, **Чередующиеся строки (Banded rows)** и выбрав стиль **Нет (None)**:



Исчезнувшее после преобразования в «умную» таблицу объединение ячеек можно симитировать с помощью выравнивания по горизонтали **По центру выделения (Center Across Selection)**:



После восстановления внешней красоты загрузим нашу таблицу в Power Query стандартным образом – с помощью кнопки **Из таблицы / диапазона** на вкладке **Данные (Data From Table/Range)**:

	Column1	Column2	Column3	Column4	Column5	Column6	Column7	Column8	Column9	Column10
1	null	null	null	2015	null	null	null	null	null	null
2	null	null	null	Q1	null	Q2	null	Q3	null	Q4
3	Город	Менеджер	Компания	План	Факт	План	Факт	План	Факт	План
4	Москва	Виктория	Skoda	81	40	92	13	79	24	51
5	null	null	Volkswagen	12	59	61	19	45	85	81
6	null	Владислав	Fiat	50	30	70	51	53	25	99
7	null	null	Saab	10	63	56	18	12	27	95
8	СПБ	Артём	Hyundai	87	20	74	33	34	26	83
9	null	null	Mercedes	49	56	16	52	73	71	96
10	null	Татьяна	Mini	48	78	72	53	15	52	15
11	null	null	Mitsubishi	76	18	91	36	39	62	98
12	null	null	Rover	60	97	35	24	76	37	90

Теперь начинается самое интересное.

1. Сначала выделим первых три столбца и заполним в них пустые ячейки (null) значениями из вышестоящих ячеек, используя кнопку **Заполнить** → **Заполнить вниз** на вкладке **Преобразование** (Transform → Fill → Fill Down):

	Column1	Column2	Column3	Column4
1		null	null	null
2		null	null	Q1
3	Город	Менеджер	Компания	План
4	Москва	Виктория	Skoda	
5	Москва	Виктория	Volkswagen	
6	Москва	Владислав	Fiat	
7	Москва	Владислав	Saab	
8	СПБ	Артём	Hyundai	
9	СПБ	Артём	Mercedes	

2. Затем склеим текст из этих же трёх выделенных столбцов в один через любой символ-разделитель, например точку с запятой, используя команду **Преобразование** → **Объединить столбцы** (Transform → Merge Columns):

	Сведено	Column4	Column5
1	::		2015
2	::	Q1	
3	Город;Менеджер;Компания	План	Факт
4	Москва;Виктория;Skoda		81
5	Москва;Виктория;Volkswagen		12
6	Москва;Владислав;Fiat		50
7	Москва;Владислав;Saab		10
8	СПБ;Артём;Hyundai		87
9	СПБ;Артём;Mercedes		49
10	СПБ;Татьяна;Mini		48

3. Теперь транспонируем нашу таблицу (поменяем местами строки и столбцы) с помощью кнопки **Транспонировать** (Transpose) на вкладке **Преобразование** (Transform):

	Column1	Column2	Column3	Column4	Column5	Column6	Column7
1	::	::	Город;Менеджер...	Москва;Виктория...	Москва;Виктория...	Москва;Владисла...	Моск
2	2015	Q1	План	81	12	50	
3	null	null	Факт	40	59	30	
4	null	Q2	План	92	61	70	
5	null	null	Факт	13	19	51	
6	null	Q3	План	79	45	53	
7	null	null	Факт	24	85	25	
8	null	Q4	План	51	81	99	
9	null	null	Факт	69	51	32	
10	2016	Q1	План	50	59	13	
11	null	null	Факт	84	95	98	
12	null	Q2	План	67	99	46	
13	null	null	Факт	83	64	79	
14	null	Q3	План	15	26	52	

4. Поднимем первую строку в заголовки кнопкой **Главная** → **Использовать первую строку в качестве заголовков** (Home → Use first row as headers) и заполним пустые ячейки в первых двух столбцах командой **Заполнить вниз** (Fill Down), как мы уже делали в п. 1.:

	1;2; ::	А;С; ;_1	А;С; Город;Менеджер;Компа...	1;2; Москва;Виктория;Skoda	1;2; Москва;Виктория;Volkswagen	1;2; Москва;Владисл
1	2015	Q1	План		81	12
2	2015	Q1	Факт		40	59
3	2015	Q2	План		92	61
4	2015	Q2	Факт		13	19
5	2015	Q3	План		79	45
6	2015	Q3	Факт		24	85
7	2015	Q4	План		51	81
8	2015	Q4	Факт		69	51
9	2016	Q1	План		50	59

5. Теперь выделим первых три столбца и развернём остальные с помощью команды **Отменить свёртывание других столбцов (Unpivot Other Columns)** на вкладке **Преобразование (Transform)**:

1.2.3 ;;	А.В.С ;:_1	А.В.С Город;Менеджер;Компа...	А.В.С Атрибут	1.2 Значение	
1	2015	Q1	План	Москва;Виктория;Skoda	81
2	2015	Q1	План	Москва;Виктория;Volkswagen	12
3	2015	Q1	План	Москва;Владислав;Fiat	50
4	2015	Q1	План	Москва;Владислав;Saab	10
5	2015	Q1	План	СПБ;Артём;Hyundai	87
6	2015	Q1	План	СПБ;Артём;Mercedes	49
7	2015	Q1	План	СПБ;Татьяна;Mini	48
8	2015	Q1	План	СПБ;Татьяна;Mitsubishi	76
9	2015	Q1	План	СПБ;Татьяна;Rover	60
10	2015	Q1	Факт	Москва;Виктория;Skoda	40
11	2015	Q1	Факт	Москва;Виктория;Volkswagen	59
12	2015	Q1	Факт	Москва;Владислав;Fiat	30
13	2015	Q1	Факт	Москва;Владислав;Saab	63

6. Осталось расцепить скленные город, имя и бренд в столбце **Атрибут** командой **Преобразование → Разделить столбец → По разделителю (Transform → Split Column → By Delimiter)** и переименовать столбцы двойным щелчком мыши:

1.2.3 Год	А.В.С Квартал	А.В.С Статус	А.В.С Город	А.В.С Менеджер	А.В.С Бренд	1.2 Значение	
1	2015	Q1	План	Москва	Виктория	Skoda	81
2	2015	Q1	План	Москва	Виктория	Volkswagen	12
3	2015	Q1	План	Москва	Владислав	Fiat	50
4	2015	Q1	План	Москва	Владислав	Saab	10
5	2015	Q1	План	СПБ	Артём	Hyundai	87
6	2015	Q1	План	СПБ	Артём	Mercedes	49
7	2015	Q1	План	СПБ	Татьяна	Mini	48
8	2015	Q1	План	СПБ	Татьяна	Mitsubishi	76
9	2015	Q1	План	СПБ	Татьяна	Rover	60
10	2015	Q1	Факт	Москва	Виктория	Skoda	40
11	2015	Q1	Факт	Москва	Виктория	Volkswagen	59
12	2015	Q1	Факт	Москва	Владислав	Fiat	30
13	2015	Q1	Факт	Москва	Владислав	Saab	63
14	2015	Q1	Факт	СПБ	Артём	Hyundai	20

Вот и всё, можно выгружать данные на лист (или оставлять в виде подключения) и строить по ним сводную любого нужного нам вида.

Вдогон хотелось бы заметить, что решение любой нестандартной задачи в Power Query – это всегда немного импровизация. Мой алгоритм, описанный выше, безусловно, лишь один из многих способов решения этой задачи. Вполне возможно, что вы придумаете свой, более удобный, изящный и за меньшее количество шагов достигающий того же результата. Пробуйте, дерзайте, ошибайтесь и учитесь на ошибках – это единственный способ приобретения настоящих знаний и навыков. А лишние шаги в Power Query всегда можно удалить в правой панели. :)

Отмена свёртывания сразу нескольких таблиц

В завершение разговора об отмене свёртывания давайте рассмотрим ещё один интересный пример, где мощь этого инструмента значительно увеличивается путем добавления всего лишь пары строчек кода на языке M.

Представьте, что нам досталась книга с большим количеством «умных» кросс-таблиц, каждая из которых представляет собой данные по продажам за определенный месяц. Каждую таблицу нам надо развернуть в плоскую, а потом соединить с другими, чтобы получить на выходе общую таблицу, по которой можно будет построить сводную:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	Менеджер	Сергиев Посад	Каменск - Уральский	Псков	Владикавказ	Арзамас	Северск	Кемерово	Новочеркасск	Северодвинск	Королёв	Пушкино	Ставрополь	Пермь	Великий Новгород	Ессентуки	Норильск		
2	Дмитрий	10	7	16	8		9	16	7	4	13	5	1	10	1	1	13		
3	Екатерина	18		7	13	8	14	19	7	17	3	11	8		19	0	19		
4	Алина		11	15	18	18		7	6	6		17	18	14	11		6		
5	Александр	7	6	13	12	12	8		17	17	3	4	4	2	12	19	6		
6																			
7																			
8	Менеджер	Курган	Нижнекамск	Москва	Ноябрьск	Курск	Одинцово	Камышин	Майкоп	Невинномысск	Махачкала	Улан-Удэ							
9	Максим	29	28	12	0	9	16	5	4	23	24	13							
10	Александр		15	5	2	2	23	14	4	14	1								
11	Алина	20	21		23	7			25	15	3	17							
12	Мария	8	10	9	19	24	25	15	29	4		23							
13	Арина	9	27	1	23	9		24	18	16		18							
14	Михаил	19	7	19	25	18	29	20	8	27	16	27							
15																			
16																			
17	Менеджер	Нижний Новгород	Новосибирск	Майкоп	Йошкар-Ола	Дмитровград	Кострома	Казань	Барнаул	Мурманск	Смоленск	Новый Уренгой	Уфа	Чита					
18	Сергей	14	23	24	15	20	10	28		16	16	20	17	24					
19	Анна	27	10	11		19	25	23	10	24	7		21	14					
20	Илья	11		23	17	13		18	5	28	24	21	21	1					
21	Арина	21	28	27	19	18	8	5	28		18	27	6	28					
22																			
23																			
24	Менеджер	Вологда	Курган	Черкесск	Сергиев Посад	Раменское	Уфа	Ижевск	Арзамас	Альметьевск	Норильск	Ангарск	Пушкино	Новый Уренгой	Махачкала	Дербент	Рыбинск		
25	Артём	37	9	31	5	37	36	38	11	12	6	22	18	24	0	39	21		
26	Анна	31		13	36	2	22	36	13	39		7	12	27	13	7	25		
27	Кирилл	23	15	36		19	25	16	5	13	39	37	14	17		23	4		
28	Елена	22	4	19	12	7	33	31	20	13		19	34	20	5	28	4		
29	Екатерина	15	35	18	33	0	22	32	14	38	3	23	9		18	4	3		
30	Алина	13	17		28	15		5	31	15		18	32	30	37	16	14		

Само собой, загружать каждую таблицу по отдельности, как мы это делали ранее, здесь не стоит. Лучше воспользоваться приемом, который мы применяли в главе [Импорт всех «умных» таблиц из текущей книги](#), а именно следующим.

Создадим в нашей книге пустой запрос, выбрав на вкладке **Данные** → **Получить данные** → **Из других источников** → **Пустой запрос** (Data → Get Data → From Other Sources → Blank query).

В открывшемся пустом окне редактора Power Query в строку формул введём формулу:

```
=Excel.CurrentWorkbook()
```

и нажмем на **Enter**, чтобы получить список содержимого текущего файла:

	ABC 123 Content	AB C Name
1	Table	янв_2016
2	Table	фев_2016
3	Table	мар_2016
4	Table	апр_2016
5	Table	май_2016
6	Table	июн_2016
7	Table	июл_2016
8	Table	авг_2016
9	Table	сен_2016
10	Table	окт_2016
11	Table	ноя_2016
12	Table	Таблицаб
13	Table	НДС
14	Table	Лист3!Область_печати

Хорошо видно, что кроме нужных нам таблиц с месяцами в книге есть еще область печати, именованный диапазон (НДС) и какая-то дополнительная ненужная нам таблица (Таблицаб). Чтобы отфильтровать ненужные данные и одновременно развернуть таблицы с месяцами, добавим вычисляемый столбец при помощи кнопки **Настраиваемый столбец** с вкладки **Добавление столбца** (Add Column → Custom Column). В открывшееся окно введём функцию языка M, которая и выполнит отмену свертывания:

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

Доступные столбцы:

Content
 Name

[Сведения о формулах Power Query](#)

Синтаксические ошибки не обнаружены.

```
=Table.UnpivotOtherColumns([Content], {"Менеджер"}, "Город", "Кол-во")
```

где

- **[Content]** – имя столбца, содержащего исходные кросс-таблицы для развёртывания;
- **{"Менеджер"}** – имя одного или нескольких (через запятую) столбцов в исходных таблицах, которые не должны разворачиваться;
- два последних аргумента **"Город"** и **"Кол-во"** – это имена новых столбцов, которые получаются после отмены свертывания (обычно они называются **Атрибут** и **Значение**, но мы можем дать им здесь другие имена).

После нажатия на **OK** мы получим новый столбец, в котором будут содержаться уже развернутые таблицы напротив месяцев или ошибки (Error) напротив ненужных нам областей печати и именованных диапазонов (т. к. в них нет столбца **Менеджер**):

	ABC 123 Content	ABC Name	ABC 123 Таблицы
1	Table	январь_2016	Table
2	Table	февраль_2016	Table
3	Table	март_2016	Table
4	Table	апрель_2016	Table
5	Table	май_2016	Table
6	Table	июнь_2016	Table
7	Table	июль_2016	Table
8	Table	август_2016	Table
9	Table	сентябрь_2016	Table
10	Table	октябрь_2016	Table
11	Table	ноябрь_2016	Table
12	Table	Таблица6	Error
13	Table	НДС	Error
14	Table	Лист3!Область_печати	Error

Менеджер	Город	Кол-во
Анна	Жуковский	46
Анна	Тверь	25
Анна	Калуга	12
Анна	Владивосток	41
Анна	Чита	2
Анна	Кызыл	15

Чтобы удалить строки с ошибками, щелчком правой кнопкой мыши по заголовку столбца **Таблицы** и выберем команду **Удалить ошибки (Remove Errors)**. Заодно можно удалить и ненужный более столбец **Content**.

Чтобы получить в столбце **Name** полноценную дату, пригодную для дальнейшей работы, группировки в сводной и т. д., можно сделать следующее.

1. Заменить подчеркивание на пробел, щелкнув по заголовку столбца правой кнопкой мыши и выбрав команду **Замена значений (Replace Values)**.
2. Выбрать для этого столбца на вкладке **Преобразование** → **Дата** → **Выполнить анализ (Transform → Date → Parse)**.

Осталось развернуть содержимое вложенных таблиц в колонке **Таблицы**, используя кнопку с двойными стрелками в шапке и не забыв отключить флажок **Использовать исходное имя столбца как префикс (Use original column name as prefix)**:

	Дата	ABC 123 Таблицы
1	01.01.2016	Table
2	01.02.2016	Table
3	01.03.2016	Table
4	01.04.2016	Table
5	01.05.2016	Table
6	01.06.2016	Table
7	01.07.2016	Table
8	01.08.2016	Table
9	01.09.2016	Table
10	01.10.2016	Table
11	01.11.2016	Table

Поиск столбцов, которые нужно развернуть

Развернуть Агрегирование

(Выбрать все столбцы)

Менеджер

Город

Кол-во

Использовать исходное имя столбца как префикс

OK Отмена

и после нажатия на **OK** мы увидим объединенное содержимое всех исходных «умных» таблиц – развёрнутое в виде одной большой плоской таблицы, как мы и хотели. Останется выгрузить результаты обратно в Excel с помощью команды **Главная** → **Закрыть и загрузить** → **Закрыть и загрузить в** → **Только создать подключение (Close&Load → Close&Load to → Only Create Connection)** и построить потом сводную таблицу по этому подключению, как мы уже делали ранее (см. главу [Построение сводной таблицы по результатам запроса](#)).

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Город	(Все)											
2													
3	Сумма по полю Кол-во	Месяцы	Дата										
4		янв	фев	мар	апр	май	июн	июл	авг	сен	окт	ноя	Общий итог
5	Менеджер												
6	Александр	142	80				796						222
7	Алёна												796
8	Алина	147	131		271								549
9	Анастасия								456				456
10	Анна			191	283	346							820
11	Арина		145	233									378
12	Артём				346								346
13	Вероника							783			716		1499
14	Виталий										711	694	1405
15	Владимир						708						708
16	Владислав							608					608
17	Глеб						718		680				1398
18	Дарина							624					624
19	Дмитрий	121			367								488
20	Егор											761	761

И последний нюанс. Если собранные данные вы не оставите как подключение, а решите выгрузить на отдельный лист в этом же файле, то полученная «умная» таблица будет попадать потом в список, формируемый функцией **Excel.CurrentWorkbook**, т. е. возникнет рекурсия. Поэтому необходимо будет добавить в наш запрос дополнительный шаг для её фильтрации, как мы уже делали ранее (см. главу [Исключаем рекурсию](#)).

Подтягивание значений к краю таблицы

Это хотя и весьма специфическая, но достаточно распространенная проблема, возникающая при наведении порядка в поступающих к нам из внешнего мира данных. Представьте, что в исходной таблице по каким-то причинам полезная информация неравномерно перемежается с пустыми ячейками. Нам же нужно «подтянуть» все значения к одному из краев таблицы (например, к верхнему), чтобы получить монолитные блоки без пустых ячеек внутри:

Мария	Иван	Юлия	Ольга	Сергей	Олег	Елена	Роман
46		9					
	25			30	52		98
		68	3		55		
48			59				
17	87	1			27		
		50	25		50	69	
	40			17			4



Мария	Иван	Юлия	Ольга	Сергей	Олег	Елена	Роман
46	25	9	3	30	52	69	98
48	87	68	59	17	55		4
17	40	1	25		27		
		50			50		

После загрузки исходной таблицы в Power Query для получения желаемого результата можно выполнить следующую цепочку шагов:

1. Опускаем названия столбцов (имена людей) из шапки в первую строку данных с помощью команды **Главная → Использовать заголовки как первую строку** (Home → Use headers as a first row).
2. Транспонируем таблицу через **Преобразование → Транспонировать** (Transform → Transpose).
3. Выделяем все столбцы, кроме первого, и склеиваем их значения в новый дополнительный столбец, используя любой подходящий символ-разделитель, например точку с запятой, через **Добавление столбца → Объединить столбцы** (Add Column → Merge Columns). Наши исходные данные склеятся, игнорируя пустые ячейки:

	ABC 123 Column1	ABC 123 Column2	ABC 123 Column3	ABC 123 Column4	ABC 123 Column5	ABC 123 Column6	ABC 123 Column7	ABC 123 Column8	ABC Сведено
1	Мария	46	null	null	48	17	null	null	46;48;17
2	Иван	null	25	null	null	87	null	40	25;87;40
3	Юлия	9	null	68	null	1	50	null	9;68;1;50
4	Ольга	null	null	3	59	null	25	null	3;59;25
5	Сергей	null	30	null	null	null	null	17	30;17
6	Олег	null	52	55	null	27	50	null	52;55;27;50
7	Елена	null	null	null	null	null	69	null	69
8	Роман	null	98	null	null	null	null	4	98;4

4. Удаляем все столбцы, кроме первого и последнего.
5. Разделяем склеенный столбец **Сведено** обратно по точке с запятой, используя команду **Преобразование → Разделить столбец → По разделителю** (Transform → Split Column → By delimiter).
6. Транспонируем таблицу обратно и поднимаем имена людей из первой строки в шапку командой **Главная → Использовать первую строку в качестве заголовков** (Home → Use first row as headers).

	1²³ Мария	1²³ Иван	1²³ Юлия	1²³ Ольга	1²³ Сергей	1²³ Олег	1²³ Елена	1²³ Роман
1	46	25	9	3	30	52	69	98
2	48	87	68	59	17	55	null	4
3	17	40	1	25	null	27	null	null
4	null	null	50	null	null	50	null	null

Операции с текстом

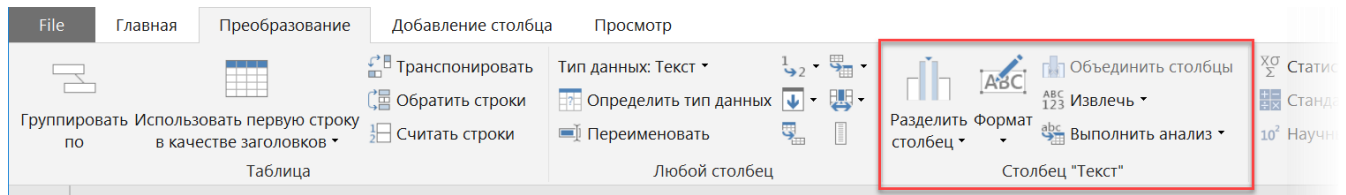
Power Query имеет в своём арсенале большой набор инструментов для обработки и всяческой трансформации текста. В этой главе мы подробно разберём их все и научимся:

- **разделять** «слипшийся» текст на отдельные столбцы или строки;
- приводить в порядок **регистр**;
- **очищать** текст от лишних пробелов, непечатаемых символов и прочего «мусора»;
- **склеивать** текст разными способами;
- использовать встроенный «искусственный интеллект» Power Query с помощью инструмента **Столбец из примеров**;
- быстро **генерировать фразы из заданных слов** с помощью декартова произведения;
- реализовывать **нечёткий (fuzzy) текстовый поиск** для поиска слов с ошибками и опечатками.

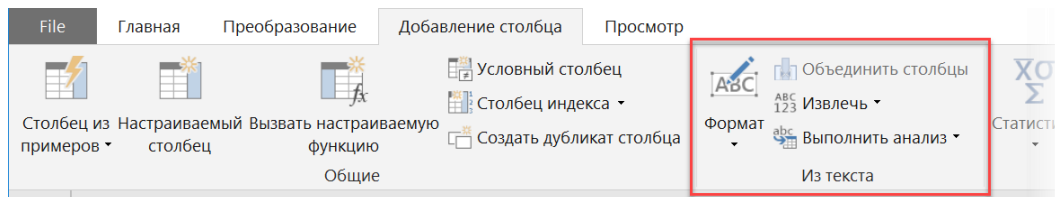


Важное замечание

Практически все инструменты, рассматриваемые в последующих главах, присутствуют на ленте редактора Power Query в двух ипостасях – на вкладке **Преобразование (Transform)**:



...и на вкладке **Добавление столбца (Add Column)**:

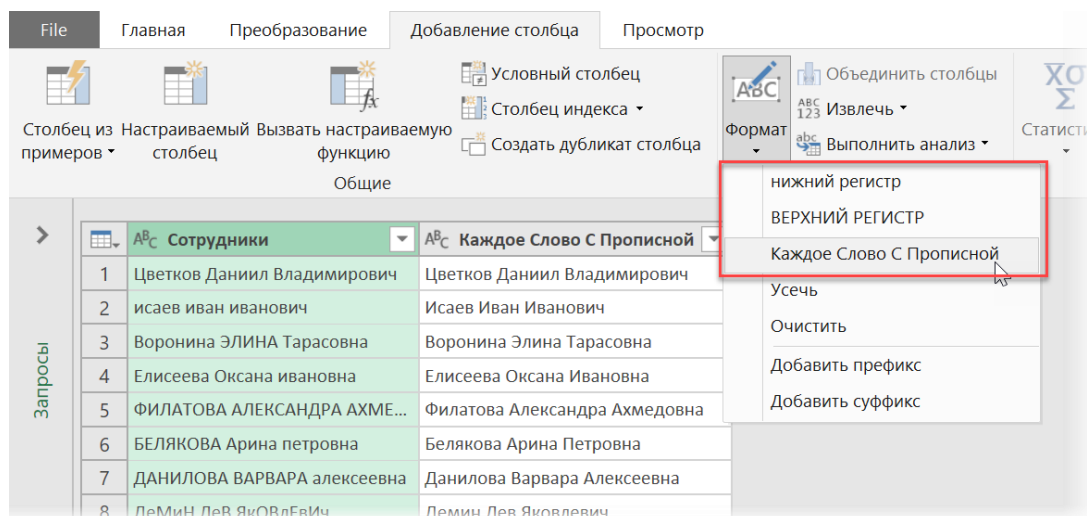


Функционал здесь практически одинаков, а разница только в том, что, применяя инструменты с вкладки **Преобразование (Transform)**, мы изменяем данные на месте, в том же столбце, где они находятся. При использовании тех же функций с вкладки **Добавление столбца (Add Column)** преобразованные данные будут помещены в новый, дополнительно созданный столбец, оставляя колонку с исходными данными нетронутой.

Причём это касается не только текстовых инструментов, но и функций обработки дат-времени, математических вычислений и т. д.

Изменение регистра

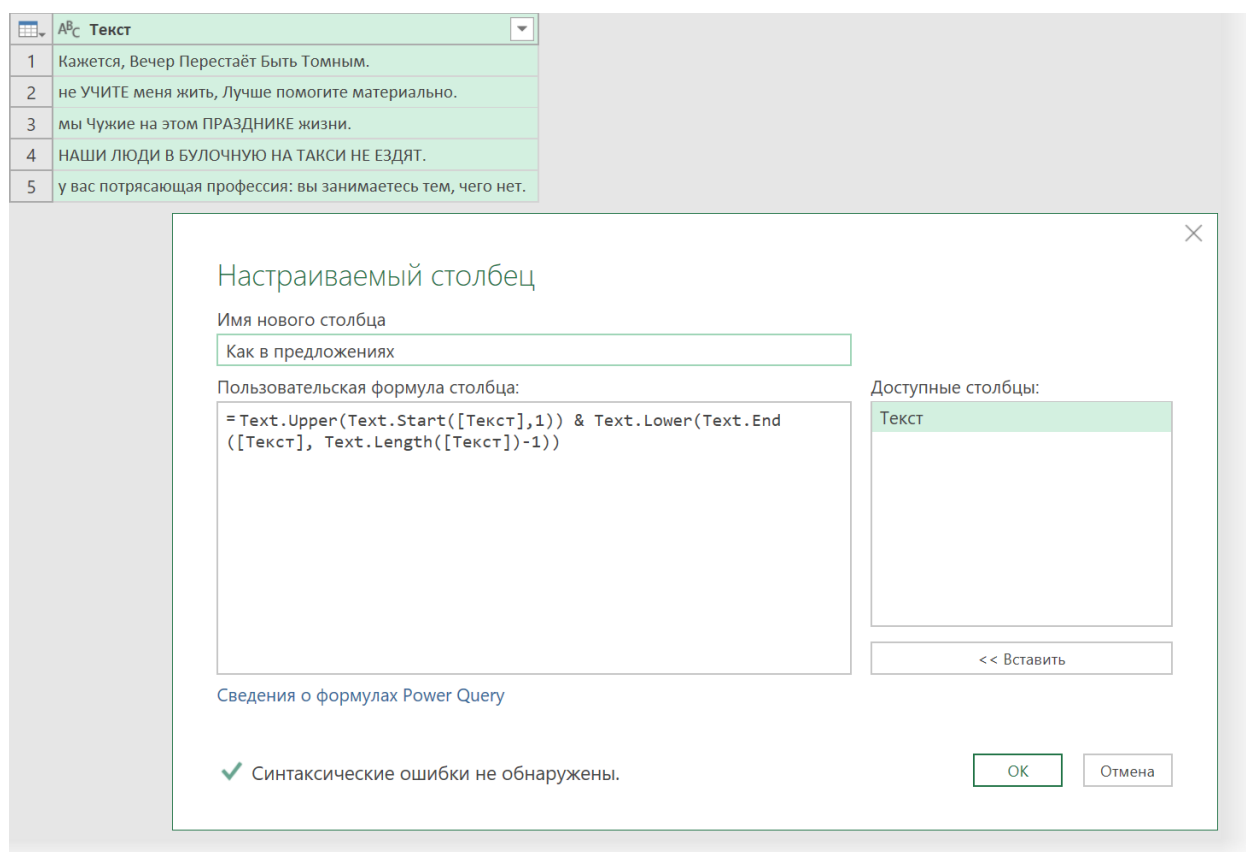
Это простой, но нужный инструмент, позволяющий быстро привести регистр исходного текста к нужному виду. Добраться до него можно, используя кнопку **Формат (Format)** на вкладке **Преобразование (Transform)** или **Добавление столбца (Add Column)**:



Как легко сообразить, первые три команды в выпадающем списке меняют регистр текста, превращая его в строчные, прописные или делая прописными только первые буквы в каждом слове. В Microsoft Excel эту роль выполняют функции рабочего листа **СТРОЧН (LOWER)**, **ПРОПИСН (UPPER)** и **ПРОПНАЧ (PROPER)**.

Единственное, чего, на мой взгляд, не хватает в этом наборе – это варианта, обычно называемого «Как в предложениях», когда заглавной становится только первая буква во всей ячейке, а не каждая начальная буква в каждом слове. Реализовать недостающее можно с помощью дополнительного столбца с небольшой формулой на встроенном в Power Query языке M.

Для этого на вкладке **Добавление столбца** выберем **Настраиваемый столбец (Add Column → Custom Column)** и введём в открывшемся окне вот такую конструкцию:



```
=Text.Upper(Text.Start([Текст],1)) & Text.Lower(Text.End([Текст], Text.Length([Текст])-1))
```

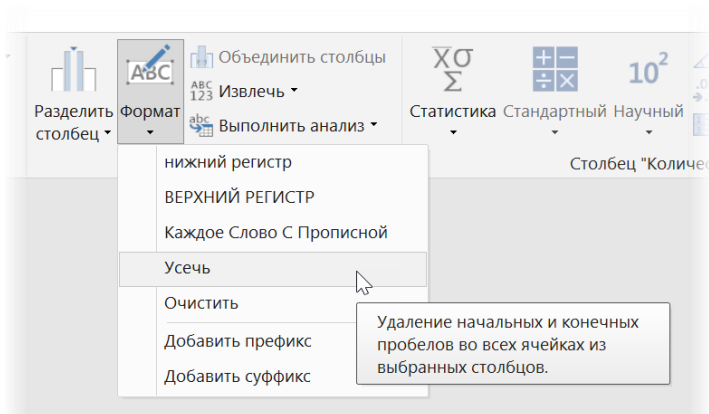
где:

- **Text.Upper** – функция языка M, конвертирующая текст, указанный в качестве аргумента, в верхний регистр (прописные);
- **Text.Lower** – то же самое, что предыдущая функция, но преобразует весь текст в нижний регистр (строчные)
- **Text.Length** – определяет длину исходной строки текста
- **Text.Start** – выдает заданное количество символов от начала строки текста – аналог экселевской функции **ЛЕВСИМВ (LEFT)**.

Логика работы этой формулы проста: мы отщипываем от строки начальный символ функцией **Text.Start** и преобразуем его в верхний регистр с помощью **Text.Upper**. Затем с помощью "&" приклеиваем к полученному заглавному символу остальную строку, преобразованную, в свою очередь, в нижний регистр с помощью функции **Text.Lower**.

Удаление лишних пробелов и SuperTrim

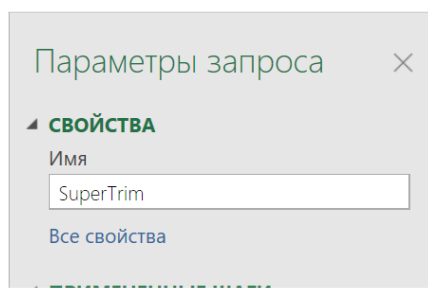
Лишние пробелы очень часто присутствуют в исходных данных и представляют собой весьма надоедливую проблему. В самом Excel для избавления от них есть встроенная функция¹ **СЖПРОБЕЛЫ** (TRIM). В Power Query есть её аналог – команда **Усечь** (Trim) в выпадающем списке **Формат** (Format) на вкладках **Преобразование** (Transform) и **Добавить столбец** (Add Column):



К сожалению, эта команда не на 100% повторяет возможности функции **СЖПРОБЕЛЫ** и почему-то убирает только начальные и конечные лишние пробелы, но не избавляет от лишних (двойных, тройных и т. д.) пробелов между словами. Если всё же потребуется это сделать, то можно использовать возможности языка M и создать свою функцию, которая будет чистить текст от всех лишних пробелов – внутренних и внешних.

Для этого:

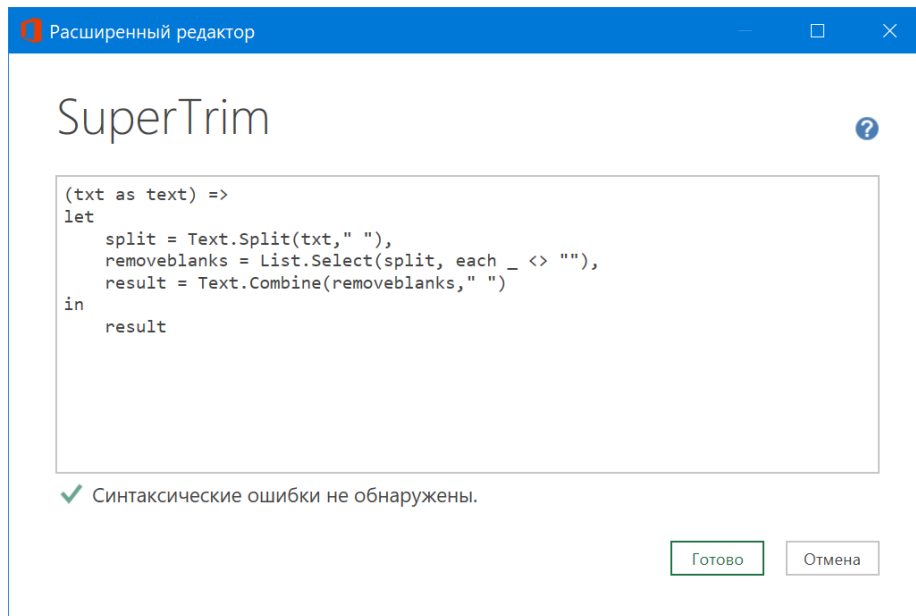
1. Откроем пустой запрос в Power Query через **Данные** → **Получить данные** → **Из других источников** → **Пустой запрос** (Data → Get Data → From Other Sources → Blank Query).
2. Сразу дадим ему понятное имя (например, *SuperTrim*) в правой панели:



3. Затем на вкладке **Просмотр** откроем **Расширенный редактор** (View → Advanced Editor) и введём туда вот такой код:

```
(txt as text) =>
let
    split = Text.Split(txt, " "),
    removeblanks = List.Select(split, each _ <> ""),
    result = Text.Combine(removeblanks, " ")
in
    result
```

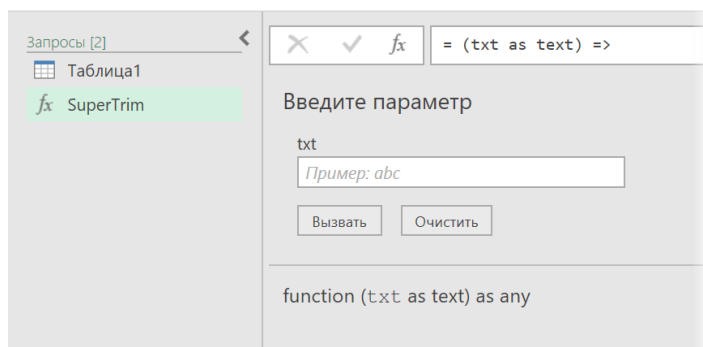
¹ Подробнее о ней и её аналогах можно почитать тут <https://www.planetaexcel.ru/techniques/25/2734/>.



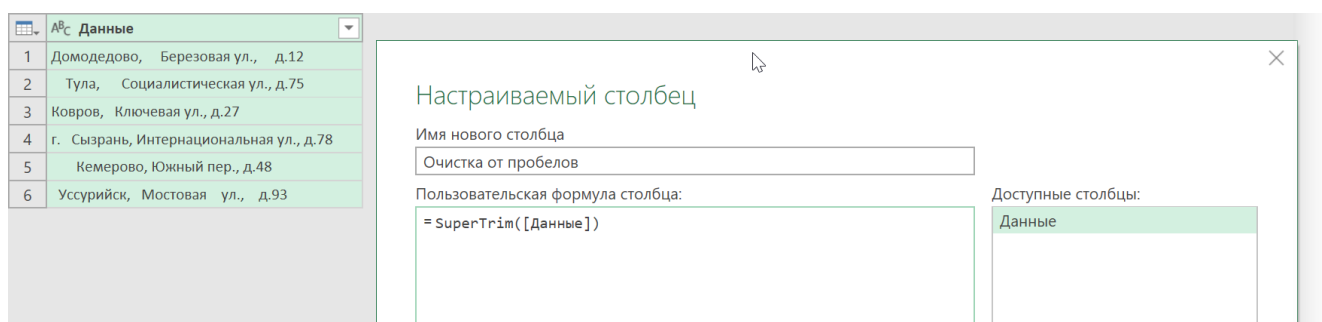
Логика работы этого кода:

1. Мы объявляем функцию с текстовой переменной **txt** в виде аргумента.
2. Делим весь полученный функцией текст по разделителю-пробелу с помощью функции **Text.Split** и получаем на выходе список (List) из отдельных слов-элементов.
3. Проходим по списку и выбираем из него непустые элементы, игнорируя пустые (они образуются из лишних пробелов).
4. Склеиваем отобранные непустые элементы обратно через один пробел функцией **Text.Combine**.

После нажатия на **Готово** мы должны увидеть вот такую картину:



Теперь можно загрузить в Power Query наш проблемный текст и добавить к нему вычисляемый столбец с нашей функцией. Для этого выберем на вкладке **Добавить столбец** команду **Настраиваемый столбец** (Add Column → Custom Column) и введём в открывшееся окно нашу функцию, указав в качестве её аргумента столбец текста с лишними пробелами.



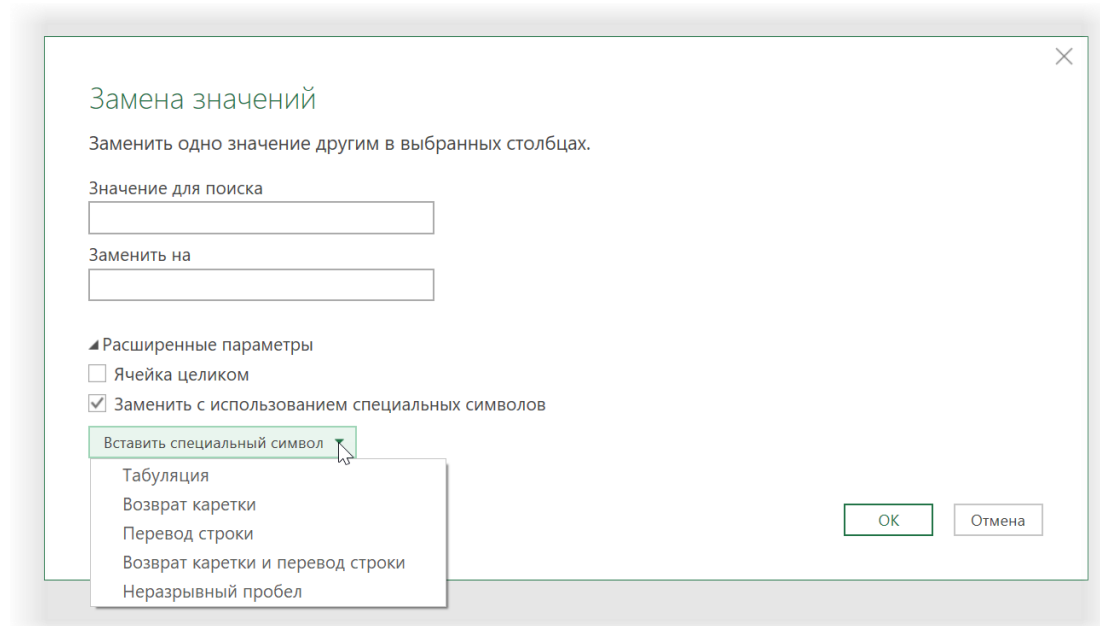
В последних версиях Power Query можно ещё воспользоваться кнопкой **Вызвать настраиваемую функцию** (Invoke Custom Function): она выполняет ту же роль, что и вычисляемый столбец с нашей функцией в данном случае.

Очистка текста от непечатаемых символов

При импорте данных с веб-страниц или из некоторых ERP-систем в числовых или текстовых данных могут содержаться различные непечатаемые символы (неразрывные пробелы, символы табуляции, переносы строк и т. д.). Конечно, они мешают последующей полноценной работе с данными и должны быть зачищены.

От переносов строк (**Alt+Enter**) и символов табуляции можно избавиться с помощью команды **Формат → Очистить** (Format → Clear) на вкладках **Преобразование** (Transform) или **Добавление столбца** (Add Column).

Однако эта функция не удаляет неразрывные пробелы. Чтобы удалить их (или заменить на обычные) лучше использовать стандартную процедуру замены «на ничего». Для этого щелкните правой кнопкой мыши по заголовку столбца с текстом и выберите команду **Замена значений** (Replace Values). В открывшемся окне в поле **Значение для поиска** можно ввести непечатаемые символы, если включить в **Расширенных параметрах** флажок **Заменить с использованием специальных символов** (Replace using special characters) и выбрать затем нужный символ из выпадающего списка:



Также можно ввести код соответствующего символа в поле поиска вручную:

Символ	Код
Табуляция	#(tab)
Возврат каретки	#(cr)
Перевод строки	#(lf)
Неразрывный пробел	#(00A0)

Разделение «слипшегося» текста

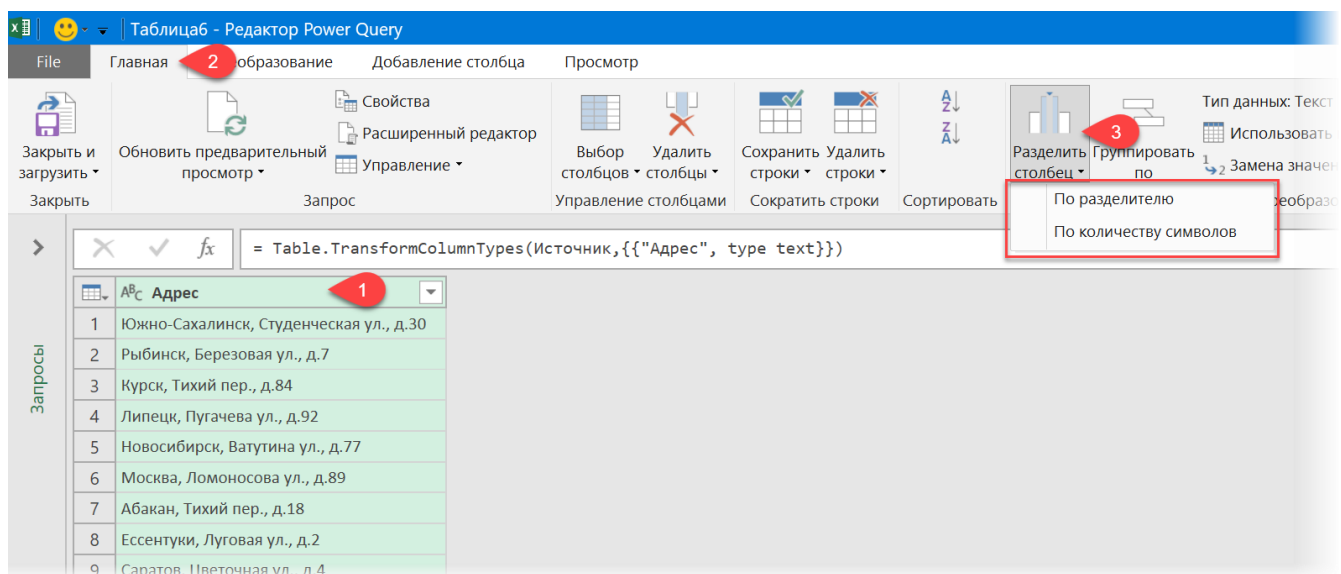
*Что один человек собрал, другой завсегда
разобрать сможет.
(К-ф «Формула любви»)*

В отличие от стандартного инструмента Excel **Текст по столбцам (Text to Columns)**¹ аналогичный инструмент в Power Query имеет ряд интересных особенностей и является гораздо более удобным в использовании. Давайте рассмотрим несколько сценариев его применения.

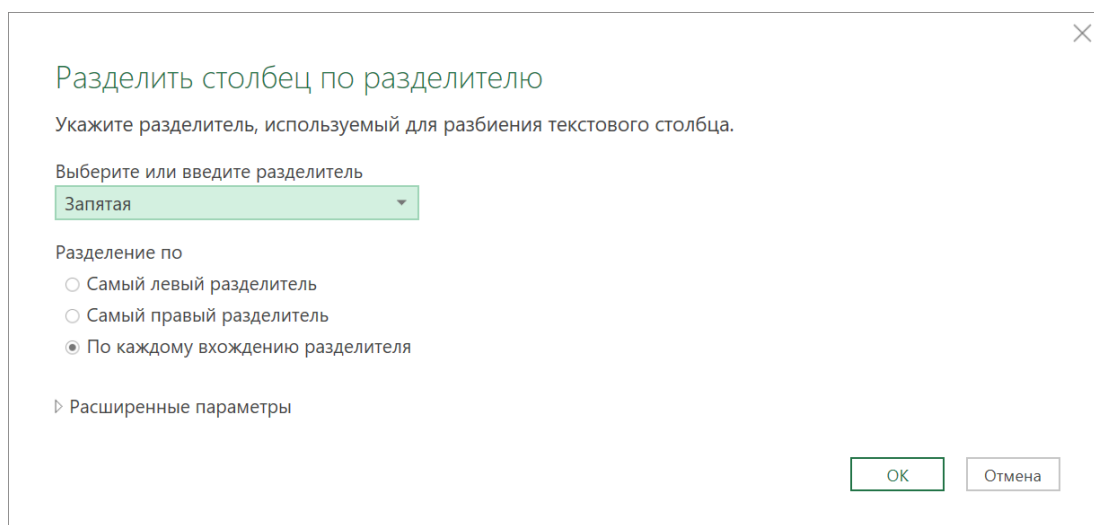
Простой случай

В самом простом случае работа этого инструмента выглядит так:

1. Выделяем столбец, который надо разделить.
2. Переходим на вкладку **Главная (Home)** или на вкладку **Преобразование (Transform)**.
3. Жмём на кнопку **Разделить столбец (Split Column)** и выбираем желаемый способ – **По разделителю (By Delimiter)** или **По количеству символов (By Number of Characters)**:



Дальнейшее зависит от того, какой метод деления мы выбрали. Если выбран первый вариант, то в следующем окне нам будет предложено выбрать символ-разделитель:



¹ Это замечательный инструмент подробно описан в моей первой книге «Microsoft Excel: Готовые решения – бери и пользуйся!» в главе «Работа с текстом».

Если же мы решили делить текст по количеству символов, то в следующем окне нужно будет ввести число знаков, после которого должно произойти отделение в новый столбец:

✕

Разделить столбец по количеству символов

Укажите количество символов для разделения текстового столбца.

Количество символов

Разделение

Однократно, как можно левее

Однократно, как можно правее

Каждый раз

▷ Расширенные параметры

Обратите внимание, что делить можно как по каждому вхождению (или количеству символов), так и однократно – с левого или правого края текстовой строки.

Также нужно отметить, что в обоих случаях исходный столбец не остаётся «в живых», т. е. мы получим вместо него несколько столбцов с разделённым текстом:

	АВС Адрес.1	АВС Адрес.2	АВС Адрес.3
1	Южно-Сахалинск	Студенческая ул.	д.30
2	Рыбинск	Березовая ул.	д.7
3	Курск	Тихий пер.	д.84
4	Липецк	Пугачева ул.	д.92
5	Новосибирск	Ватутина ул.	д.77
6	Москва	Ломоносова ул.	д.89
7	Абакан	Тихий пер.	д.18
8	Ессентуки	Луговая ул.	д.2
9	Саратов	Цветочная ул.	д.4

Если нужно получить их и при этом оставить оригинал в нетронутым состоянии, то лучше предварительно сделать копию столбца, щелкнув по его заголовку правой кнопкой мыши и выбрав команду **Создать дубликат столбца (Duplicate Column)**, а потом уже делить текст.

Деление на строки вместо столбцов

Классический инструмент **Текст по столбцам (Text to Columns)** в Excel – хорошая штука, но умеет делить только на столбцы. Иногда же оказывается, что удобнее и правильнее делить вместо столбцов на строки! Посмотрите на следующую задачу, которую озвучил мне недавно на тренинге один из слушателей:

	А Менеджер	В Суммы сделок	С	D	Е Менеджер	Ф Итого	Г
1	Марк	1970; 1334; 552			Марк	3856	
2	Злата	5347; 492			Злата	5839	
3	Милена	5842; 9317; 4688; 8058; 8877; 5508			Милена	42290	
4	Егор	2731; 6587; 7813; 7626			Егор	24757	
5	Виктор	2090			Виктор	2090	
6	Яна	3865; 2417; 7000; 6910; 3079; 6254; 3559; 5527			Яна	38611	
7	Оксана	4846; 593; 4117; 5623; 1748; 817			Оксана	17744	
8							
9							

Оставим в стороне пока вопрос о том, какой злой гений мог сделать такую «красивую» табличку или из какой адской программы в таком виде выгружаются данные. Суть в том, что нужно получить итоговые суммы по

каждому менеджеру, т. е. банально сложить все числа в каждой строке. Причем количество чисел может быть очень разным – от одного и до нескольких десятков.

Безусловно, решить эту задачу можно и стандартными средствами Microsoft Excel (текст по столбцам с последующей «доработкой напильником» или формулами той или иной степени сложности), но посмотрите, как красиво она решается средствами Power Query.

Загрузим таблицу в редактор Power Query и, выделив столбец **Суммы сделок**, выберем на вкладке **Главная (Home)** или **Преобразование (Transform)** команду **Разделить столбец → По разделителю (Split Column → By Delimiter)**.

В следующем окне выберем:

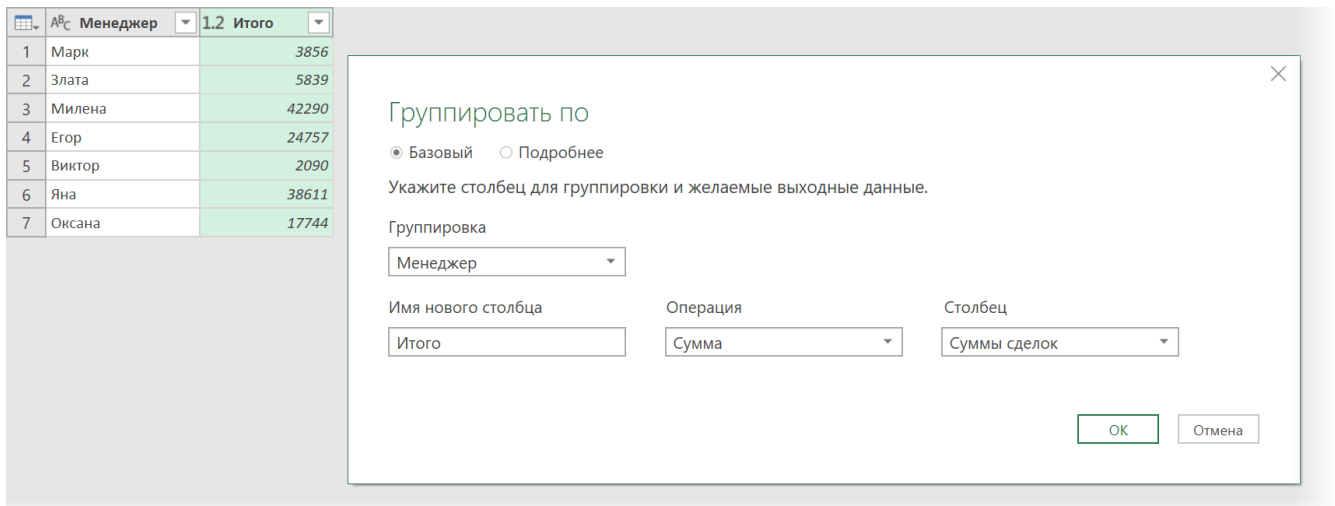
- в качестве разделителя – **Точку с запятой (Semicolon)**;
- разделение **По каждому вхождению разделителя (Each occurrence of the delimiter)**;
- и в **Расширенных параметрах (Advanced options)** деление на **Строки (Rows)**, а не на столбцы.

Менеджер	Суммы сделок
Марк	1970; 1334; 552
Злата	5347; 492
Милена	5842; 9317; 4688; 8058
Егор	2731; 6587; 7813; 7667
Виктор	2090
Яна	3865; 2417; 7000; 6900
Оксана	4846; 593; 4117; 5620

После нажатия на **ОК** мы увидим, что строки размножились, и в каждой строке теперь есть имя менеджера и отдельная сумма:

Менеджер	Суммы сделок
Марк	1970
Марк	1334
Марк	552
Злата	5347
Злата	492
Милена	5842
Милена	9317
Милена	4688
Милена	8058
Милена	8877

Останется применить группировку сумм по менеджерам, как мы это делали в главе [Простая группировка](#) с вкладки **Преобразование → Группировать по (Transform → Group by)**, и мы легко получим желаемый результат:



Несколько строк в одной ячейке

Теперь рассмотрим случай, когда текст в ячейках разделен не пробелом, запятой или чем-то аналогичным, а нестандартными символами, например неразрывным пробелом, табуляцией или разрывом строки (сочетание клавиш **Alt+Enter** при вводе в Excel):

	А	В
1	Менеджер	№ договора
		1627
		7105
2	Виктор	6017
		6172
		5921
		3676
		2271
3	Милана	9797
		1688
		1651
		8921
		9109
4	Арина	7812
		7498
5	Елизавета	2134
		2249

Для Power Query это тоже не проблема. После загрузки таблицы в редактор и выбора для выделенного столбца команды **Преобразование → Разделить столбец → По разделителю** (Transform → Split Column → By Delimiter) в открывшемся окне надо выбрать:

1. **Пользовательский (Custom)** символ-разделитель.
2. Разделение **По каждому вхождению разделителя** (Each occurrence of the delimiter).
3. В расширенных параметрах – деление на **Строки**, а не на столбцы (By rows).
4. И, самое главное, включить флажок **Разделить с помощью специальных символов** (Split using special characters), а затем выбрать из выпадающего списка **Перевод строки** (Line Feed), обозначаемый кодом **#(lf)**.

Всё вышеперечисленное нужно только в том случае, если Power Query сам не догадался это сделать (что, по моему опыту, происходит редко).

№	Менеджер	№ договора
1	Виктор	1627 7105 6017
2	Милана	6172 5921 3676 2271 9797
3	Арина	1688 1651 8921 9109 7812
4	Елизавета	7498 2134
5	Артур	2249 9877 4906 7428 4204
6	Кира	2681 7511 2417 2126
7	Марк	2521 5992
8	Алина	9406 3424
9	Денис	6681

Разделить столбец по разделителю

Укажите разделитель, используемый для разбиения текстового столбца.

Выберите или введите разделитель

--Пользовательский-- 1

#(lf) 2

Разделение по

- Самый левый разделитель
- Самый правый разделитель
- По каждому вхождению разделителя 2

Расширенные параметры

Разделение на

- Столбцы
- Строки 3

Символ кавычек

"

Разделить с помощью специальных символов 4

Вставить специальный символ

- Табуляция
- Возврат каретки
- Перевод строки 5
- Возврат каретки и перевод строки
- Неразрывный пробел

OK Отмена

После нажатия на **OK** мы опять легко получаем таблицу в гораздо более пристойном виде, удобном для дальнейшей обработки (группировки, фильтрации и т. д.):

№	Менеджер	№ договора
1	Виктор	1627
2	Виктор	7105
3	Виктор	6017
4	Милана	6172
5	Милана	5921
6	Милана	3676
7	Милана	2271
8	Милана	9797
9	Арина	1688
10	Арина	1651
11	Арина	8921
12	Арина	9109
13	Арина	7812

Разбор буквенно-цифровой каши

Иногда доставшиеся нам данные могут выглядеть совсем печально, представляя собой мешанину из букв и цифр. Извлечь из этой каши только числа для выполнения с ними дальнейших вычислений – задача нетривиальная и весьма трудоемкая, хотя и решаемая, обычно с помощью макросов или сложных формул¹.

В Power Query можно изящно победить эту проблему, если пойти от противного – удалить из исходного «грязного» текста все буквы, пробелы, точки, знаки доллара и т. д. Тогда в сухом остатке мы получим именно то, что нужно, – чистые числа, с которым можно дальше работать.

Чтобы это реализовать, добавим к нашей таблице ещё один вычисляемый столбец через вкладку **Добавление столбца** → **Настраиваемый столбец** (Add Column → Custom Column). Затем введём в открывшееся окно функцию, которая удаляет из исходного текста все указанные символы по списку:

	ABC 123 Данные
1	123,5 руб.
2	300 рублей
3	Цена вопроса 5 руб
4	Вопрос на 1 000 000 \$
5	100500 USD
6	Цена по 5,5 руб/шт

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

Доступные столбцы:

Данные

<< Вставить

[Сведения о формулах Power Query](#)

✓ Синтаксические ошибки не обнаружены.

```
=Text.Remove([Данные], {"$", ".", " ", "/" , "A".."я", "A".."z"})
```

Первый аргумент этой функции – это столбец с исходными данными. Второй аргумент – это символы, которые надо удалить. Обратите внимание на синтаксис.

- Набор (список) удаляемых символов заключается в фигурные скобки.
- Отдельные удаляемые символы заключаются в кавычки и перечисляются в наборе через запятую ("\$", "." и т. д.).
- Чтобы не перечислять весь алфавит, можно задать интервал удаляемых символов, используя две точки ("A"..."я").
- Маленькие и большие буквы различаются, поэтому вариант "A"..."Я" удалил бы только прописные, "a"..."я" – только строчные, а вариант "A"..."я" - удалит и те и другие.

После нажатия на **ОК** мы увидим очищенные от лишних символов цифровые данные. Останется задать для нового столбца числовой формат и работать дальше с этими данными уже нормально.

¹ См. главу «Извлечение чисел из текста» в моей книге «Microsoft Excel: Мастер Формул».

ABC 123	Данные	1.2 Только числа
1	123,5 руб.	123,5
2	300 рублей	300
3	Цена вопроса 5 руб	5
4	Вопрос на 1 000 000 \$	1000000
5	100500 USD	100500
6	Цена по 5,5 руб/шт	5,5

Разумеется, аналогичным способом можно избавляться от любых других ненужных символов. Так, например, можно легко решить и обратную задачу – оставить только текст, но убрать все цифры, если использовать формулу:

```
=Text.Remove([Данные], {"0"."9"})
```

Описанный выше трюк хорошо сочетается на практике с удалением лишних столбцов созданной нами ранее функцией SuperTrim и разделением текста на столбцы.

Рассмотрим в качестве примера более тяжелый случай – извлечение номеров и дат счетов из описаний платежей в банковской выписке:

ABC 123	Описание счета
1	Предоплата за оборудование по счету N10229 от 19.02.2015 г. в т.ч. НДС 18% - 30884.80
2	Оплата по счет-фактуре №26022015 от 26,02,2015 г. за товар 9057-09 руб. В т.ч. НДС (18%) 1381-59
3	Оплата по счету 10693 от 25.03.15, за насос.Сумма 258807-50В т.ч. НДС(18%) 39479-11
4	Оплата за материалы по счету №11089 от 17.04.2015гСумма 52350-06В т.ч. НДС(18%) 7985-60
5	Оплата по Сч N 11273 от 29.04.15г за насосСумма 32475-02В т.ч. НДС (18%) 4953-82
6	ЗА МОТОР ПО СЧЕТУ 1541 ОТ 20.05.2015 НДС В ТОМ ЧИСЛЕ: 2179-63
7	Оплата по сч-ф 1500147 от 02.06.15 за насосы В том числе НДС 2092.03
8	Оплата по сч 11768 от 03.06.15 за сервисный комплект в т.ч. НДС (18 %) - 550-48
9	Оплата по счету N12214 от 01.07.2015 за насос в т.ч. НДС 2654-37
10	По Счет 1148 от 19.08.15 За товар Сумма 786800-00 В т.ч. НДС(18%) 120020-34
11	Окончательный расчет по счету №12303 от 06.07.15г за печать листовок Сумма 321472-45В т.ч. НДС (18%) 49038-17
12	Оплата по счету 1338 от 31.08.2015 г. Сумма 61 342-18 руб. за оборуд. в т.ч. НДС 18% - 9357.28 руб.
13	Оплата по счету № 14655 от 10.11.15 за ТМЦ, В т.ч.НДС 23810,44 руб.
14	Оплата по счету 14992 от 26.11.15, за комплектующие.Сумма 46341-79В т.ч. НДС(18%) 7069-09

Если присмотреться, то можно заметить, что в подавляющем большинстве строк первым числом слева является номер счёта, а за ним следует обычно дата. Так что если добавить к нашим данным настраиваемый столбец с формулой:

```
=Text.Remove([Описание счета],{"А"."я", "№", "-", "/", "А"."z"})
```

то мы вычистим из описаний все русские и английские буквы и ненужные знаки препинания, оставив только точки и запятые как разделитель дат и пробел, чтобы всё не слиплось в одну кучу:

ABC 123	Числа_и_точки
	10229 19.02.2015 . .. 18% 30884.80
	26022015 26,02,2015 . 905709 . .. (18%) 138159
	10693 25.03.15, . 25880750 .. (18%) 3947911
	11089 17.04.2015 5235006 .. (18%) 798560
	11273 29.04.15 3247502 .. (18%) 495382
	1541 20.05.2015 :217963
	1500147 02.06.15 2092.03
	11768 03.06.15 .. (18 %) 55048
	12214 01.07.2015 .. 265437
	1148 19.08.15 78680000 .. (18%) 12002034
	12303 06.07.15 32147245 .. (18%) 4903817

Теперь к полученному столбцу можно добавить ещё один вычисляемый столбец с созданной нами ранее для удаления лишних пробелов функцией **SuperTrim**:

=SuperTrim([Числа_и_точки])

ABC 123	Без_лишних_пробелов
	10229 19.02.2015 ... 18% 30884.80
	26022015 26,02,2015 . 905709 ... (18%) 138159
	10693 25.03.15, . 25880750 .. (18%) 3947911
	11089 17.04.2015 5235006 .. (18%) 798560
	11273 29.04.15 3247502 .. (18%) 495382
	1541 20.05.2015 : 217963
	1500147 02.06.15 2092.03
	11768 03.06.15 .. (18 %) 55048
	12214 01.07.2015 .. 265437
	1148 19.08.15 78680000 .. (18%) 12002034
	12303 06.07.15 32147245 .. (18%) 4903817
	1338 31.08.2015 . 61 34218 ... 18% 9357.28 .

И останется разделить этот столбец по пробелу с помощью команды **Преобразование → Разделить столбец → По разделителю** (Transform → Split Column → By delimiter), чтобы в первых двух столбцах как раз и оказались нужные нам номера счетов и их даты:

123	Без_лишних_пробелов.1	Без_лишних_пробелов.2	ABC
	10229	19.02.2015	.
	26022015	26.02.2015	.
	10693	25.03.2015	.
	11089	17.04.2015	523
	11273	29.04.2015	324
	1541	20.05.2015	:
	1500147	02.06.2015	209
	11768	03.06.2015	..
	12214	01.07.2015	..
	1148	19.08.2015	786
	12303	06.07.2015	321
	1338	31.08.2015	.
	14655	10.11.2015	,

Безусловно, подобная техника не является универсальной и сработает, возможно, не в 100% случаев, но является хорошей иллюстрацией возможностей Power Query.

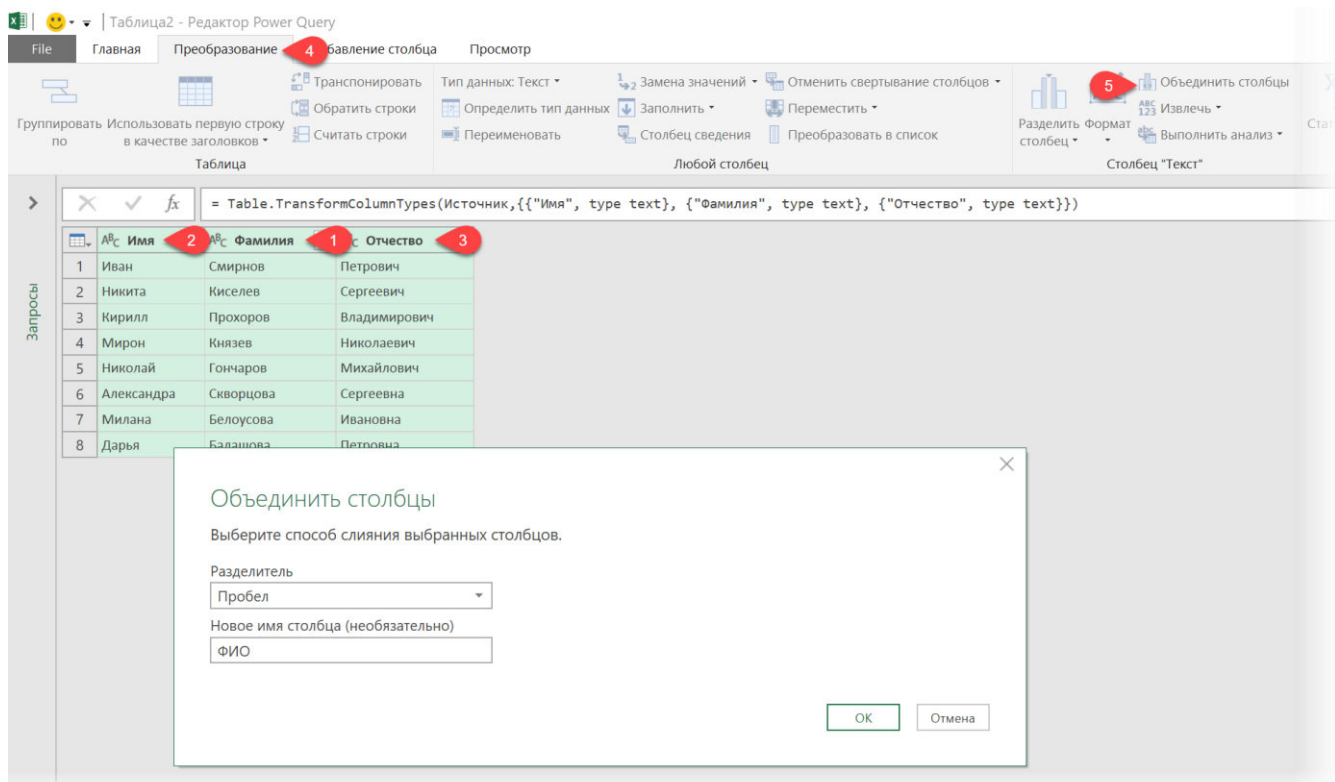
Склеивание текста

Если вам нужно не разобрать текст на столбцы, а, наоборот, склеить из нескольких столбцов один, то это легко можно сделать в Power Query разными способами.

Команда «Объединить столбцы»

Это самый простой и в большинстве случаев самый удобный способ сцепки содержимого разных столбцов. Воспользоваться им очень просто.

1. Выделите (удерживая **Ctrl**) те столбцы, содержимое которых хотите склеить в единое целое. При этом играет роль последовательность выделения: в какой последовательности вы их выделите, в том же порядке Power Query их и склеит.
2. На вкладке **Преобразование (Transform)** нажмите на кнопку **Объединить столбцы (Merge Columns)** и выберите символ-разделитель для склейки и придумайте имя новому столбцу:



После нажатия на **ОК** все наши выделенные предварительно столбцы сольются в один в том же порядке, в котором мы их выделяли:

	АВС ФИО
1	Смирнов Иван Петрович
2	Киселев Никита Сергеевич
3	Прохоров Кирилл Владимирович
4	Князев Мирон Николаевич
5	Гончаров Николай Михайлович
6	Скворцова Александра Сергеевна
7	Белуосова Милана Ивановна
8	Балашова Дарья Петровна

Склейка формулой

Если нужно сцепить много фрагментов через разные разделители или добавить к исходному тексту что-то ещё, то удобнее осуществить склейку напрямую, добавив к загруженным в Power Query данным столбец с формулой

через **Добавить столбец** → **Настраиваемый столбец** (Add Column → Custom Column). Как и в самом Excel, здесь для сцепки можно использовать символ "&" (амперсанд), а любые добавляемые текстовые фрагменты должны быть заключены в кавычки:

	Имя	Должность	Место	Склейка
1	Пупкин Василий	менеджер	Москва	Пупкин Василий, менеджер - г.Москва
2	Трутнев Сергей	стажер	Самара	Трутнев Сергей, стажер - г.Самара
3	Шестопалов Пётр	лаборант	Тамбов	Шестопалов Пётр, лаборант - г.Тамбов

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

Доступные столбцы:
 Имя
 Должность
 Место

Неочевидный момент здесь в том, что Power Query, в отличие от Excel, не умеет сцеплять данные разного типа, т. е. при попытке склеить такой формулой текстовый и числовой столбцы или дату с текстом мы получим сообщение об ошибке:

	Товар	Количество	Склейка
1	Телевизор	3	Error
2	Холодильник	5	Error
3	Пылесос	2	Error

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

[Сведения о формулах Power Query](#)

✓ Синтаксические ошибки не обнаружены.

⚠ Expression.Error: Не удастся применить оператор & к типам Text и Number.
 Сведения:
 Operator=&
 Left=Телевизор -
 Right=3

Чтобы обойти это ограничение, нужно использовать функции преобразования типов, чтобы привести всё к одному общему знаменателю – тексту. Подробнее об этом – в следующих главах.

Склеивание текста и чисел

Для преобразования чисел в текст можно использовать функцию **Text.From**, указав в качестве её аргумента столбец с числовыми данными для склейки:

	АВс Товар	123 Количество	АВс 123 Сцепка
1	Телевизор		3 Телевизор - 3 шт.
2	Холодильник		5 Холодильник - 5 шт.
3	Пылесос		2 Пылесос - 2 шт.

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

```
= [Товар] & " - " & Text.From([Количество]) & " шт."
```

Доступные столбцы:

- Товар
- Количество

Если же нужно настраивать представление числа в виде текста более детально, т. е. выводить число с заданным количеством знаков после запятой или, например, в формате процентов, то лучше использовать функцию **Number.ToText**:

	1.2 Скидка	АВс 123 Сцепка
1	0,1	Продажа со скидкой 10,00%
2	0,05	Продажа со скидкой 5,00%
3	0,15	Продажа со скидкой 15,00%

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

```
= "Продажа со скидкой " & Number.ToText([Скидка], "P")
```

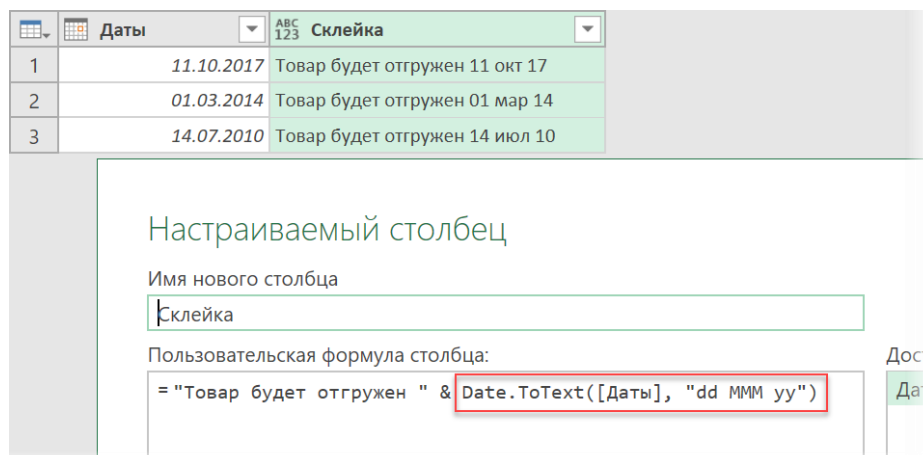
Первый аргумент этой функции – это колонка с числовыми данными, а второй – это код формата, который мы хотим получить на выходе:

	1.2 Числа	АВс 123 F	АВс 123 N	АВс 123 E	АВс 123 P
1	10	10,00	10,00	1,000000E+001	1 000,00%
2	-10	-10,00	-10,00	-1,000000E+001	-1 000,00%
3	12,34568	12,35	12,35	1,234568E+001	1 234,57%
4	-12,34352	-12,34	-12,34	-1,234352E+001	-1 234,35%
5	152200	152200,00	152 200,00	1,522000E+005	15 220 000,00%
6	0,23659	0,24	0,24	2,365900E-001	23,66%

- "F" – числовой формат с фиксированным (2) количеством разрядов после запятой
- "N" – числовой формат с тысячными разделителями и двумя разрядами после запятой
- "E" – экспоненциальный формат
- "P" – процентный формат (умножение на 100) с двумя разрядами после запятой и символом процента

Склеивание текста и дат

Аналогичным образом можно склеить с текстом и дату, если предварительно преобразовать её в текст с помощью функции **Date.ToText**:



Вторым (необязательным) аргументом этой функции может быть строка, задающая внешний вид (формат) даты, где можно использовать буквы "d", "M" и "y" (соблюдая регистр!) в разных вариациях:

- **d** – день месяца (одно- или двузначное число)
- **dd** – день месяца (двузначное число)
- **ddd** – день недели сокращенно (пн)
- **dddd** – день недели полностью (понедельник)
- **M** – номер месяца (одно- или двузначное число)
- **MM** – номер месяца (двузначное число)
- **MMM** – название месяца сокращенно (янв)
- **MMMM** – название месяца полностью (январь)
- **yy** или **yyyy** – двузначный или четырехзначный год

Массовая склейка функцией Text.Combine

Чтобы окончательно завершить разговор о различных способах склейки текста в Power Query, осталось упомянуть функцию языка M, которая называется **Text.Combine**. Эту функцию удобно использовать, когда вам нужно сцепить между собой все элементы списка (List) через заданный разделитель.

Рассмотрим в качестве примера задачу, похожую на те, что мы разбирали в главе [Группировка с выводом всех значений](#). Предположим, что нам нужно склеить через дефис все города по каждому маршруту:

	A	B	C	D	E	F	G	H
1	Авто	Город						
2	Машина 1	Москва						
3	Машина 1	Подольск						
4	Машина 1	Серпухов						
5	Машина 2	Москва						
6	Машина 2	Зеленоград						
7	Машина 2	Клин						
8	Машина 2	Тверь				Авто	Маршрут	
9	Машина 3	Москва				Машина 1	Москва-Подольск-Серпухов	
10	Машина 3	Электросталь				Машина 2	Москва-Зеленоград-Клин-Тверь	
11	Машина 3	Орехово-Зуево				Машина 3	Москва-Электросталь-Орехово-Зуево	
12	Машина 4	Москва				Машина 4	Москва-Одинцово-Наро-Фоминск-Обнинск-Малоярославец-Кондрово	
13	Машина 4	Одинцово				Машина 5	Москва-Домодедово-Михнево-Ступино	
14	Машина 4	Наро-Фоминск						
15	Машина 4	Обнинск						
16	Машина 4	Малоярославец						
17	Машина 4	Кондрово						
18	Машина 5	Москва						
19	Машина 5	Домодедово						
20	Машина 5	Михнево						
21	Машина 5	Ступино						
22								

Если помните, эту проблему решает команда **Группировать по** на вкладке **Преобразование** (Transform → Group by) с последующим выбором в качестве операции варианта **Все строки** (All rows):

АВС	Авто	АВС	Город
1	Машина 1		Москва
2	Машина 1		Подольск
3	Машина 1		Серпухов
4	Машина 2		Москва
5	Машина 2		Зеленоград
6	Машина 2		Клин
7	Машина 2		Тверь
8	Машина 3		Москва
9	Машина 3		Электросталь
10	Машина 3		Орехово-Зуево
11	Машина 4		Москва
12	Машина 4		Одинцово
13	Машина 4		Наро-Фоминск
14	Машина 4		Обнинск
15	Машина 4		Малоярославец
16	Машина 4		Кондрово
17	Машина 5		Москва

Группировать по

Базовый Подробнее

Укажите столбец для группировки и желаемые выходные данные.

Группировка: Авто

Имя нового столбца: Сборка Операция: Все строки Столбец:

OK Отмена

После группировки мы получаем в каждой ячейке нового столбца **Сборка** вложенные таблицы, куда собраны все строки по каждому маршруту. Далее, если помните, мы выдергивали из сформированных таблиц столбцы **Город** и превращали их в списки (List) функцией **Table.Column**, а затем вручную разворачивали эти списки через заданный разделитель.

При помощи функции **Text.Combine** можно немного упростить этот процесс.

Добавим к нашим данным столбец командой **Добавление столбца** → **Настраиваемый столбец** (Add Column → Custom Column) и введём в него вот такую формулу:

АВС	Авто	Сборка	АВС	Города
1	Машина 1	Table	123	Москва-Подольск-Серпухов
2	Машина 2	Table		Москва-Зеленоград-Клин-Тверь
3	Машина 3	Table		Москва-Электросталь-Орехово-Зуево
4	Машина 4	Table		Москва-Одинцово-Наро-Фоминск-Обнинск-Малоярославец-Кондрово
5	Машина 5	Table		Москва-Домодедово-Михнево-Ступино

Настраиваемый столбец

Имя нового столбца: Города

Пользовательская формула столбца:

```
=Text.Combine(Table.Column([Сборка], "Город"), "-")
```

Доступные столбцы: Авто, Сборка

<< Вставить

Сведения о формулах Power Query

✓ Синтаксические ошибки не обнаружены.

OK Отмена

Авто	Город
Машина 2	Москва
Машина 2	Зеленоград
Машина 2	Клин
Машина 2	Тверь

```
=Text.Combine(Table.Column([Сборка], "Город"), "-")
```

Уже знакомая нам функция **Table.Column** извлекает из таблиц в столбце **Сборка** колонку **Город** и возвращает её содержимое в виде списка, а потом функция **Text.Combine** склеивает всё полученные города через заданный разделитель (дефис).

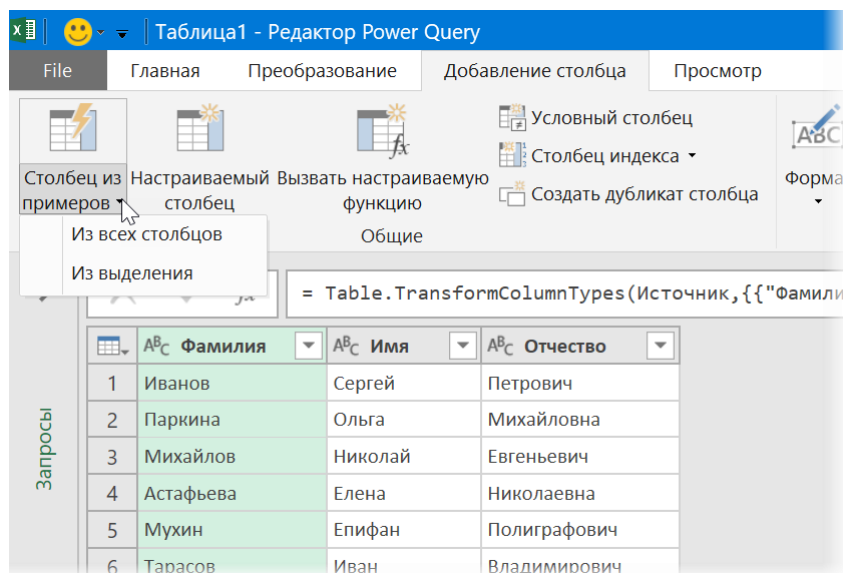
Столбец из примеров

Возможно, вам известна функция **Мгновенное заполнение (Flash Fill)**, появившаяся в Microsoft Excel начиная с 2013 версии. Суть её в том, что если вам надо как-то преобразовать ваши исходные текстовые данные, то достаточно просто начать набирать в соседнем с ними столбце тот результат, который вы хотите получить. После нескольких вручную набранных ячеек (обычно хватает 2–3) Excel вычислит итоговый паттерн и автоматически продолжит набранное, завершив всю монотонную работу за вас.

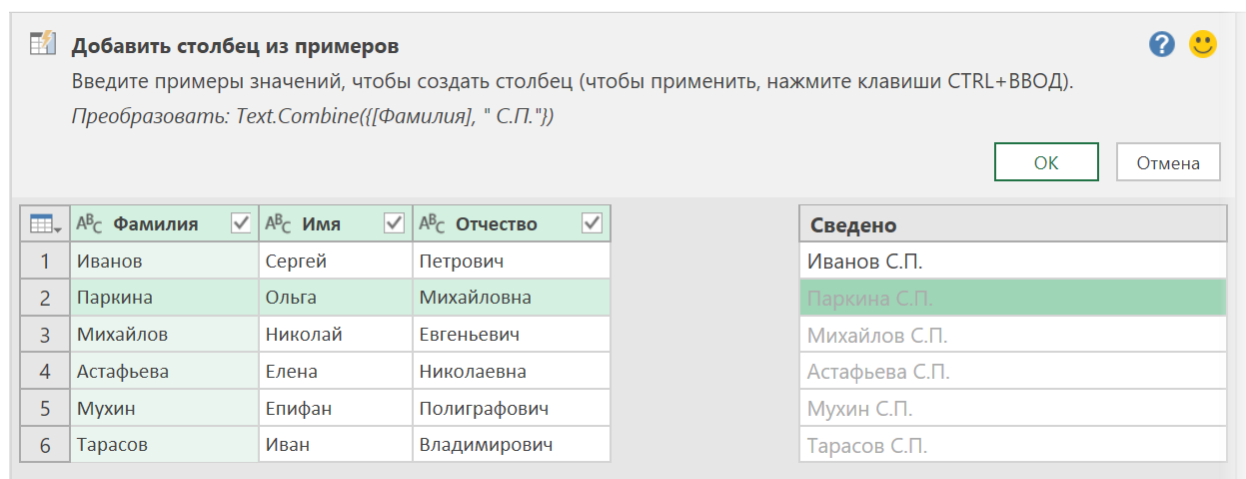
	A	B	C	D
1	Фамилия	Имя	Отчество	
2	Иванов	Сергей	Петрович	Иванов С.П.
3	Паркина	Ольга	Михайловна	Паркина О.М.
4	Михайлов	Николай	Евгеньевич	Михайлов Н.Е.
5	Астафьева	Елена	Николаевна	Астафьева Е.Н.
6	Мухин	Епифан	Полиграфович	Мухин Е.П.
7	Тарасов	Иван	Владимирович	Тарасов И.В.

Весьма приятная фишка, не правда ли?

В Power Query существует нечто похожее – инструмент, который называется **Столбец из примеров (Column From Templates)** на вкладке **Добавление столбца (Add Column)**:



Выберем опцию **Из всех столбцов (From all columns)**. В правой части окна появится пустой столбец, куда надо вручную начать набирать желаемый результат. Причем, возможно, одного введенного значения Power Query будет недостаточно для выявления закономерности. После заполнения первой ячейки программа будет думать, что мы хотим приклеить ко всем фамилиям символы "С.П.":



Но если не останавливаться и продолжить набирать, то уже после второй ячейки (Паркина О.М.) Power Query догадается, что мы хотели брать именно инициалы:

Добавить столбец из примеров

Введите примеры значений, чтобы создать столбец (чтобы применить, нажмите клавиши CTRL+ВВОД).
 Преобразовать: Text.Combine({[Фамилия], " ", Text.Start([Имя], 1), ".", Text.Start([Отчество], 1), "."})

OK Отмена

	АВс Фамилия	АВс Имя	АВс Отчество	Пользовательская
1	Иванов	Сергей	Петрович	Иванов С.П.
2	Паркина	Ольга	Михайловна	Паркина О.М.
3	Михайлов	Николай	Евгеньевич	Михайлов Н.Е.
4	Астафьева	Елена	Николаевна	Астафьева Е.Н.
5	Мухин	Епифан	Полиграфович	Мухин Е.П.
6	Тарасов	Иван	Владимирович	Тарасов И.В.

Обратите внимание, как меняется формула, по которой идёт преобразование. Если после первой ячейки это была простая склейка фамилии и "С.П." уже знакомой нам функцией **Text.Combine**:

```
Text.Combine({[Фамилия], "С.П."})
```

то после второй ячейки с примером формула уже извлекает первые символы из имени и отчества с помощью функции **Text.Start** и склеивает их потом с пробелом и точками функцией **Text.Combine**:

```
Text.Combine({[Фамилия], " ", Text.Start([Имя], 1), ".", Text.Start([Отчество], 1), "."})
```

Если результат нас устроил, то достаточно нажать на кнопку **OK** или сочетание клавиш **Ctrl+Enter** – и созданный на основе примеров столбец добавится к нашим данным:

	АВс Фамилия	АВс Имя	АВс Отчество	АВс Пользовательская
1	Иванов	Сергей	Петрович	Иванов С.П.
2	Паркина	Ольга	Михайловна	Паркина О.М.
3	Михайлов	Николай	Евгеньевич	Михайлов Н.Е.
4	Астафьева	Елена	Николаевна	Астафьева Е.Н.
5	Мухин	Епифан	Полиграфович	Мухин Е.П.
6	Тарасов	Иван	Владимирович	Тарасов И.В.

Надо понимать, что, несмотря на удобный user-friendly интерфейс, позволяющий просто вводить желаемый результат и не думать о формулах, в исходный код запроса записывается именно формула. Так, например, если мы щелкнем по значку настройки (шестерёнка) справа от этого шага, то увидим классическое окно редактирования формулы в добавленном настраиваемом столбце:

Елена	Николаевна	Астафьева Е.Н.
Епифан	Полиграфович	Мухин Е.П.
Иван	Владимирович	Тарасов И.В.

Настраиваемый столбец

Имя нового столбца:

Пользовательская формула столбца:
 =Text.Combine({[Фамилия], " ", Text.Start([Имя], 1), ".", Text.Start([Отчество], 1), "."})

Доступные столбцы:
 Фамилия
 Имя
 Отчество

Кроме приведённого выше сценария, где нужно склеивать колонки текста, инструмент **Столбец из примеров** успешно справляется и с их разбором. Представим, что нам надо извлечь год выхода фильма из списка Топ-25 фильмов с сайта «КиноПоиск»¹:

Добавить столбец из примеров
Введите примеры значений, чтобы создать столбец (чтобы применить, нажмите клавиши CTRL+ВВОД).
Преобразовать: Text.BetweenDelimiters([Фильм], "(", ")")

OK Отмена

АВС Фильм	Текст между разделителями
1 2001 год: Космическая одиссея 2001: A Space Odyssey (1968) 149 мин. Великобритания... реж. Стэнли Кубрик (фантастика, приключения) Кир...	1968
2 Чужой Alien (1979) 116 мин. Великобритания... реж. Ридли Скотт (ужасы, фантастика, триллер) Сигурни Уивер, Том Скеррит	1979
3 Всё о Еве All About Eve (1950) 133 мин. США, реж. Джозеф Лео Манкевич (драма) Бетт Дэвис, Энн Бакстер	1950
4 Амадей Amadeus (1984) 153 мин. США... реж. Милош Форман (драма, биография, история...) Том Халс, Ф. Мюррэй Абрахам	1984
5 Амели Le Fabuleux destin d'Amélie Poulain (2001) 122 мин. Франция... реж. Жан-Пьер Жёне (мелодрама, комедия) Одри Тоту, Матьё Кассовиц	2001
6 Американские граффити American Graffiti (1973) 110 мин. США, реж. Джордж Лукас (драма, комедия) Ричард Дрейфусс, Рон Ховард	1973
7 Энни Холл Annie Hall (1977) 93 мин. США, реж. Вуди Аллен (мелодрама, комедия) Вуди Аллен, Дайан Китон	1977
8 Квартира The Apartment (1960) 125 мин. США, реж. Билли Уайлдер (драма, мелодрама, комедия) Джек Леммон, Ширли МакЛейн	1960
9 Апокалипсис сегодня Apocalypse Now (1979) 194 мин. США, реж. Фрэнсис Форд Coppola (драма, военный) Марлон Брандо, Мартин Шин	1979

После ввода двух первых дат Power Query успешно сообразит, что именно мы имели в виду, и предложит формулу для извлечения текста между двумя заданными символами-разделителями "(" и ")" с функцией **Text.BetweenDelimiters**:

```
=Text.BetweenDelimiters([Фильм], "(", ")")
```

Если же нам понадобится вытащить из описания жанр, то это тоже не потребует много времени: достаточно будет ввести для примера жанр в первую же ячейку:

Добавить столбец из примеров
Введите примеры значений, чтобы создать столбец (чтобы применить, нажмите клавиши CTRL+ВВОД).
Преобразовать: Text.BetweenDelimiters([Фильм], "(", ")", 1, 0)

OK Отмена

АВС Фильм	Текст между разделителями
1 2001 год: Космическая одиссея 2001: A Space Odyssey (1968) 149 мин. Великобритания... реж. Стэнли Кубрик (фантастика, приключения) Кир...	фантастика, приключения
2 Чужой Alien (1979) 116 мин. Великобритания... реж. Ридли Скотт (ужасы, фантастика, триллер) Сигурни Уивер, Том Скеррит	ужасы, фантастика, триллер
3 Всё о Еве All About Eve (1950) 133 мин. США, реж. Джозеф Лео Манкевич (драма) Бетт Дэвис, Энн Бакстер	драма
4 Амадей Amadeus (1984) 153 мин. США... реж. Милош Форман (драма, биография, история...) Том Халс, Ф. Мюррэй Абрахам	драма, биография, история...
5 Амели Le Fabuleux destin d'Amélie Poulain (2001) 122 мин. Франция... реж. Жан-Пьер Жёне (мелодрама, комедия) Одри Тоту, Матьё Кассовиц	мелодрама, комедия
6 Американские граффити American Graffiti (1973) 110 мин. США, реж. Джордж Лукас (драма, комедия) Ричард Дрейфусс, Рон Ховард	драма, комедия
7 Энни Холл Annie Hall (1977) 93 мин. США, реж. Вуди Аллен (мелодрама, комедия) Вуди Аллен, Дайан Китон	мелодрама, комедия
8 Квартира The Apartment (1960) 125 мин. США, реж. Билли Уайлдер (драма, мелодрама, комедия) Джек Леммон, Ширли МакЛейн	драма, мелодрама, комедия

Обратите внимание, что в этом случае к предыдущей функции добавятся ещё два необязательных аргумента, уточняющие, что мы хотим извлечь текст между второй открывающей скобкой (1) и первой после неё закрывающей (0). Нумерация разделителей начинается с нуля.

Еще одним впечатляющим примером использования этого инструмента является анализ с его помощью числовой информации.

Предположим, что в качестве исходных данных у нас есть столбец с количествами проданных товаров. Допустим также, что если количество меньше 10, то такая сделка считается розничной, если от 10 до 50, то мелким оптом, а если где-то удалось продать 50 штук или больше, то это уже крупный опт.

Если загрузить такую таблицу в Power Query и добавить наш **Столбец из примеров** с вкладки **Добавление столбца**, а потом начать вручную проставлять напротив каждого количества соответствующую ему категорию, то через несколько ячеек программа выявит алгоритм и успешно продолжит сама. Причем сверху будет хорошо видно формулу из нескольких вложенных друг в друга конструкций **if ... then ... else**:

¹ <https://www.kinopoisk.ru/top/lists/231/>

Добавить столбец из примеров ? 😊

Введите примеры значений, чтобы создать столбец (чтобы применить, нажмите клавиши CTRL+ВВОД).

Преобразовать: if [Количество] >= 50 then "опт" else if [Количество] >= 10 then "мелкий опт" else "розница"

	123 Количество	✓	Пользовательская
1	12		мелкий опт
2	7		розница
3	50		опт
4	51		опт
5	10		мелкий опт
6	93		опт
7	77		опт
8	2		розница
9	5		розница
10	60		опт
11	9		розница
12	49		мелкий опт
13	73		опт
14	22		мелкий опт
15	55		опт
16	72		опт
17	31		мелкий опт
18	53		опт
19	13		мелкий опт
20	98		опт
21	14		мелкий опт

Здорово, правда?

Безусловно, инструмент **Столбец из примеров** не является «волшебной таблеткой», и рано или поздно найдутся нестандартные ситуации, когда он спасует и не сможет правильно отработать для нас желаемое. Но в большинстве случаев его можно смело использовать как весьма удобное вспомогательное средство анализа текстовой информации – почти искусственный интеллект!

Генератор фраз декартовым произведением

*Верно определяйте слова, и вы освободите мир от половины недоразумений.
(Рене Декарт)*

В математике *декартовым произведением* множеств A и B называется множество всех пар, первая компонента которых принадлежит множеству A, а вторая компонента принадлежит множеству B. Причем элементами множеств могут быть как числа, так и текст.

В переводе на человеческий язык это означает, что если в первом множестве у нас, например, слова «синий» и «красный», а во втором множестве «свитер» и «галстук», то после декартова произведения этих двух наборов мы получим на выходе совокупность всех возможных вариантов фраз, составленных из слов обоих списков:

- Синий свитер
- Синий галстук
- Красный свитер
- Красный галстук

На практике такая необходимость может возникать, например, при составлении списков ключевых слов и фраз для интернет-рекламы и SEO-продвижения, когда нужно перебрать все возможные варианты перестановок слов в поисковом запросе:

	A	B	C	D	E	F	G	H	I
1	Набор1		Набор2		Набор3			Сведено	
2	Купить		комнату		в Москве			Купить комнату в Москве	
3	Продать		квартиру		в Ярославле			Купить комнату в Ярославле	
4	Снять		дом		в Тюмени			Купить комнату в Тюмени	
5			дачу		в Самаре			Купить комнату в Самаре	
6					в Воронеже			Купить комнату в Воронеже	
7								Купить квартиру в Москве	
8								Купить квартиру в Ярославле	
9								Купить квартиру в Тюмени	
10								Купить квартиру в Самаре	
11								Купить квартиру в Воронеже	
12								Купить дом в Москве	
13								Купить дом в Ярославле	
14								Купить дом в Тюмени	
15								Купить дом в Самаре	
16								Купить дом в Воронеже	
17								Купить дачу в Москве	
18								Купить дачу в Ярославле	
19								Купить дачу в Тюмени	
20								Купить дачу в Самаре	
21								Купить дачу в Воронеже	
22								Продать комнату в Москве	
23								Продать комнату в Ярославле	
24								Продать комнату в Тюмени	
25								Продать комнату в Самаре	
26								Продать комнату в Воронеже	
27								Продать квартиру в Москве	
28								Продать квартиру в Ярославле	
29								Продать квартиру в Тюмени	

Реализовать такое в Power Query гораздо проще, чем кажется на первый взгляд.

Для начала превратим все три набора исходных фрагментов по очереди в «умные» таблицы и загрузим их в Power Query, используя кнопку **Из таблицы/диапазона (From Table/Range)** с вкладки **Данные (Data)**.

После загрузки каждой таблицы выберем команду **Главная → Закрывать и загрузить → Закрывать и загрузить в.. (Home → Close&Load → Close&Load to...)** и затем опцию **Только создать подключение (Create Only Connection)**.

На выходе должны получиться три запроса в режиме **Только подключение** с именами **Набор1, 2 и 3** соответственно.

Теперь щелкнем правой кнопкой мыши по первому запросу и выберем команду **Ссылка (Reference)**, чтобы сделать его обновляемую копию, а затем добавим к данным дополнительный столбец через команду **Добавление столбца → Настраиваемый столбец (Add Column → Custom Column)**. В окне ввода формулы введём предельно простое выражение:

=Набор2

После нажатия на **ОК** мы увидим новый столбец, в каждой ячейке которого будет лежать вложенная таблица с фразами из второго набора:

Набор1	Пользовательская
1 Купить	Table
2 Продать	Table
3 Снять	Table

Останется развернуть всё содержимое этих вложенных таблиц с помощью кнопки с двойными стрелками в заголовке полученного столбца, и мы получим все возможные сочетания элементов из первых двух наборов:

	Набор1	Набор2
1	Купить	комнату
2	Купить	квартиру
3	Купить	дом
4	Купить	дачу
5	Продать	комнату
6	Продать	квартиру
7	Продать	дом
8	Продать	дачу
9	Снять	комнату
10	Снять	квартиру
11	Снять	дом
12	Снять	дачу

Дальше всё аналогично. Добавляем еще один вычисляемый столбец с формулой:

=Набор3

... а затем ещё раз разворачиваем вложенные таблицы – и вот у нас уже все возможные варианты перестановок фраз из трёх наборов соответственно:

	ABC 123 Набор1	ABC 123 Набор2	ABC 123 Набор3
1	Купить	комнату	в Москве
2	Купить	комнату	в Ярославле
3	Купить	комнату	в Тюмени
4	Купить	комнату	в Самаре
5	Купить	комнату	в Воронеже
6	Купить	квартиру	в Москве
7	Купить	квартиру	в Ярославле
8	Купить	квартиру	в Тюмени
9	Купить	квартиру	в Самаре
10	Купить	квартиру	в Воронеже
11	Купить	дом	в Москве
12	Купить	дом	в Ярославле
13	Купить	дом	в Тюмени
14	Купить	дом	в Самаре

Осталось выделить все три столбца и сцепить их содержимое через пробел, используя команду **Объединить столбцы** (Merge Columns) с вкладки **Преобразование** (Transform) – и задача решена!

Нечёткий текстовый поиск

— Любите ли вы Кафку?
— Оооофень! Особенно грефневую...
(Автор неизвестен)

Удачным продолжением темы про декартово произведение будет глава о реализации в Power Query нечеткого (fuzzy) текстового поиска, т. е. поиска и сопоставления неточно совпадающих текстовых строк.

Где-то есть Идеальная Вселенная, где все пользователи никогда не ошибаются при вводе данных, не пишут «исчо» и «Чилиябинск» и всегда соблюдают правила грамматики, орфографии и корпоративные стандарты. В нашей же реальности мы с вами, к сожалению, часто имеем в качестве входных данных «креатив» той или иной степени жести.

Естественно, все стандартные инструменты Power Query (слияние запросов, функции поиска и т. д.) с такими кривыми данными не работают. Для поиска наиболее похожих, но не совпадающих точно текстовых строк обычно используют макросы на Visual Basic или же специальные надстройки, выполняющие нечеткий поиск по различным алгоритмам. Однако при желании что-то подобное вполне можно реализовать и в Power Query, чуть-чуть углубившись в исходный код запросов на языке M.

Давайте посмотрим, как это можно сделать.

Шаг 1. Создаем функцию коэффициента подобия

Перво-наперво нам с вами нужно договориться о том, как мы будем мерить степень подобия двух текстовых строк. На этот счёт есть огромное количество научных методик и алгоритмов (Джакартово подобие, метод N-грамм, расстояние Левенштейна и т. д.). Мы же, чтобы чрезмерно всё не усложнять, давайте примем за основу простую формулу:

$$\text{Коэф. подобия двух строк} = \frac{\text{Количество совпадающих символов}}{\text{Средняя длина текста}}$$

Если мы сравниваем две абсолютно одинаковых строки, например «Ананас» и «Ананас», то количество совпадающих букв в этих двух словах равно пяти, и средняя длина этих двух слов тоже пять символов. Таким образом, коэффициент подобия будет $5/5 = 1$ или 100%.

Если же мы сравниваем, например, строки «Тула» и «г.Тула», то количество совпадающих символов тут только четыре, а вот средняя длина этих двух фрагментов $(4+6)/2 = 5$. И степень подобия будет уже $4/5 = 0,8$.

И так далее, я думаю, вы уже поняли принцип.

Безусловно, этот подход очень упрощён и не лишён недостатков, т. к. не чувствует разницы, например, в словах с переставленными буквами («Москва» и «Моксва»), но для большинства реальных задач вполне сойдет.

Давайте сначала создадим универсальную функцию расчета коэффициента подобия, чтобы потом использовать её в будущем при сравнении текстов. Для этого выберем на вкладке **Данные** → **Получить данные** → **Из других источников** → **Пустой запрос** (Data → Get Data → From Other Sources → Blank Query). Сразу введём имя запроса (т. е. нашей функции), например **Fuzzy**, в панели справа, а затем откроем редактор M-кода на вкладке **Просмотр** → **Расширенный редактор** (View → Advanced Editor) и введём туда следующие команды:

```
(text1 as text, text2 as text) as number =>
let
    text1 = Text.Upper(text1),
    text2 = Text.Upper(text2),
    matching_chars = List.Count(List.Intersect({Text.ToList(text1), Text.ToList(text2)})),
    average_length = (Text.Length(text1) + Text.Length(text2)) / 2,
    coef = matching_chars / average_length
```

```
in
```

```
coef
```

Давайте разберём этот код построчно.

Сначала мы объявляем, что наша функция будет возвращать на выходе числовой (**number**) результат – коэффициент подобия, и у функции будет два аргумента – переменные **text1** и **text2** текстового типа:

```
(text1 as text, text2 as text) as number =>
```

Чтобы избавиться от регистрочувствительности, преобразуем все наши данные в верхний регистр:

```
text1 = Text.Upper(text1),
text2 = Text.Upper(text2),
```

Конечно, это можно и не делать, если хотите, чтобы наша функция различала строчные и прописные буквы.

Разбиваем каждый текст на отдельные символы функцией **Text.ToList** и ищем совпадения в полученных наборах символов функцией **List.Intersect**. Количество найденных совпадений подсчитывается функцией **List.Count**:

```
matching_chars = List.Count(List.Intersect({Text.ToList(text1), Text.ToList(text2)})),
```

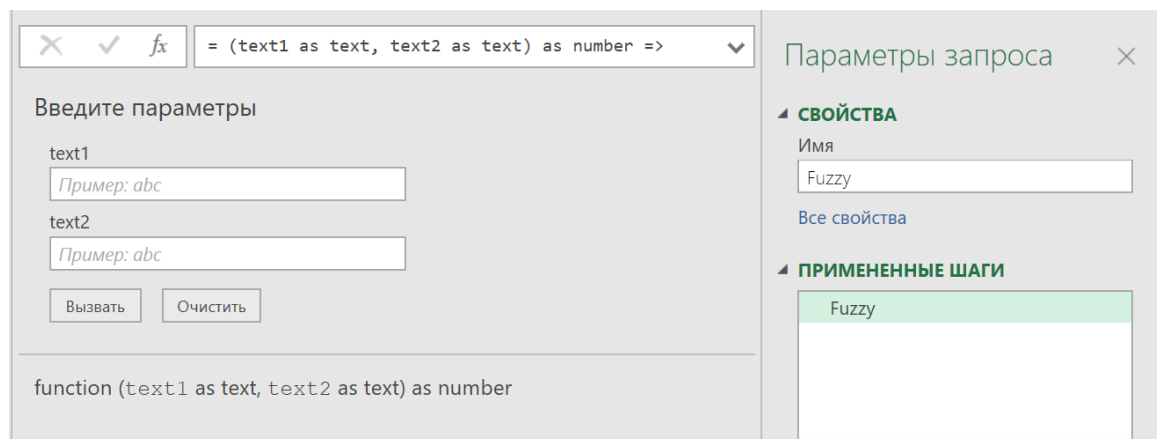
Среднюю длину наших исходных строк текста считаем просто: сумму их длин, полученную функцией **Text.Length**, делим пополам:

```
average_length = (Text.Length(text1) + Text.Length(text2)) / 2,
```

И наконец, вычисляем коэффициент подобия делением количества найденных совпадений символов на среднюю длину сравниваемых текстовых фрагментов:

```
coef = matching_chars / average_length
```

После нажатия на **OK** мы получим готовую для применения функцию **Fuzzy**:



Давайте проверим её работу на какой-нибудь тестовой табличке, сделанной на скорую руку. Например, на такой:

	А	В
1	Текст1	Текст2
2	Эвкалипт	Эвкалипт
3	Эвкалипт	Эквалипт
4	Эвкалипт	Эвкалип
5	Эвкалипт	Евкалипт
6	Эвкалипт	Ивколипт
7	Эвкалипт	Берёза
8	Эвкалипт	Ясень
9		

После загрузки таблицы в Power Query нажмём на кнопку **Вызвать настраиваемую функцию** на вкладке **Добавление столбца** (Add Column → Invoke Custom Function) и выберем затем имя функции и её аргументы в следующем окне:

The screenshot shows the Power Query Editor interface. The ribbon is set to 'Добавление столбца' (Add Column). A dialog box titled 'Вызвать настраиваемую функцию' (Invoke Custom Function) is open. The dialog contains the following fields:

- Имя нового столбца (New column name): Коэф.подобия
- Запрос функции (Function query): Fuzzy
- text1: Текст1
- text2: Текст2

Below the dialog, a preview table shows the results of the function:

Текст1	Текст2	Коэф.подобия
1 Эвкалипт	Эвкалипт	1
2 Эвкалипт	Эвкалипт	1
3 Эвкалипт	Эвкалипт	0,933333333
4 Эвкалипт	Евкалипт	0,875
5 Эвкалипт	Ивколипт	0,75
6 Эвкалипт	Берёза	0,142857143
7 Эвкалипт	Ясень	0

После нажатия на **ОК** мы получим вычисленные коэффициенты подобия для каждой пары слов в каждой строке. Беглый просмотр подтверждает, что тест пройден, наша функция Fuzzy работает вполне корректно, можно переходить к решению основной задачи.

Шаг 2. Выполняем декартово произведение списков

Следующим шагом будет уже знакомая нам магия: нужно будет загрузить оба исходных списка в Power Query как подключения (я назвал их **Таблица1** и **Таблица2**):

	A	B	C	D	E
1	Список1		Список2		
2	Олександр Пушкин		Александр Грибоедов		
3	Крылов Иван		Александр Островский		
4	антуан чехонте		Александр Пушкин		
5	Солтыков-щидрин		Антон Чехов		
6	Федор Достоефский		Денис Фонвизин		
7	Коля Некрасов		Иван Бунин		
8	тургеев		Иван Крылов		
9	Лермантов		Иван Тургенев		
10	Алек Острофский		Лев Толстой		
11	Чютчев Фёдор		Максим Горький		
12	Фанвизин Д		Михаил Зоценко		
13	Карамзин		Михаил Лермонтов		
14	гриба едов		Михаил Ломоносов		
15	ДОВЛАТОВ		Михаил Салтыков-Щедрин		
16	Зоценко михайил		Николай Гоголь		
17	Гоголь Моголь		Николай Карамзин		
18	искондер фозиль		Николай Некрасов		
19	Толстой Лев Николаевич		Сергей Довлатов		
20	Михайло Васильевич Ломоносов		Фазиль Искандер		
21	Горький М.		Федор Достоевский		
22	Бунен Ваня		Федор Тютчев		
23					
24					

Запросы и подключения

Запросы | Подключения

Запросов: 8

- Таблица1
Только подключение.
- Таблица2
Только подключение.
- Fuzzy
Только подключение.

Теперь выполним опять же уже знакомое по предыдущей главе декартово умножение этих списков друг на друга. Для этого щелкнем правой кнопкой мыши по первому запросу и выберем команду **Ссылка (Reference)**, а затем добавим вычисляемый столбец через **Добавление столбца → Настраиваемый столбец (Add Column → Custom Column)** и введём следующую формулу:

```
=Таблица2
```

После нажатия на **ОК** мы увидим новый столбец, где в каждой ячейке будет вложена таблица с данными из второго списка:

Развернем вложенные таблицы кнопкой с двойными стрелками в шапке и получим в итоге таблицу со всеми возможными сочетаниями строк из обеих таблиц:

	АВС Список1	АВС 123 Список2
1	Олександр Пушкин	Александр Грибоедов
2	Олександр Пушкин	Александр Островский
3	Олександр Пушкин	Александр Пушкин
4	Олександр Пушкин	Антон Чехов
5	Олександр Пушкин	Денис Фонвизин
6	Олександр Пушкин	Иван Бунин
7	Олександр Пушкин	Иван Крылов
8	Олександр Пушкин	Иван Тургенев
9	Олександр Пушкин	Лев Толстой
10	Олександр Пушкин	Максим Горький
11	Олександр Пушкин	Михаил Зощенко
12	Олександр Пушкин	Михаил Лермонтов
13	Олександр Пушкин	Михаил Ломоносов
14	Олександр Пушкин	Михаил Салтыков-Щедрин
15	Олександр Пушкин	Николай Гоголь
16	Олександр Пушкин	Николай Карамзин
17	Олександр Пушкин	Николай Некрасов
18	Олександр Пушкин	Сергей Довлатов
19	Олександр Пушкин	Фазиль Искандер
20	Олександр Пушкин	Федор Достоевский
21	Олександр Пушкин	Федор Тютчев
22	Крылов Иван	Александр Грибоедов
23	Крылов Иван	Александр Островский
24	Крылов Иван	Александр Пушкин
25	Крылов Иван	Антон Чехов
26	Крылов Иван	Денис Фонвизин
27	Крылов Иван	Иван Бунин
28	Крылов Иван	Иван Крылов
29	Крылов Иван	Иван Тургенев

Шаг 3. Ищем самые похожие пары

Дальше – проще. Уже знакомым образом, используя кнопку **Вызвать настраиваемую функцию** с вкладки **Добавление столбца** (Add Column → Invoke Custom Function), добавляем столбец, где наша функция **Fuzzy** посчитает коэффициент подобия для каждой созданной пары фраз:

	АВС Список1	АВС 123 Список2	АВС 123 Коэф.подобия
1	Олександр Пушкин	Александр Грибоедов	0,628571429
2	Олександр Пушкин	Александр Островский	0,611111111
3	Олександр Пушкин	Александр Пушкин	0,875
4	Олександр Пушкин	Антон Чехов	0,444444444
5	Олександр Пушкин	Денис Фонвизин	0,466666667
6	Олександр Пушкин	Иван Бунин	0,384615385
7	Олександр Пушкин	Иван Крылов	0,518518519
8	Олександр Пушкин	Иван Тургенев	0,551724138
9	Олександр Пушкин	Лев Толстой	0,37037037

Нас, конечно, интересуют только те пары, где коэффициент максимальный, поэтому дальше делаем следующее:

1. Сначала сортируем таблицу по столбцу **Список2** по возрастанию (алфавиту).
2. Затем сортируем таблицу по столбцу **Коэф.подобия** по убыванию. Пары с максимальным коэффициентом, таким образом, встанут в начало каждой группы.
3. Чтобы данные впоследствии не перемешались, добавляем столбец индекса на вкладке **Добавление столбца** → **Столбец индекса** → **От 1** (Add Column → Index Column → From 1).
4. Удаляем повторы, оставив только фразы с максимальным коэффициентом: щелкаем правой кнопкой мыши по заголовку столбца **Список2** и выбираем команду **Удалить дубликаты** (Remove duplicates).

5. Ненужный более вспомогательный столбец с индексами можно удалить.

В итоге должна получиться таблица с максимально похожими фразами, состыкованная из двух исходных списков:

	ABC 123 Список1	ABC 123 Список2	ABC 123 Коэф.подобия
1	гриба едов	Александр Грибоедов	0,689655172
2	Алек Острофский	Александр Островский	0,8
3	Олександр Пушкин	Александр Пушкин	0,875
4	антуан чехонте	Антон Чехов	0,72
5	Фанвизин Д	Денис Фонвизин	0,75
6	Бунен Ваня	Иван Бунин	0,8
7	Крылов Иван	Иван Крылов	1
8	тургеенев	Иван Тургенев	0,727272727
9	Толстой Лев Николаевич	Лев Толстой	0,666666667
10	Горький М.	Максим Горький	0,75
11	Зощенко михайил	Михаил Зощенко	0,933333333
12	Лермантов	Михаил Лермонтов	0,72
13	Михайло Васильевич Ломонос...	Михаил Ломоносов	0,727272727
14	Солтыков-щидрин	Михаил Солтыков-Щедрин	0,756756757
15	Гоголь Моголь	Николай Гоголь	0,666666667
16	Зощенко михайил	Николай Карамзин	0,75
17	Коля Некрасов	Николай Некрасов	0,827586207
18	ДОВЛАТОВ	Сергей Довлатов	0,695652174
19	искондер фозиль	Фазиль Искандер	0,866666667
20	Федор Достоефский	Федор Достоевский	0,882352941

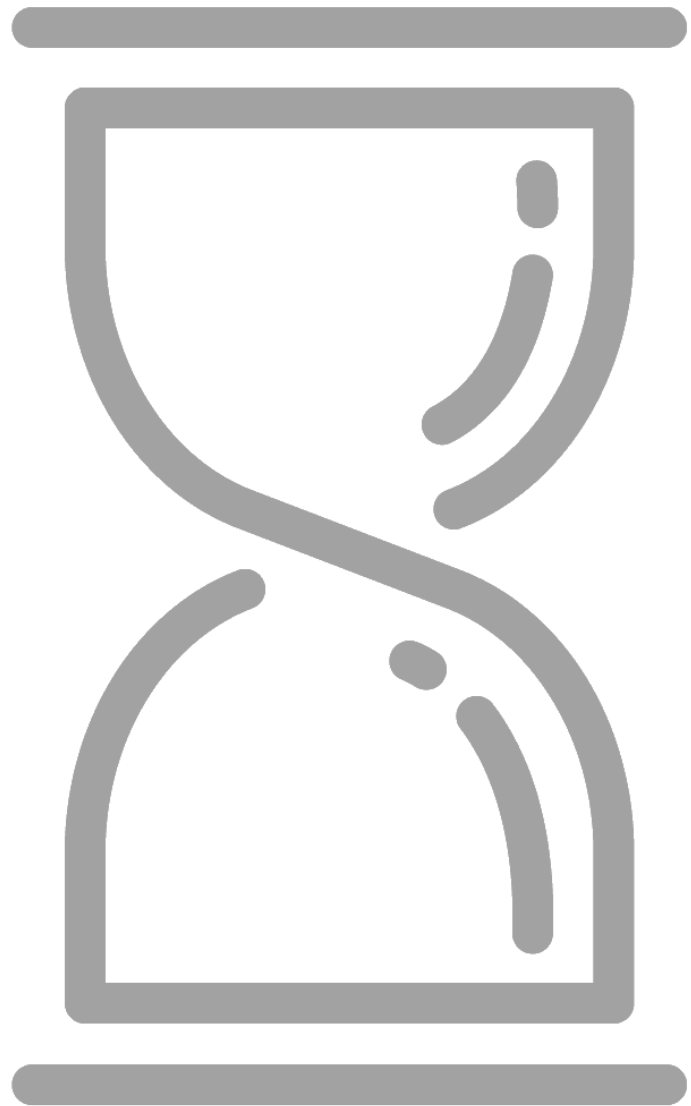
Наблюдательный читатель может заметить, что в особо запущенных случаях (строка 16, например) наш алгоритм всё же спасовал, но со всеми остальными строками справился на отлично. Для реальной жизни, как мне кажется, такой процент попадания является вполне приемлемым результатом. Особенно учитывая тот факт, что весь запрос мы сделали за считанные минуты.

Обработка дат и времени

Безусловно, даты и время будут встречаться в данных при работе с ними в Power Query. На наше с вами счастье, эта надстройка умеет замечательно с ними работать.

В этой главе мы:

- научимся **распознавать различные виды дат и времени**, превращая их в корректный тип данных, пригодный для дальнейшей работы;
- будем **извлекать из даты отдельные составляющие** (день, месяц, год, квартал, день недели и т. д.);
- разберёмся с тем, как можно определить **номер недели по дате** в Power Query;
- освоим **вычисления с датами** (расчёт длительностей, сроков, стажа, возраста и т. д.);
- научимся быстро **заполнять пробелы в датах** и находить самую **раннюю и позднюю даты** в наборе.



Распознавание дат

В отличие от текста (который всегда текст) с датами всё традиционно сложнее. В данных, которые мы с вами получаем из внешнего мира, дата может быть представлена в очень разных (и не всегда корректных) формах. Кто-то напишет месяц словом, кто-то – числом, кто-то добавит букву «г» после года и т. д. Не говоря уже о случаях, когда дата идёт в текстовом формате или изначально неправильных значениях типа «30 февраля».

Для решения подобных задач в самом Excel обычно используются функции **ДАТАЗНАЧ** (DATEVALUE)¹ или предварительная нарезка даты на составляющие (день, месяц, год) функциями **ЛЕВСИМВ** (LEFT), **ПРАВСИМВ** (RIGHT) и **ПСТР** (MID) с последующей сборкой даты обратно в уже правильном виде.

Что же может предложить Power Query в такой ситуации? Давайте посмотрим.

Формат даты для столбца

Для тестирования возьмём столбец с датами в текстовом формате и разных формах записи:

	A
1	Дата текстом
2	15.03.17
3	15.03.2017
4	15.03
5	15.03.2017 г
6	15.03.2017 г.
7	15 мар 2017
8	15/03/17
9	15 марта 2017 г.
10	15-03-2017
11	15 mar 17
12	Отгрузка 15.03.17
13	15-е марта 2017 года
14	2017.03.15
15	2017/03/15
16	03.15.2017
17	03/15/17
18	20170315
19	

Загрузим эти данные в Power Query и сделаем копию столбца, щёлкнув правой кнопкой мыши по заголовку и выбрав команду **Дублировать столбец** (Duplicate column), чтобы удобно было сравнить потом исходные данные и результат.

Теперь применим к столбцу-дубликату формат **Дата** (Date), выбрав его из выпадающего списка в шапке или на вкладке **Главная** → **Тип данных** → **Дата** (Home → Data type → Date). Тот же эффект будет, если использовать специальный инструмент для анализа (парсинга) дат и извлечения их из текста – команду **Дата** → **Выполнить анализ** (Add Column → Date Parse) на вкладках **Добавить столбец** (Add Column) или **Преобразовать** (Transform):

¹ См. главу «Преобразование текстовой даты в полноценную дату» из моей книги «Microsoft Excel: Готовые решения – бери и пользуйся».

	АВС Дата текстом	Копия Дата текстом
1	15.03.17	15.03.2017
2	15.03.2017	15.03.2017
3	15.03	15.03.2019
4	15.03.2017 г	15.03.2017
5	15.03.2017 г.	15.03.2017
6	15 мар 2017	15.03.2017
7	15/03/17	15.03.2017
8	15 марта 2017 г.	15.03.2017
9	15-03-2017	15.03.2017
10	15 mar 17	15.03.2017
11	Отгрузка 15.03.17	Error
12	15-е марта 2017 года	Error
13	2017.03.15	15.03.2017
14	2017/03/15	15.03.2017
15	03.15.2017	Error
16	03/15/17	Error
17	20170315	15.03.2017

Давайте внимательно посмотрим на полученную таблицу и сделаем выводы:

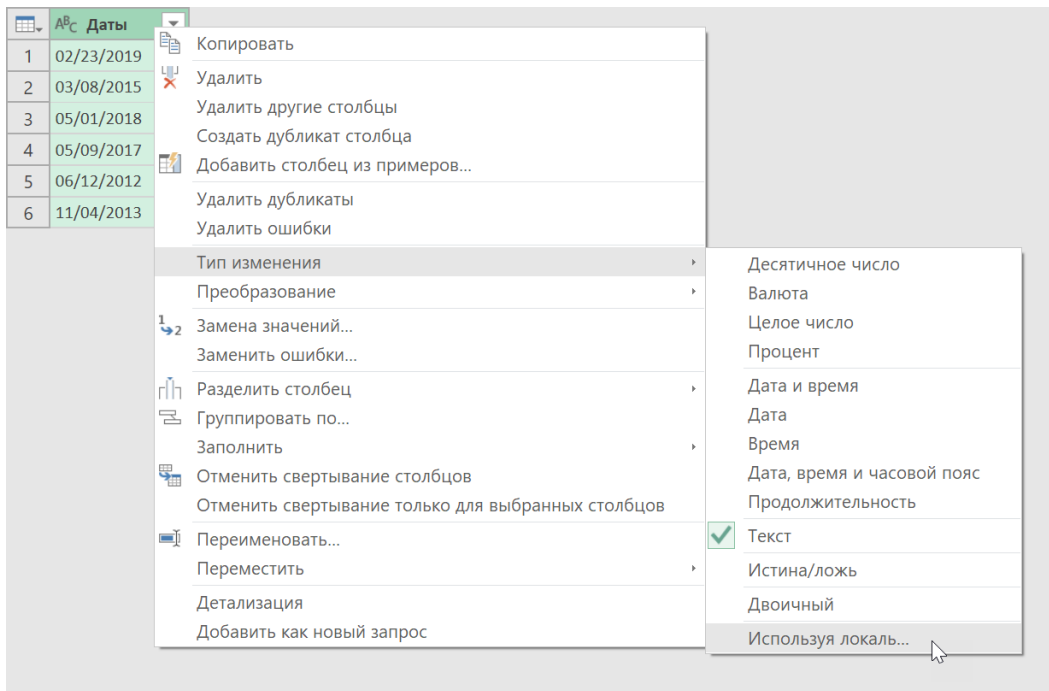
- Почти любые текстовые формы записи даты распознаются стандартными средствами Power Query совершенно замечательно. Он справился и с «г» после года, и с разными вариантами текстового написания месяца («мар», «марта», «mar»).
- Если дата была написана в обратной последовательности (год-месяц-день), то она тоже отлично распознается, даже если записана вообще без разделителей (20170315).
- Не распознаются даты, если они смешаны с каким-то посторонним текстом («Отгрузка 15.03.2017»).
- Не распознаются даты, если они идут в другом региональном формате, например в американском формате «месяц-день-год», если ваш компьютер при этом имеет стандартные российские региональные настройки. И наоборот, в США не будут корректно распознаваться «русские» даты в привычном нам виде «день-месяц-год».

По-моему, совершенно замечательный КПД. И это учитывая, что мы пока не использовали никаких специальных инструментов или хитрых формул, а всего лишь поменяли формат данных столбца.

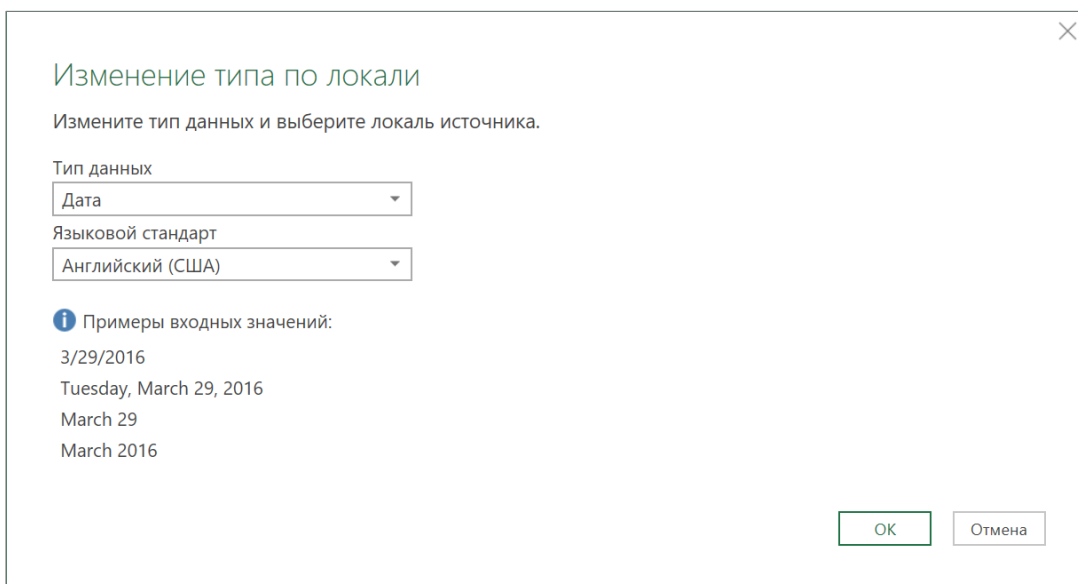
Использование локали для дат других стран

Напомню, что термином *локаль* обозначается набор региональных настроек, соответствующих определенной стране. Сюда входят форматы записи чисел, дат, времени, официальные единицы измерения, национальная валюта и т. д.

Если вам достались даты в несовпадающей с настройками вашего компьютера локали, например «американские» даты вида ММ.ДД.ГГГГ на компьютере с «русскими» настройками даты в виде ДД.ММ.ГГГГ, то для их исправления нужно будет воспользоваться командой **Используя локаль (Use locals)** в выпадающем списке форматов данных или в контекстном меню столбца в пункте **Тип изменения (Changing type)**:



В последующем окне нужно выбрать тип данных в колонке и страну. По моему опыту, в подавляющем большинстве некорректно распознаваемых случаев стоит начать с варианта **Английский (США)**:



Он же поможет исправить числа с нестандартными разделителями, отрицательные числа в скобках, минусы на конце и т. д.

Региональные настройки, используемые Power Query по умолчанию при автоматическом распознавании чисел, дат и времени для данного файла, можно поменять через **Данные → Получить данные → Параметры запроса → Текущая книга → Региональные настройки** (Data → Get Data → Query settings → Current file → Regional settings).

Столбцы с датами смешанного формата

Чтобы закончить разговор о распознавании различных типов даты, давайте рассмотрим неприятный случай, когда в одном столбце у нас встречаются даты разных локалей – и в формате ДД.ММ.ГГГГ, и в формате ММ.ДД.ГГГГ.

	ABC Товар	123 Цена	ABC Страна	ABC Дата поставки
1	Кардиган	8800	RU	25/02/2019
2	Кофта	1300	US	04/30/2019
3	Бриджи	7500	US	02/25/2019
4	Жакет	9700	RU	21/01/2019
5	Сарафан	5000	RU	12/02/2019
6	Джинсы	3900	US	01/19/2019
7	Топ	9900	RU	23/02/2019

Предположим, что мы создали в Power Query запрос, который собирает общий итоговый прайс из прайс-листов нескольких поставщиков. Предположим также, что кто-то из поставщиков был из России, а кто-то – из США, поэтому на выходе мы имеем столбец с датами в разных форматах вперемешку.

Давайте сначала разделим дату на три составляющих: день, месяц и год. Для этого выделим столбец **Дата поставки** и выберем команду **Преобразовать → Разделить столбец → По разделителю** (Transform → Split Column → By delimiter). Зададим соответствующий символ-разделитель и выберем опцию **По каждому вхождению разделителя**. В итоге из одного столбца с датой получим три.

Следующим шагом мы должны собрать дату обратно, но уже в правильном порядке. Для этого добавим к нашей таблице вычисляемый столбец через вкладку **Добавление столбца → Настраиваемый столбец** (Add Column → Custom Column) и введём в открывшемся окне следующую формулу:

```
=if [Страна]="RU" then
    #date([Дата поставки.3],[Дата поставки.2],[Дата поставки.1])
else
    #date([Дата поставки.3],[Дата поставки.1],[Дата поставки.2])
```

Используемая здесь функция **#date** является аналогом экселевской функции **ДАТА (DATE)** и формирует полноценную дату из трёх аргументов: номера года, месяца и дня. После нажатия на **ОК** мы получим столбец с исправленными датами:

Цена	ABC Страна	123 Дата поставки.1	123 Дата поставки.2	123 Дата поставки.3	ABC Дата
8800	RU	25	2	2019	25.02.2019
1300	US	4	30	2019	30.04.2019
7500	US	2	25	2019	25.02.2019
9700	RU	21	1	2019	21.01.2019
5000	RU				
3900	US				
9900	RU				

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

```
= if [Страна]="RU" then #date([Дата поставки.3],[Дата поставки.2],[Дата поставки.1]) else #date([Дата поставки.3],[Дата поставки.1],[Дата поставки.2])
```

Доступные столбцы:
Товар
Цена
Страна
Дата поставки.1
Дата поставки.2
Дата поставки.3

<< Вставить

Сведения о формулах Power Query

✔ Синтаксические ошибки не обнаружены.

Останется удалить ненужные более колонки **Дата поставки 1,2,3** – и наша задача решена.

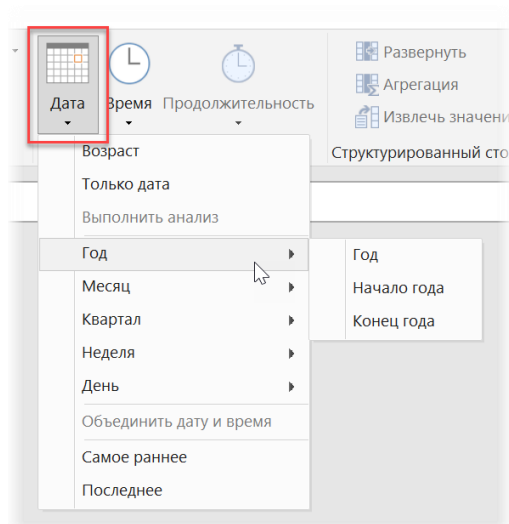
Преобразование дат

В Microsoft Excel есть куча встроенных функций, позволяющих извлекать из даты только нужный нам фрагмент (**ГОД**, **МЕСЯЦ**, **ДЕНЬ**, **ДЕНЬНЕД**...) или трансформирующих дату в нужный нам вид (**ТЕКСТ**). Однако для этого придется создавать новый столбец в таблице, вводить в него соответствующую формулу, копировать её, затем, скорее всего, заменять формулы на значения, чтобы избавиться от ссылок, и т. д.

Power Query может легко помочь в подобных задачах. За это отвечают кнопки **Дата** (**Date**), находящиеся на вкладках **Преобразование** (**Transform**) и **Добавление столбца** (**Add Column**).

Как и при обработке текста, разница в работе этих одинаковых на вид кнопок в том, что **Дата** на вкладке **Преобразование** изменяет столбец на месте, а такая же кнопка на вкладке **Добавление столбца** сохранит исходную колонку с датами и добавит к нашей таблице ещё один столбец с преобразованными значениями.

Давайте пробежимся по набору возможностей, которые предоставляет Power Query при анализе дат.



Раздел **Год** (**Year**)

- **Год** (**Year**) – четырёхзначный номер года в числовом формате.
- **Начало года** (**Start of Year**) – дата первого дня (1 января) текущего года.
- **Конец года** (**End of Year**) – Power Query при использовании этой команды почему-то выдает не интуитивно ожидаемую дату последнего дня (31 декабря) года, а дату первого дня года следующего. Если нужно получить именно 31 декабря, то дополнительно придется вычесть ещё один день, используя команду **Добавление столбца** → **Настраиваемый столбец** (**Add Column** → **Custom Column**) и введя затем формулу:

```
=[Конец года] - #duration(1,0,0,0)
```

Выражение **#duration(1,0,0,0)** здесь означает, что мы вычитаем интервал времени, равный одному дню, ноль часов, ноль минут и ноль секунд.

	Дата продажи	Год	Начало года	Конец года	Конец года 2
1	25.06.2012 6:52:18	2012	01.01.2012 0:00:00	01.01.2013 0:00:00	31.12.2012 0:00:00
2	11.07.2017 6:04:33	2017	01.01.2017 0:00:00	01.01.2018 0:00:00	31.12.2017 0:00:00
3	29.05.2016 3:13:47	2016	01.01.2016 0:00:00	01.01.2017 0:00:00	31.12.2016 0:00:00
4	27.04.2018 4:28:52	2018	01.01.2018 0:00:00	01.01.2019 0:00:00	31.12.2018 0:00:00
5	14.10.2013 5:31:54	2013	01.01.2013 0:00:00	01.01.2014 0:00:00	31.12.2013 0:00:00
6	24.08.2018 7:04:15	2018	01.01.2018 0:00:00	01.01.2019 0:00:00	31.12.2018 0:00:00

Раздел **Месяц** (**Month**)

- **Месяц** (**Month**) – номер месяца числом от 1 до 12.
- **Начало месяца** (**Start of Month**) – дата первого дня текущего месяца и года.
- **Конец месяца** (**End of Month**) – как и в предыдущем пункте, здесь мы получим не дату последнего дня текущего месяца, а дату первого дня следующего. Аналогично предыдущему пункту можно вычесть один день, чтобы получить желаемое.
- **Дней в месяце** (**Days in Month**) – количество дней в месяце числом от 28 до 31.
- **Название месяца** (**Name of Month**) – название месяца словом.

	Дата продажи	1 ² Месяц	Начало месяца	Конец месяца	1 ² Дней в месяце	A ^B C Название месяца
1	25.06.2012 6:52:18	6	01.06.2012 0:00:00	01.07.2012 0:00:00	30	Июнь
2	11.07.2017 6:04:33	7	01.07.2017 0:00:00	01.08.2017 0:00:00	31	Июль
3	29.05.2016 3:13:47	5	01.05.2016 0:00:00	01.06.2016 0:00:00	31	Май
4	27.04.2018 4:28:52	4	01.04.2018 0:00:00	01.05.2018 0:00:00	30	Апрель
5	14.10.2013 5:31:54	10	01.10.2013 0:00:00	01.11.2013 0:00:00	31	Октябрь
6	24.08.2018 7:04:15	8	01.08.2018 0:00:00	01.09.2018 0:00:00	31	Август

Раздел **Квартал (Quarter)**

- **Квартал года (Quarter of Year)** – номер квартала в году числом от 1 до 4. Если необходимо отображать квартал как "Q1" или "1 кв.", то можно легко добавить к числу нужный текст, используя команду **Преобразование → Формат → Добавить префикс/суффикс (Transform → Format → Add Prefix/Suffix)**.
- **Начало квартала (Start of Quarter)** – дата первого дня квартала.
- **Конец квартала (End of Quarter)** – дата первого дня следующего квартала, но можно вычесть один день, чтобы получить именно последний день текущего, как мы делали ранее.

	Дата продажи	1 ² Квартал	A ^B C Префикс	Начало квартала	Конец квартала	A ^B C 123 Конец квартала 2
1	25.06.2012 6:52:18	2	Q2	01.04.2012 0:00:00	01.07.2012 0:00:00	30.06.2012 0:00:00
2	11.07.2017 6:04:33	3	Q3	01.07.2017 0:00:00	01.10.2017 0:00:00	30.09.2017 0:00:00
3	29.05.2016 3:13:47	2	Q2	01.04.2016 0:00:00	01.07.2016 0:00:00	30.06.2016 0:00:00
4	27.04.2018 4:28:52	2	Q2	01.04.2018 0:00:00	01.07.2018 0:00:00	30.06.2018 0:00:00
5	14.10.2013 5:31:54	4	Q4	01.10.2013 0:00:00	01.01.2014 0:00:00	31.12.2013 0:00:00
6	24.08.2018 7:04:15	3	Q3	01.07.2018 0:00:00	01.10.2018 0:00:00	30.09.2018 0:00:00

Раздел **Неделя (Week)**

- **Неделя года (Week of Year)** – номер недели в году. Обратите внимание, что Power Query использует нумерацию недель по американскому стандарту, который отличается от принятого в России и Европе стандарта ISO 8601. Если вам нужна нумерация именно по ISO, то придется немного «пошаманить», что мы и сделаем в следующей главе.
- **Неделя месяца (Week of Month)** – порядковый номер (начиная с 1) недели в месяце, куда попадает текущая дата.
- **Начало недели (Start of Week)** – дата предыдущего понедельника.
- **Конец недели (End of Week)** – дата следующего понедельника.

	Дата продажи	1 ² Неделя года	1 ² Неделя месяца	Начало недели	Конец недели
1	25.06.2012 6:52:18	27	5	25.06.2012 0:00:00	02.07.2012 0:00:00
2	11.07.2017 6:04:33	29	3	10.07.2017 0:00:00	17.07.2017 0:00:00
3	29.05.2016 3:13:47	22	5	23.05.2016 0:00:00	30.05.2016 0:00:00
4	27.04.2018 4:28:52	17	5	23.04.2018 0:00:00	30.04.2018 0:00:00
5	14.10.2013 5:31:54	42	3	14.10.2013 0:00:00	21.10.2013 0:00:00
6	24.08.2018 7:04:15	34	4	20.08.2018 0:00:00	27.08.2018 0:00:00

Раздел **День (Day)**

- **День (Day)** – номер дня в месяце.
- **День недели (Day of Week)** – порядковый номер дня недели, начиная с нуля, т. е. Пн=0, Вт=1 и т. д. При необходимости можно привести нумерацию к более привычному виду, добавив к полученным числам единицу командой **Преобразование → Стандартные → Сложить (Transform → Standard → Add)**.
- **День года (Day of Year)** – порядковый номер дня в году.
- **Начало дня (Start of Day)** – дата-время начала дня (полночь).
- **Конец дня (End of Day)** – дата-время начала следующего дня (полночь).
- **Название дня (Name of Day)** – день недели словом.

	Дата продажи	День	День недели	День года	Начало дня	Конец дня	Название дня
1	25.06.2012 6:52:18	25	0	177	25.06.2012 0:00:00	26.06.2012 0:00:00	понедельник
2	11.07.2017 6:04:33	11	1	192	11.07.2017 0:00:00	12.07.2017 0:00:00	вторник
3	29.05.2016 3:13:47	29	6	150	29.05.2016 0:00:00	30.05.2016 0:00:00	воскресенье
4	27.04.2018 4:28:52	27	4	117	27.04.2018 0:00:00	28.04.2018 0:00:00	пятница
5	14.10.2013 5:31:54	14	0	287	14.10.2013 0:00:00	15.10.2013 0:00:00	понедельник
6	24.08.2018 7:04:15	24	4	236	24.08.2018 0:00:00	25.08.2018 0:00:00	пятница

Номер недели по ISO

Power Query использует нумерацию недель по американскому стандарту, когда первой неделей года считается та, куда попадает 1 января. В большинстве же стран Европы и Азии (и в России в том числе) принят международный стандарт ISO 8601, в котором первой неделей года считается та, куда попадает первый четверг года (или 4 января). Другими словами, за первую берётся неделя, которая содержит как минимум 4 дня, т. е. больше половины недели из нового года.

В Microsoft Excel за вычисление номеров недель отвечает функция рабочего листа **НОМНЕДЕЛИ.ISO** (**WEEKNUM.ISO**)¹, но в Power Query встроенной функции для этого пока не существует.

Что же делать, если вам нужна нумерация именно по ISO, т. е. по российской системе отсчёта?

Предположим, что у нас есть загруженный в Power Query столбец с датами, для каждой из которых нам нужно определить номер недели по системе ISO:

	Дата
1	31.12.2016
2	01.01.2017
3	25.10.2009
4	20.04.1997
5	06.10.1992
6	05.09.2006
7	27.10.2000
8	17.04.2017
9	27.06.1992
10	21.02.1991
11	20.04.2011
12	11.01.2009

Для лучшего понимания процесса разделим его на три этапа и выполним следующие действия.

Сначала добавим пользовательский столбец **ISO Year**, чтобы понять, к какому году по ISO относится каждая дата, через **Добавление столбца → Пользовательский столбец** (Add Column → Custom Column):

```
=Date.Year(Date.AddDays([Дата], 3 - Date.DayOfWeek([Дата], 1)))
```

Аналогичным образом добавим еще один столбец (назовем его **Start Date**) с датой 3 января каждого ISO-года для каждой даты:

```
=#date([ISO Year], 1, 3)
```

Наконец, добавим вычисляемый столбец **ISO Week**, где вычислим номер недели по ISO формулой:

```
=Number.IntegerDivide(Duration.Days([Дата] - [Start Date]) + Date.DayOfWeek([Start Date], 0) + 6, 7)
```

	Дата	ISO Year	Start Date	ISO Week
1	31.12.2016	2016	03.01.2016	52
2	01.01.2017	2016	03.01.2016	52
3	25.10.2009	2009	03.01.2009	43
4	20.04.1997	1997	03.01.1997	16
5	06.10.1992	1992	03.01.1992	41
6	05.09.2006	2006	03.01.2006	36
7	27.10.2000	2000	03.01.2000	43
8	17.04.2017	2017	03.01.2017	16
9	27.06.1992	1992	03.01.1992	26

¹ Подробнее о способах вычисления в Microsoft Excel номеров недели можно почитать тут

<https://www.planetaexcel.ru/techniques/6/86/> или в моей книге «Microsoft Excel: Готовые решения – бери и пользуйся!» в главе «Номер недели по дате».

Само собой, всё вышеизложенное можно уложить в один шаг и одну формулу (если она вас не напугает):

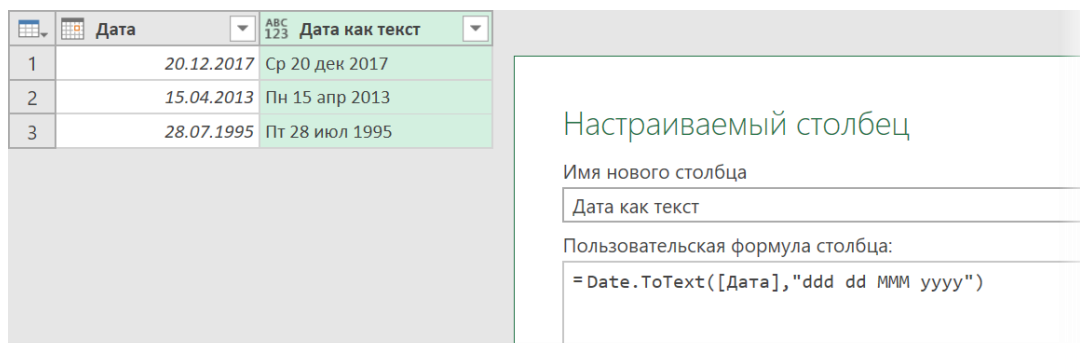
```
=Number.IntegerDivide(Duration.Days([Дата]-#date(Date.Year(Date.AddDays([Дата],3-Date.DayOfWeek([Дата],1))),1,3))+Date.DayOfWeek(#date(Date.Year(Date.AddDays([Дата],3-Date.DayOfWeek([Дата],1))),1,3),0))+6,7)
```

Если подобное придётся использовать часто, то удобнее будет один раз сделать для этого пользовательскую функцию, чтобы потом просто и удобно применять её в каждом таком случае (см. главу [Пользовательские функции](#) в конце книги).

Конвертирование даты в текст

Если нужно представить имеющуюся у нас дату в каком-либо нестандартном виде, то в Microsoft Excel обычно используют функцию **ТЕКСТ (TEXT)**, которая умеет трансформировать входную дату по заданному шаблону. Во встроенном в Power Query языке M существует аналогичная и тоже очень полезная функция **Date.ToText()**.

Выберите на вкладке **Добавление столбца (Add Column)** команду **Настраиваемый столбец (Custom Column)** и введите туда для примера следующую конструкцию:



У этой функции два аргумента: столбец с входными датами и формат (шаблон) представления даты, которую мы хотим получить на выходе. Шаблон представляет собой текстовую строку, которая может содержать английские буквы d, M, y и т. д. (с соблюдением регистра!) в любых сочетаниях.

Рассмотрим несколько примеров для ясности:

Шаблон	Описание	Результат для даты 8 марта 2018 г.
"d"	День месяца (одно- или двузначный)	8
"dd"	День месяца двузначный	08
"ddd"	День недели сокращенно	Чт
"dddd"	День недели полностью	Четверг
"M"	Месяц числом (одно- или двузначным)	3
"MM"	Месяц двузначным числом	03
"MMM"	Название месяца сокращенно	Мар
"MMMM"	Название месяца полностью	Март
"yyyy"	Год полностью (четырёхзначный)	2018
"yy"	Год сокращенно (двузначный)	18
"m"	День и месяц текстом	8 Марта
"yyuuMMdd"	Дата в Unix-формате	20180308
"MM/dd/yyyy"	Дата в «американском» формате	03/08/2018

Вычисление длительностей

*Что ж ты врал-то, что тебе 7 лет?
Весишь-то ты на все 8!
(Карлсон Малышу)*

Типовая задача вычисления разницы между двумя датами (возраст, стаж, срок хранения товара, длительность доставки и т. д.) в обычном Microsoft Excel решается разными способами – от простого арифметического вычитания дат до использования недокументированных функций типа **РАЗНДАТ**¹ (DATEDIF) и макросов.

Давайте посмотрим, как это реализовано в Power Query.

Разница в полных днях

Предположим, что у нас есть даты завоза заказов на склад и последующей их оттуда отправки. Нам нужно выяснить, сколько полных дней каждый заказ пробыл на складе.

Выделим, удерживая клавишу **Ctrl**, сначала столбец с конечными, а потом с начальными датами и выберем на вкладке **Добавление столбца** (Add Column) команду **Дата → Вычесть дни** (Date → Subtract days):

	1 ² 3 Заказ	Дата поставки	Дата отгрузки	1 ² 3 Вычитание
1	1	25.09.2018 8:00:00	25.09.2018 13:38:24	0
2	2	27.06.2018 13:00:00	28.06.2018 10:21:36	0
3	3	19.05.2018 16:21:05	20.05.2018 17:00:00	1
4	4	05.09.2018 6:58:54	15.09.2018 6:44:30	9
5	5	18.08.2018 1:02:10	15.08.2018 13:02:10	-2

Обратите внимание, что эта функция выдаёт на выходе обычное целое число – **количество полных дней** между двумя исходными датами (а не округляет разницу в датах до целых суток). Также она корректно работает и в том случае, если первая выделенная дата раньше последующей: результат будет отрицательным и может в зависимости от ситуации показывать просрочку или отставание от графика.

Продолжительность как тип данных

В отличие от Excel в Power Query есть особый тип данных **Продолжительность** (Duration) для хранения именно длительности или продолжительности какого-либо процесса. Когда мы арифметически вычитаем из одной даты другую, то получаем столбец, по сути, как раз такого типа:

	1 ² 3 Заказ	Дата поставки	Дата отгрузки	ABC 123 Длительность
1	1	25.09.2018 8:00:00	25.09.2018 13:38:24	0.05:38:24
2	2	27.06.2018 13:00:00	28.06.2018 10:21:36	0.21:21:36
3	3	19.05.2018 16:21:05	20.05.2018 17:00:00	1.00:38:54.7700000
4	4	05.09.2018 6:58:54	15.09.2018 6:44:30	9.23:45:35.7670000
5	5	18.08.2018 1:02:10	15.08.2018 13:02:10	-2.12:00:00.3080000

Настраиваемый столбец

Имя нового столбца

Длительность

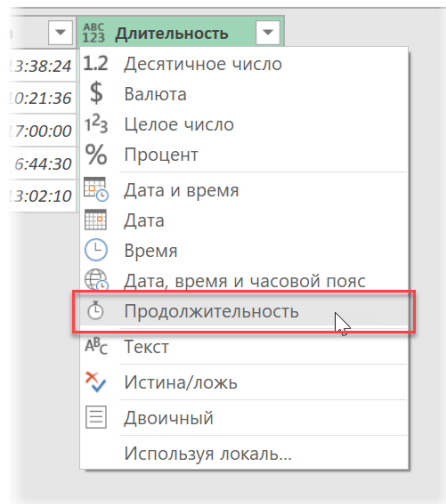
Пользовательская формула столбца:

= [Дата отгрузки]-[Дата поставки]

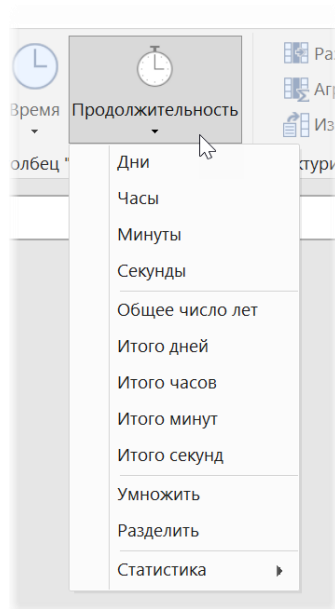
Для 4-го заказа полученный результат интерпретируется как «9 дней, 23 часа, 45 минут и 35 с долями секунд».

¹ Подробнее про эту тайную функцию можно почитать тут <https://www.planetaexcel.ru/techniques/6/105/>.

Для более удобного управления и форматирования лучше явно задать для такого столбца тип данных **Продолжительность (Duration)**, выбрав его из выпадающего списка в шапке столбца:



Это позволит в дальнейшем легко и быстро преобразовывать полученные значения, используя выпадающий список **Продолжительность (Duration)** на вкладке **Преобразование (Transform)**:



Полученные дробные значения можно легко округлить там же, на вкладке **Преобразование (Transform)** с помощью кнопки **Округление (Round)**.

Вычисление возраста

Частным случаем подсчёта длительности является вычисление возраста, т. е. разницы между начальной датой (датой рождения) и сегодняшним моментом. В Power Query это делается совсем легко. Просто выделите столбец с датами и выберите на вкладке **Добавление столбца → Дата → Возраст (Add Column → Date → Age)**.

Полученный столбец с типом данных **Продолжительность** можно затем легко преобразовать в любой нужный вид, используя команду **Преобразование → Продолжительность (Transform → Duration)**.

	Имя	Дата рождения	1.2 Возраст в годах
1	Иван	30.09.1983	35,35890411
2	Александр	16.10.1995	23,30684932
3	Олеся	31.07.2005	13,50958904
4	Надежда	21.10.1980	38,30136986
5	Владимир	24.03.1988	30,8739726

Сдвиг даты на N периодов

Одной из типовых задач при работе с датами является сдвиг даты на заданное количество периодов (дней, месяцев, лет и т. д.) в будущее или прошлое. Предположим, что у нас имеются даты заключения договоров, каждый из которых имеет определенный срок действия (в месяцах). Задача состоит в том, чтобы вычислить дату окончания каждого договора.

К сожалению, Power Query не позволяет через интерфейс выполнять подобные вычисления, но их легко реализовать формулой на языке M. Для этого добавим к нашей таблице ещё один вычисляемый столбец через **Добавление столбца → Настраиваемый столбец (Add Column → Custom Column)** и воспользуемся функцией **Date.AddMonths**:

	Дата подписания	Срок действия	Дата окончания
1	08.07.2013	12	08.07.2014
2	09.04.2013	6	09.10.2013
3	01.10.2015	36	01.10.2018
4	08.08.2014	120	08.08.2024

Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

```
= Date.AddMonths([Дата подписания],[Срок действия])
```

У неё два аргумента: начальная дата и количество месяцев, на которое нужно её сдвинуть вперед или назад.

И совершенно аналогичным образом работают функции сдвига на другие временные интервалы:

- **Date.AddDays**
- **Date.AddQuarters**
- **Date.AddWeeks**
- **Date.AddYears**

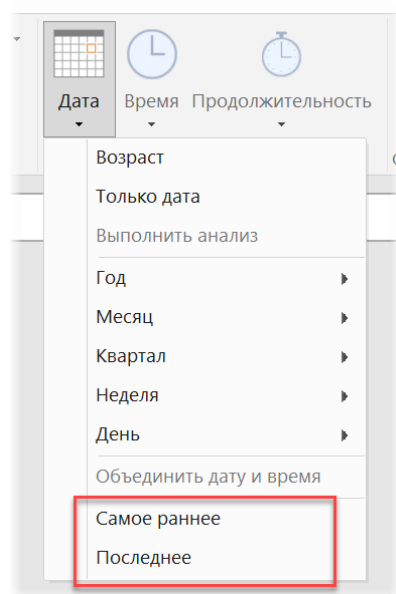
Поиск самой ранней и самой поздней даты

Также одна из весьма распространенных в повседневной практике задач – поиск самой ранней или поздней даты из всех, что у нас есть. Предположим, что в качестве входных данных мы имеем загруженную в Power Query таблицу с продажами по товарам:

	АВС Товар	Дата продажи
1	Джемпер	06.09.2016
2	Комбинезон	14.11.2015
3	Комбинезон	25.11.2015
4	Комбинезон	22.06.2013
5	Бриджи	15.06.2011
6	Рубашка	12.05.2013
7	Бриджи	10.04.2017
8	Джемпер	19.01.2012
9	Джемпер	15.11.2018
10	Бриджи	27.11.2010
11	Блуза	27.06.2012
12	Блуза	26.02.2011
13	Платье	11.08.2015

Во всем столбце

Если нам нужно найти дату самой первой или самой последней сделки по всем данным (т. е. не по каждому товару, а во всём столбце), то можно просто выделить столбец с **Дата продажи** и воспользоваться готовым инструментом – кнопкой **Дата** → **Самое раннее / Последнее** (Date → Earliest / Latest) на вкладке **Преобразование** (Transform):



В этом случае на выходе мы получим не таблицу, а одно-единственное значение – самой ранней или самой поздней даты соответственно.

По каждой группе значений

Это было просто. Теперь представим, что перед нами стоит более сложная задача: нужно найти дату самой ранней и самой последней сделки по каждому товару. В этом случае нам поможет уже знакомый инструмент группировки, который отлично умеет работать с датами. Нажмем на вкладке **Преобразование** (Transform) кнопку **Группировать по** (Group by) и введём в открывшееся окно следующие параметры:

1. Переключитесь в расширенный режим выбором опции **Подробнее** (Advanced).
2. Выберите группировку по товарам.

3. Добавьте ещё одну строку для агрегирования кнопкой **Добавление агрегирования** (Add aggregation).
4. Введите имена столбцов и выберите функции **Мин.** и **Макс.** для поля **Дата продажи**.

Группировать по

Базовый Подробнее

Укажите столбцы для группировки и желаемые выходные данные.

Группировка

Товар

Добавление группирования

Имя нового столбца	Операция	Столбец
Первая сделка	Мин.	Дата продажи
Последняя сделка	Макс.	Дата продажи

Добавление агрегирования

OK Отмена

После нажатия на **OK** мы увидим желаемое:

	Товар	Первая сделка	Последняя сделка
1	Джемпер	19.01.2012	15.11.2018
2	Комбинезон	13.03.2013	09.07.2016
3	Бриджи	27.11.2010	10.04.2017
4	Рубашка	12.05.2013	12.05.2013
5	Блуза	26.02.2011	31.01.2015
6	Джинсы	02.03.2011	17.05.2018
7	Брюки	03.12.2011	16.11.2016
8	Шарфы	06.11.2015	11.05.2018

При необходимости можно выделить полученные столбцы (в обратном порядке) и добавить столбец с длительностью продаж по каждому товару через **Добавление столбца** → **Дата** → **Вычесть дни** (Add Column → Date → Subtract days), как мы это делали в предыдущих главах.

Заполнение пробелов в датах

Рассмотрим следующую задачу: предположим, что вы ведете несколько проектов с разным бюджетом и хотите наглядно представить свои расходы на каждый из них. То есть из вот такой исходной таблицы:

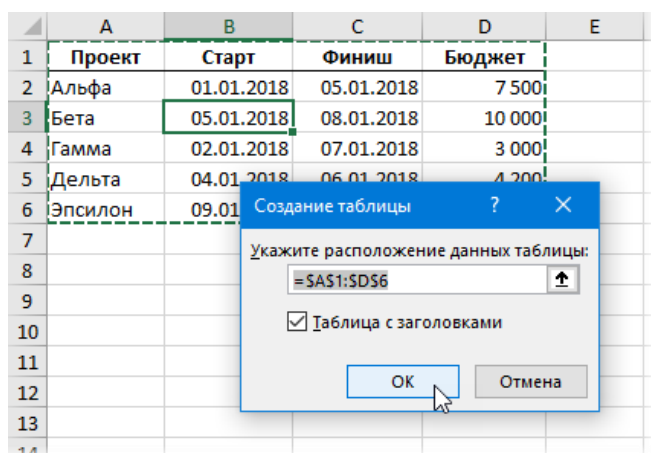
	A	B	C	D
1	Проект	Старт	Финиш	Бюджет
2	Альфа	01.01.2018	05.01.2018	7 500
3	Бета	05.01.2018	08.01.2018	10 000
4	Гамма	02.01.2018	07.01.2018	3 000
5	Дельта	04.01.2018	06.01.2018	4 200
6	Эпсилон	09.01.2018	13.01.2018	10 000
7				

... хочется получить что-то похожее на такую:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	
1	Проект	Старт	Финиш	Бюджет	1 янв	2 янв	3 янв	4 янв	5 янв	6 янв	7 янв	8 янв	9 янв	10 янв	11 янв	12 янв	13 янв	
2	Альфа	01.01.2018	05.01.2018	7 500	1 500	1 500	1 500	1 500	1 500									
3	Бета	05.01.2018	08.01.2018	10 000					2 500	2 500	2 500	2 500						
4	Гамма	02.01.2018	07.01.2018	3 000		500	500	500	500	500	500							
5	Дельта	04.01.2018	06.01.2018	4 200							1 400	1 400	1 400					
6	Эпсилон	09.01.2018	13.01.2018	10 000									2 000	2 000	2 000	2 000	2 000	
7																		
8																		

Другими словами, необходимо размазать бюджет по дням каждого проекта и получить упрощенный вариант проектной диаграммы Ганта. Руками такое делать долго и скучно, макросами сложно, а вот Power Query в такой ситуации проявляет свою мощь во всей красе.

Сначала превратим нашу исходную таблицу в «умную», выбрав команду **Форматировать как таблицу** на вкладке **Главная** (Home → Format as Table) или нажав сочетание клавиш **Ctrl+T**:



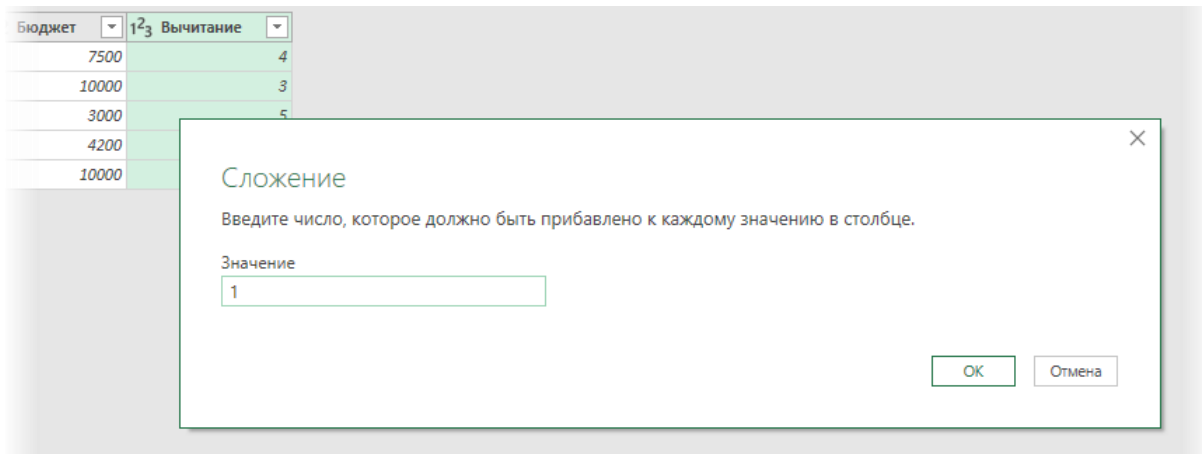
Затем переходим на вкладку **Данные** (Data) и жмем кнопку **Из таблицы / диапазона** (From Table/Range), чтобы загрузить эту таблицу в Power Query.

Чтобы посчитать бюджет в день, нужно вычислить длительность каждого проекта. Для этого выделим (удерживая клавишу **Ctrl**) сначала столбец **Финиш**, а потом **Старт** и выберем команду **Добавление столбца → Дата → Вычесть дни** (Add Column → Date → Subtract days):

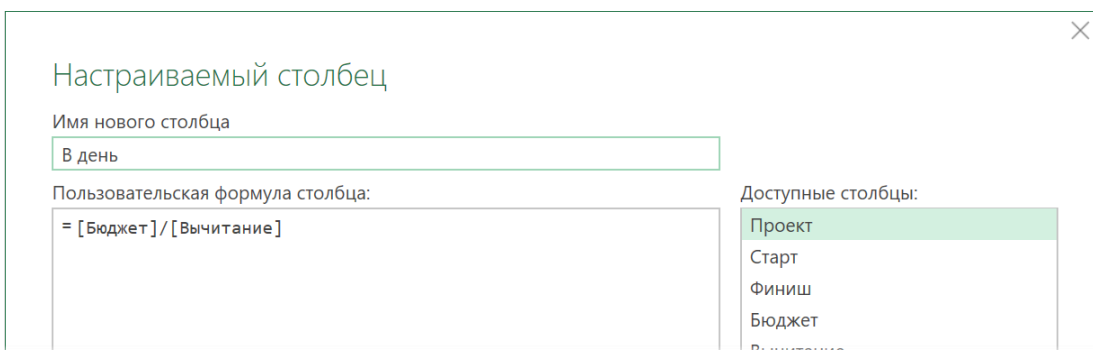
	A	B	C	D	E	F
1	Проект	Старт	Финиш	Бюджет	Вычитание	
1	Альфа	01.01.2018	05.01.2018	7500	4	
2	Бета	05.01.2018	08.01.2018	10000	3	
3	Гамма	02.01.2018	07.01.2018	3000	5	
4	Дельта	04.01.2018	06.01.2018	4200	2	
5	Эпсилон	09.01.2018	13.01.2018	10000	4	

Обратите внимание, что полученные числа на 1 меньше, чем нужно, т. к. предполагается, что начинаем каждый проект мы в первый день утром, а заканчиваем в последний день вечером. Поэтому выделим полученный

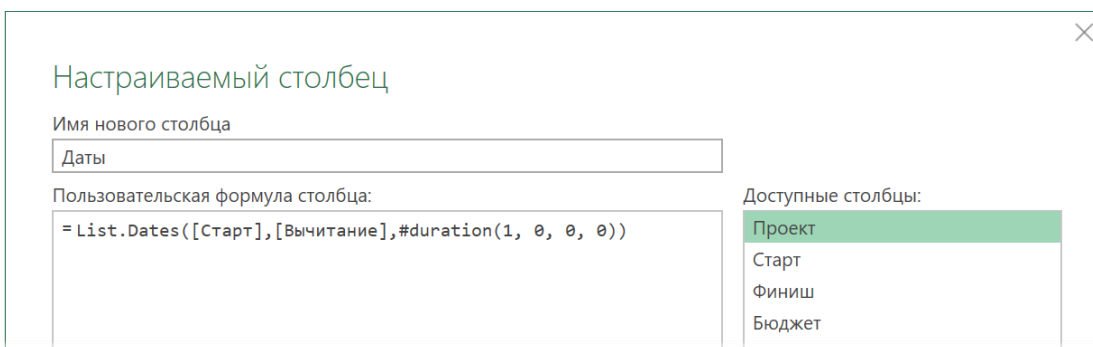
столбец и добавим к нему единицу с помощью команды **Преобразование** → **Стандартный** → **Добавить** (Transform → Standard → Add):



Теперь добавим столбец, где вычислим бюджет в день. Для этого на вкладке **Добавление столбца** (Add Column) нажмем кнопку **Настраиваемый столбец** (Custom Column) и в появившееся окно введём имя нового поля и формулу расчета, используя имена столбцов из списка справа:



Теперь самый тонкий момент: создадим еще один вычисляемый столбец со списком дат от старта до финиша с шагом 1 день. Для этого опять жмем кнопку **Настраиваемый столбец** (Custom Column) и используем функцию встроенного языка M, которая называется **List.Dates()**:



У этой функции три аргумента:

- **Начальная дата** – в нашем случае берется из столбца **[Старт]**.
- **Количество дат**, которые надо сгенерировать, – в нашем случае это число дней по каждому проекту, которое мы посчитали ранее в столбце **[Вычитание]**.
- **Временной шаг** – задается конструкцией **#duration(1, 0, 0, 0)**, означающей на языке M «один день, ноль часов, ноль минут, ноль секунд».

После нажатия на **ОК** получим столбец **Даты**, содержащий в каждой ячейки список (List) дат в заданном интервале с шагом один день:

ABC	Проект	Старт	Финиш	1.2 Бюджет	1.2 Вычитание	ABC 123 В день	ABC 123 Даты
1	Альфа	01.01.2018	05.01.2018	7500		5	1500 List
2	Бета	05.01.2018	08.01.2018	10000		4	2500 List
3	Гамма	02.01.2018	07.01.2018	3000		6	500 List
4	Дельта	04.01.2018	06.01.2018	4200		3	1400 List
5	Эпсилон	09.01.2018	13.01.2018	10000		5	2000 List

List
01.01.2018
02.01.2018
03.01.2018
04.01.2018
05.01.2018

Останется развернуть содержимое этих вложенных списков, воспользовавшись кнопкой с двойными стрелками в шапке таблицы и выбрав затем команду **Развернуть в новые строки (Extract to new rows)**:

ABC	Проект	Старт	Финиш	1.2 Бюджет	1.2 Вычитание	ABC 123 В день	ABC 123 Даты
1	Альфа	01.01.2018	05.01.2018	7500		5	1500 01.01.2018
2	Альфа	01.01.2018	05.01.2018	7500		5	1500 02.01.2018
3	Альфа	01.01.2018	05.01.2018	7500		5	1500 03.01.2018
4	Альфа	01.01.2018	05.01.2018	7500		5	1500 04.01.2018
5	Альфа	01.01.2018	05.01.2018	7500		5	1500 05.01.2018
6	Бета	05.01.2018	08.01.2018	10000		4	2500 05.01.2018
7	Бета	05.01.2018	08.01.2018	10000		4	2500 06.01.2018
8	Бета	05.01.2018	08.01.2018	10000		4	2500 07.01.2018
9	Бета	05.01.2018	08.01.2018	10000		4	2500 08.01.2018
10	Гамма	02.01.2018	07.01.2018	3000		6	500 02.01.2018
11	Гамма	02.01.2018	07.01.2018	3000		6	500 03.01.2018
12	Гамма	02.01.2018	07.01.2018	3000		6	500 04.01.2018
13	Гамма	02.01.2018	07.01.2018	3000		6	500 05.01.2018
14	Гамма	02.01.2018	07.01.2018	3000		6	500 06.01.2018
15	Гамма	02.01.2018	07.01.2018	3000		6	500 07.01.2018
16	Дельта	04.01.2018	06.01.2018	4200		3	1400 04.01.2018
17	Дельта	04.01.2018	06.01.2018	4200		3	1400 05.01.2018

Дальше можно смело строить по этим данным сводную таблицу, поместив поле **Проект** в строки, поле **Даты** – в столбцы, а поле **В день** в область данных:

	A	B	C	D	E	F	G	H	I	J	K
1											
2											
3	Сумма по полю В день	Названия столбцов									
4	Названия строк	01.01.2018	02.01.2018	03.01.2018	04.01.2018	05.01.2018	06.01.2018	07.01.2018	08.01.2018	09.01.2018	10.01.2018
5	Альфа	1500	1500	1500	1500	1500					
6	Бета					2500	2500	2500	2500		
7	Гамма		500	500	500	500	500	500			
8	Дельта				1400	1400	1400				
9	Эпсилон									2000	2000
10	Общий итог	1500	2000	2000	3400	5900	4400	3000	2500	2000	2000
11											
12											

Поскольку это сводная таблица, то даты в шапке можно «нарезать» не только по дням, а с любым нужным вам шагом (по месяцам, кварталам, годам и т. д.), если щелкнуть по любой дате правой кнопкой мыши и выбрать команду **Группировать (Group)**.

Если сводная таблица вам по каким-то причинам не подходит, можно использовать её аналог в Power Query, который мы уже разбирали ранее, – инструмент **Столбец сведения (Pivot Column)** на вкладке **Преобразование (Transform)**, выделив предварительно столбец с датами:

Столбец сведения

Использовать имена в столбце "Даты", чтобы создать новые столбцы.

Столбец значений (i)

▼ В день

▷ Расширенные параметры

[Дополнительные сведения о столбце сведения](#)

OK
Отмена

После нажатия на **OK** получаем результат, очень близкий к желаемому:

	AB C Проект	Старт	Финиш	1.2 Бюджет	1.2 Вычитание	ABC 123 01.01.2018	ABC 123 02.01.2018	ABC 123 03.01.2018
1	Альфа	01.01.2018	05.01.2018	7500	5	1500	1500	1500
2	Бета	05.01.2018	08.01.2018	10000	4	null	null	null
3	Гамма	02.01.2018	07.01.2018	3000	6	null	500	500
4	Дельта	04.01.2018	06.01.2018	4200	3	null	null	null
5	Эпсилон	09.01.2018	13.01.2018	10000	5	null	null	null

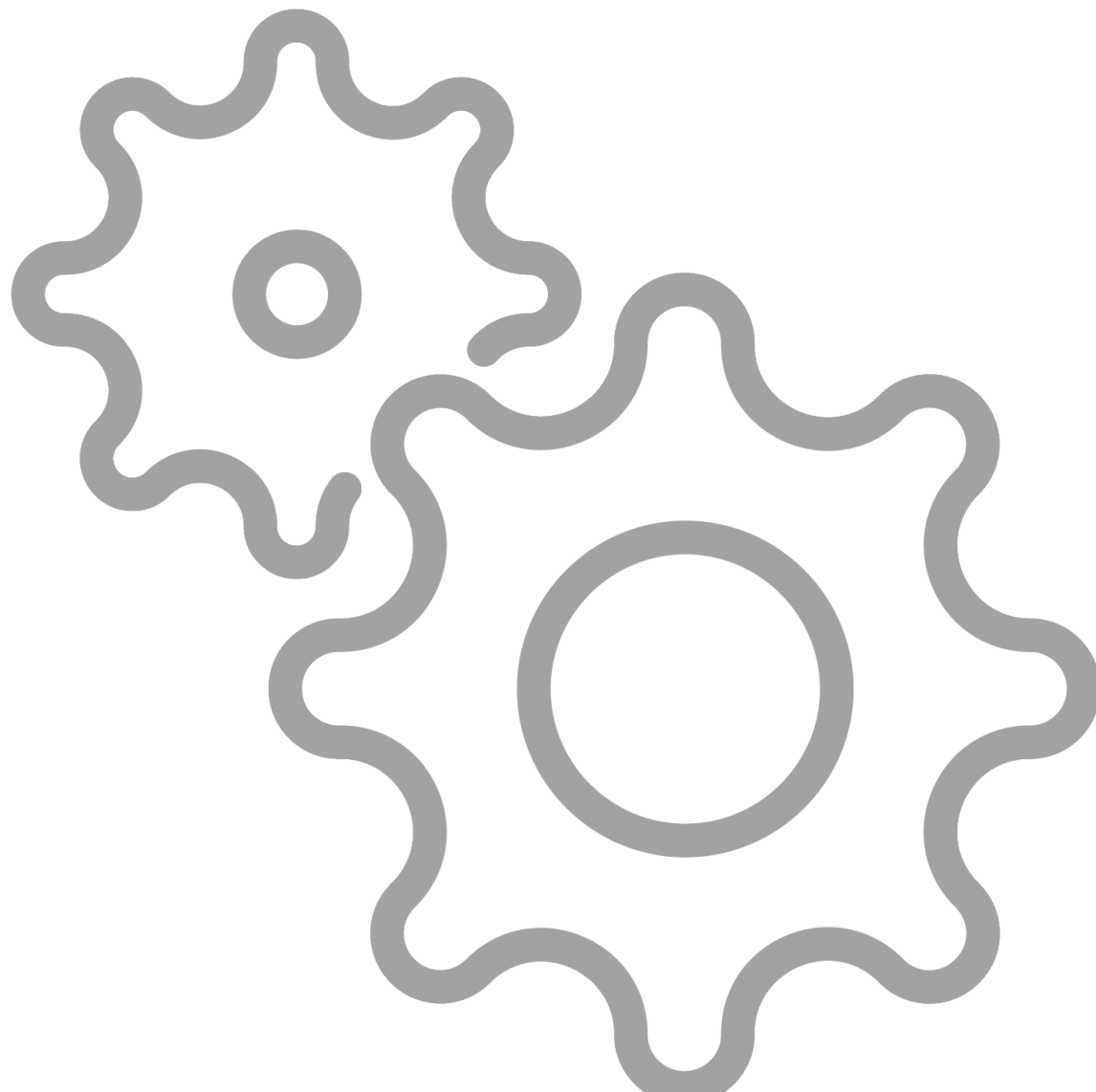
Останется только удалить лишние столбцы и выгрузить таблицу обратно на лист.

И самое приятное, как всегда, в том, что в будущем можно смело редактировать старые или дописывать к исходной таблице новые проекты, а правую таблицу с датами затем обновлять правой кнопкой мыши – и все проделанные нами действия Power Query повторит автоматически.

Работа с запросами

Запросы – наш основной инструмент в Power Query, и неплохо бы изучить их поближе, верно? Кроме создания и обновления в этой главе мы научимся:

- **группировать** запросы в папки для удобства;
- **защищать** запросы от «шаловливых ручек» других пользователей (чтобы могли обновлять, но не могли испортить);
- **просматривать зависимости** между запросами, если они связаны друг с другом;
- обновлять запросы по **заданному расписанию** (вы спите, а тяжёлый запрос сам обновляется – ну не прелесть?);
- **копировать, дублировать и делать ссылку** на запросы и понимать, чем отличаются эти три действия;
- создавать и удалять запросы **макросами на VBA**.



Группировка запросов

Если вы активно используете Power Query и создали в книге уже больше десятка запросов для импорта и обработки данных, то имеет смысл заняться наведением порядка в правой панели **Запросы и подключения** (Queries & Connections), где они все отображаются.

Во-первых, порядок отображения запросов в этой панели не всегда бывает удобен, т. к. запросы там отображаются не по алфавиту или как-то ещё, а просто в той последовательности, в которой вы их создали. Перетаскивать их мышью нельзя, но можно щёлкнуть по любому запросу правой кнопкой мыши и выбрать в контекстном меню команды **Вверх** (Move Up) или **Вниз** (Move down).

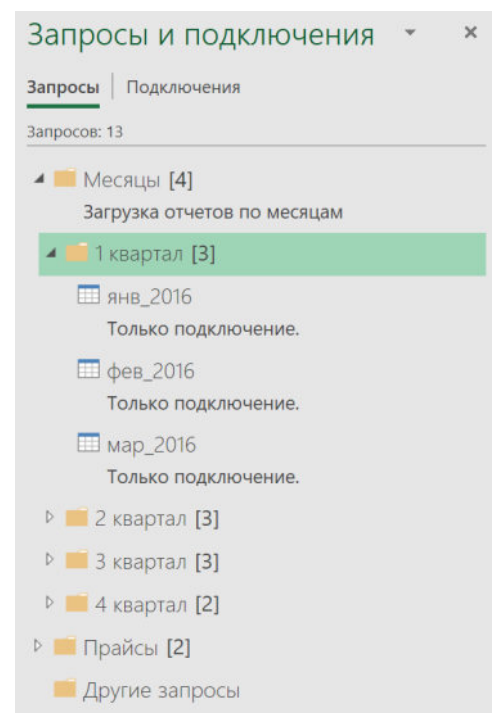
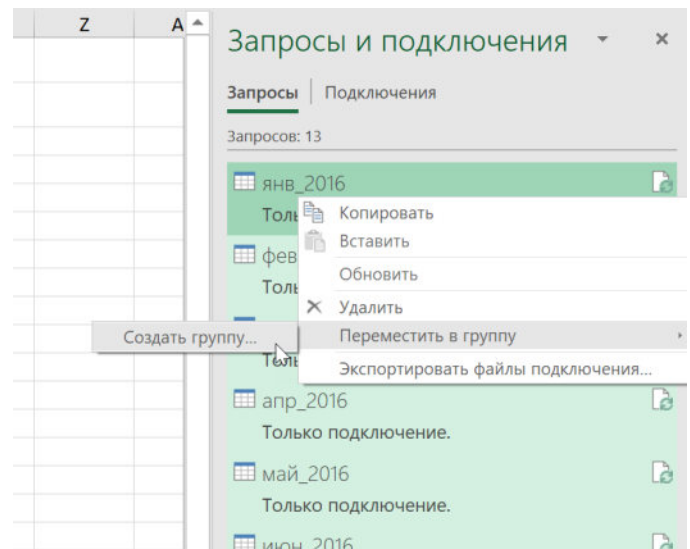
Ещё более удобным инструментом является возможность группировать запросы в папки. Для этого нужно выделить несколько запросов, удерживая клавишу **Ctrl**, а потом щёлкнуть по ним правой кнопкой мыши и выбрать команду **Переместить в группу → Создать группу** (Move to group → New group) или уже имеющуюся группу, куда хотим добавить эти запросы:

Получится аккуратная и компактная папка, содержимое которой можно сворачивать и разворачивать при необходимости.

В особо тяжёлых случаях можно даже делать подпапки, помещая их в другие папки (группы), используя всё ту же команду **Переместить в группу** (Move to group).

Будьте осторожны при удалении папок, т. к. при этом будут безвозвратно удалены все запросы, которые в них хранятся. Если же вам нужно удалить только группировку, а сами запросы оставить в исходном виде, то лучше использовать в контекстном меню команду **Разгруппировать** (Ungroup), а не **Удалить** (Delete).

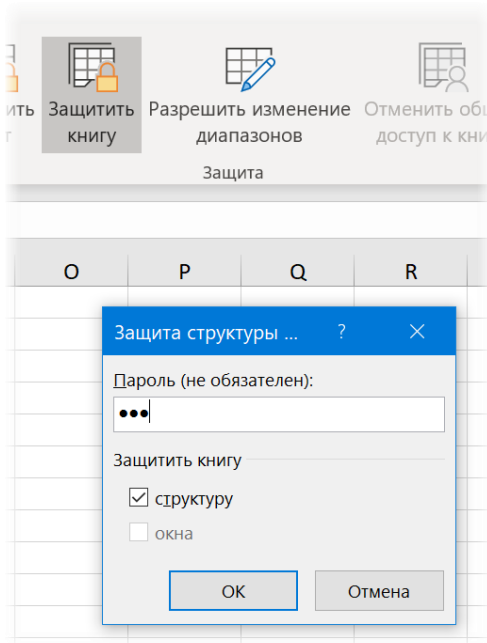
Все вышеперечисленные действия никак не влияют на работу или быстродействие выполнения ваших запросов, а только на удобство и наглядность при работе с ними, но в больших проектах с десятками запросов это весьма важный нюанс.



Защита запросов

Если вы планируете в дальнейшем отдать вашу книгу с запросом Power Query другим пользователям или выложить ее в общий доступ, то, возможно, у вас возникнет желание запретить чужим людям «лазить под капот» и вмешиваться в настроенный запрос.

Это легко сделать, если на вкладке **Рецензирование** нажать кнопку **Защитить книгу** (Review → Protect Workbook):



Придумайте и введите пароль, включите флажок **Структуру** (Structure) и нажмите **ОК** – и редактирование запросов станет недоступным, но возможность их обновлять останется, т. е. рабочий функционал сохранится.

Просмотр зависимостей между запросами

Реализуя более-менее сложный проект с использованием Power Query, вы очень быстро можете оказаться в ситуации, когда у вас в правой панели три десятка запросов, и вдобавок между ними есть ещё связи и зависимости. Результаты одного запроса могут использоваться в качестве входных данных для другого, вы можете выполнять объединение запросов (добавление или слияние) и т. д.

Разобраться в этой каше очень помогает наглядная диаграмма, которую можно включить в редакторе запросов кнопкой **Зависимости запроса (Query Dependencies)** на вкладке **Просмотр (View)**:

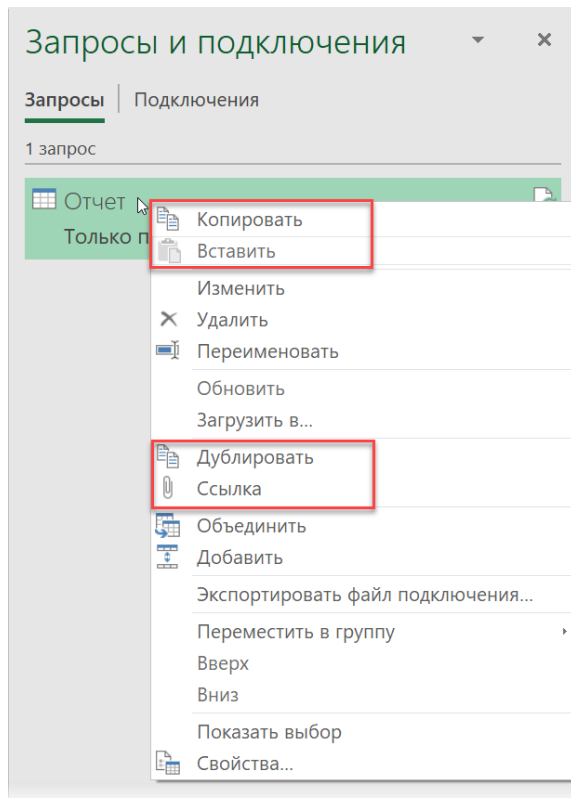
The screenshot shows the Power Query Editor interface. The main window displays a table with columns: Дата, Товар, Продавец, Город, Статус, Регион, and Выручка. The 'View' tab is active, and the 'Query Dependencies' button is highlighted with a red circle '2'. A red circle '1' points to the 'View' tab. A red circle '3' points to the 'Structure' (Структура) dropdown menu at the bottom right of the dependency window, which is open to show layout options like 'Макет с направлением сверху вниз' (Top-down layout), 'Макет с направлением снизу вверх' (Bottom-up layout), 'Макет с направлением слева направо' (Left-to-right layout), and 'Макет с направлением справа налево' (Right-to-left layout).

Как-то редактировать (удалять, добавлять) связи в этой схеме, конечно, нельзя, но она явно добавит наглядности вашему проекту.

Также в нижней части окна есть выпадающий список **Структура (Structure)**, который дополнительно позволяет настроить внешний вид отображаемой схемы.

Копирование, дублирование и ссылка на запрос

Если щёлкнуть правой кнопкой мыши по любому имеющемуся запросу в правой панели **Запросы и подключения** (Queries & Connections), то среди прочих опций и действий с запросами будут две похожие, на первый взгляд, по результату, но совершенно разные по внутренней механике команды – **Дублировать** (Duplicate) и **Ссылка** (Reference):

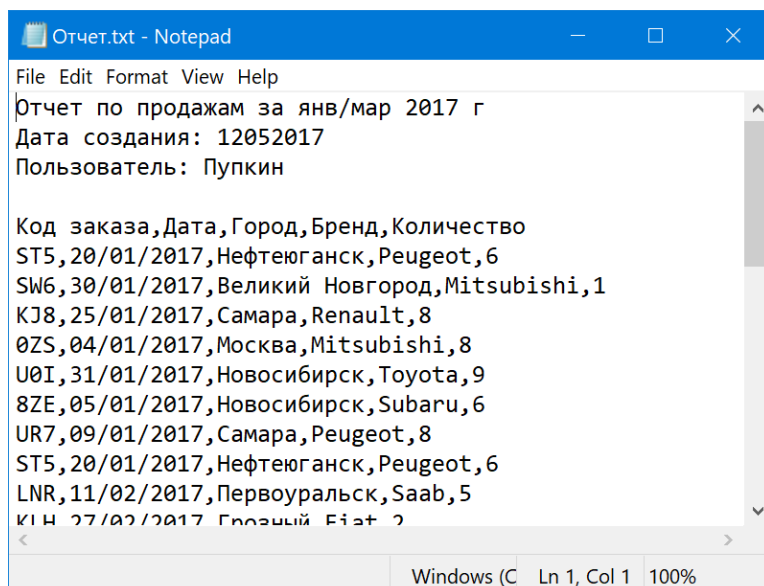


Плюс ко всему там же рядом находятся еще и команды **Копировать** (Copy) и **Вставить** (Paste), которые тоже добавляют тумана, порождая вопросы типа «А чем тогда копирование отличается от дублирования?».

Давайте разберёмся, в чем разница между этими действиями и когда что лучше использовать.

Дублирование

Представьте, что у вас есть запрос, который загружает в Power Query и приводит в порядок содержимое вот такого текстового файла:



Шагов, как вы понимаете, тут будет немного:

1. Грузим данные из файла на шаге **Источник (Source)**.
2. Убираем четыре верхние строки.
3. Поднимаем первую строку данных в шапку.
4. Настраиваем форматы данных для получившихся столбцов.

The screenshot shows the Power Query editor in Excel. The main area displays a table with 11 rows and 6 columns: 'Код заказа', 'Дата', 'Город', 'Бренд', and 'Количество'. The formula bar at the top shows the query name: `Table.TransformColumnTypes("#Повышенные заголовки",{"Data", type date})`. On the right, the 'Parameters of query' pane is open, showing the 'Applied Steps' section with the following steps: 'Source', 'Remove top rows', 'Promote headers', and 'Changed type'. The 'Changed type' step is currently selected.

Теперь представьте, что у вас вдруг появилась необходимость загрузить в Power Query еще один очень похожий файл, но из другой папки.

Можно, конечно, создать новый запрос и проделать все эти шаги заново, но это скучно, непродуктивно, и шагов в запросе может быть много. Гораздо проще будет сделать дубликат запроса со всеми входящими в него шагами, а потом просто подправить путь к исходному файлу на первом шаге. Для этого щёлкнем правой кнопкой мыши по первоначальному запросу и выберем команду **Дублировать (Duplicate)**, а затем нажмём на значок шестерёнки справа от шага **Источник (Source)** и укажем путь к новому файлу:

The screenshot shows the 'Queries and Connections' pane on the right, which lists two queries: 'Отчет' and 'Отчет (2)'. The 'Отчет (2)' query is highlighted in green. On the left, the 'Parameters of query' pane is open, showing the 'Applied Steps' section. The 'Source' step is selected, and a gear icon (settings) is visible next to it, indicating that the user is about to edit the source path.

Обратите внимание, что созданные запросы совершенно независимы и никак не связаны друг с другом, т. е. любые правки, вносимые в один из запросов, никак не влияют на другой.

Ссылка

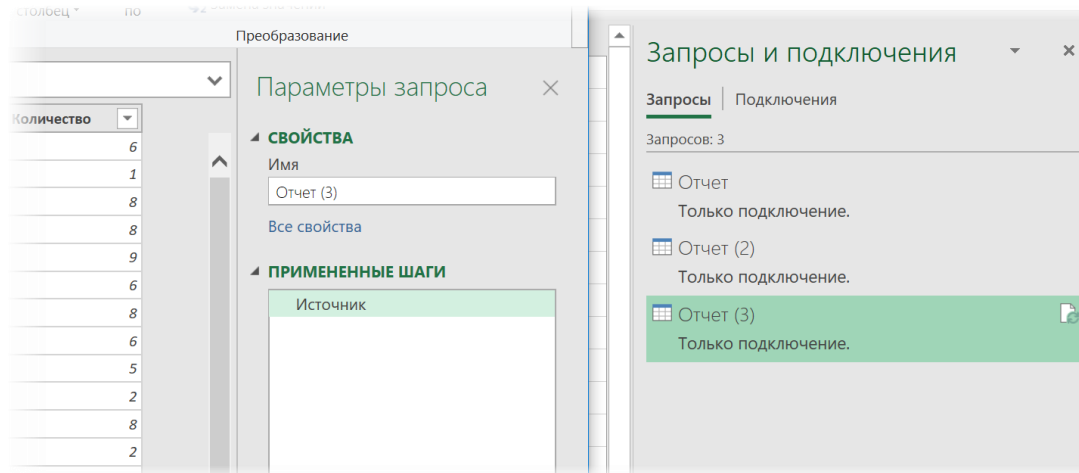
Теперь представим себе другую ситуацию. Предположим, что у вас есть исходный запрос, который загружает всё те же данные из того же текстового файла, что мы разобрали выше. К вам приходят двое ваших коллег и просят дополнительно отфильтровать данные только по их городам: одному нужен отчет по *Самаре*, другому – по *Новосибирску*. Вам же при этом нужен полный набор данных для дальнейшего анализа, и вы у себя в отчете ничего фильтровать не собирались.

Как быть?

Опять же, можно, как мы разбирали в предыдущем пункте, дублировать наш запрос ещё два раза и добавить в каждый дубликат дополнительные шаги фильтрации для нужного города. Но представьте, что завтра поменяется формат выгрузки текстовых данных, и вам нужно будет удалять уже не три верхние строки, а четыре или дополнительно настраивать числовые форматы, убирать лишние столбцы и т. д. Все эти правки кропотливо

придется вносить не только в ваш оригинальный запрос, но и в каждый созданный дубликат – и для *Самары*, и для *Новосибирска* отдельно.

Гораздо правильнее будет в такой ситуации щёлкнуть мышью по исходному запросу и выбрать команду **Ссылка (Reference)**. В результате мы получим новый запрос **Отчет (3)**, где будет только один шаг **Источник (Source)**, который ссылается на результаты последнего шага нашего первоначального запроса **Отчет**:

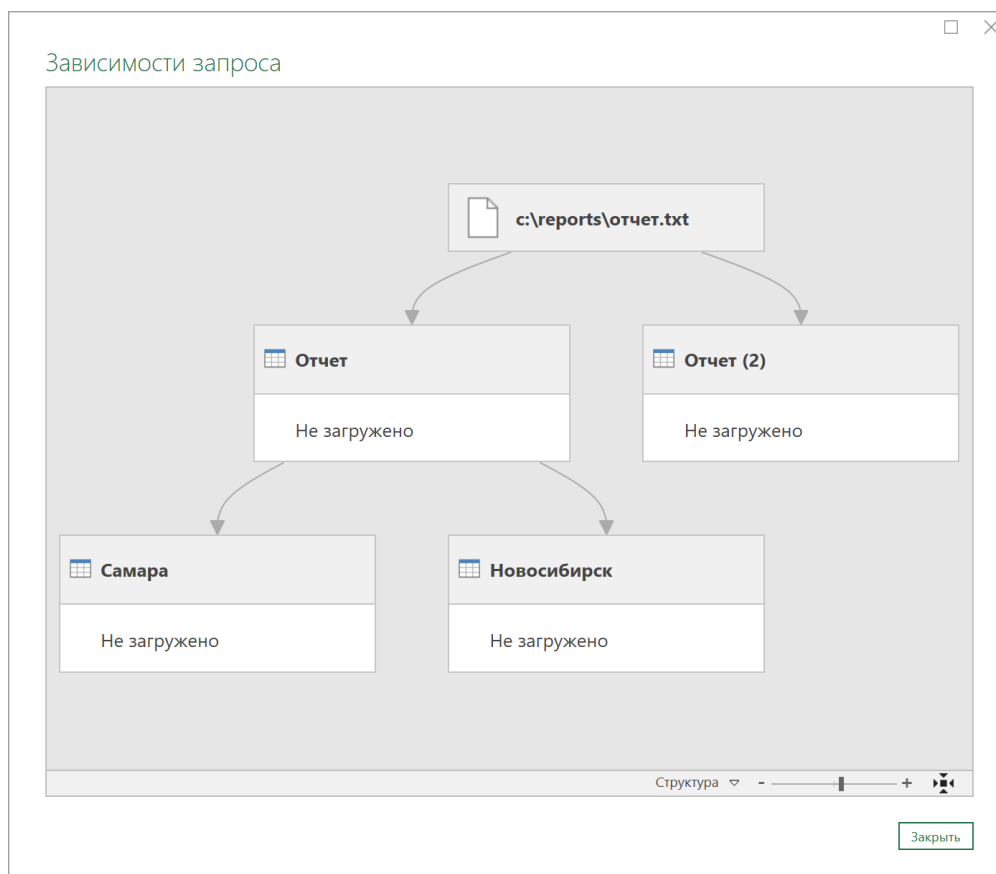


Теперь можно добавить фильтрацию по нужному городу (Самаре), переименовать запрос соответствующим образом и поделиться им с первым коллегой.

Для второго коллеги мы также создаем ссылку на исходный запрос, но фильтруем уже по Новосибирску.

Преимущество такого подхода в том, что при внесении каких-либо изменений в наш первоначальный запрос **Отчет** нам уже не потребуется править запросы **Самара** и **Новосибирск**, потому что они наследуют те данные, что он им выдаёт. Другими словами, ссылки удобно использовать, когда мы хотим сделать разветвление в последовательности шагов, сохраняя все результаты, полученные до этого.

Всё описанное выше отлично иллюстрирует диаграмма зависимостей запросов, которую можно включить одноименной кнопкой на вкладке **Просмотр (View)** в окне редактора Power Query:



Для полноты картины имеет смысл добавить, что если дублирование можно делать всегда, то со ссылками всё не так однозначно. Например, невозможно сделать обратные ссылки (это была бы стрелка вверх на диаграмме зависимостей) или циклические ссылки, когда результаты запроса могут менять источник, откуда сам же запрос берет данные.

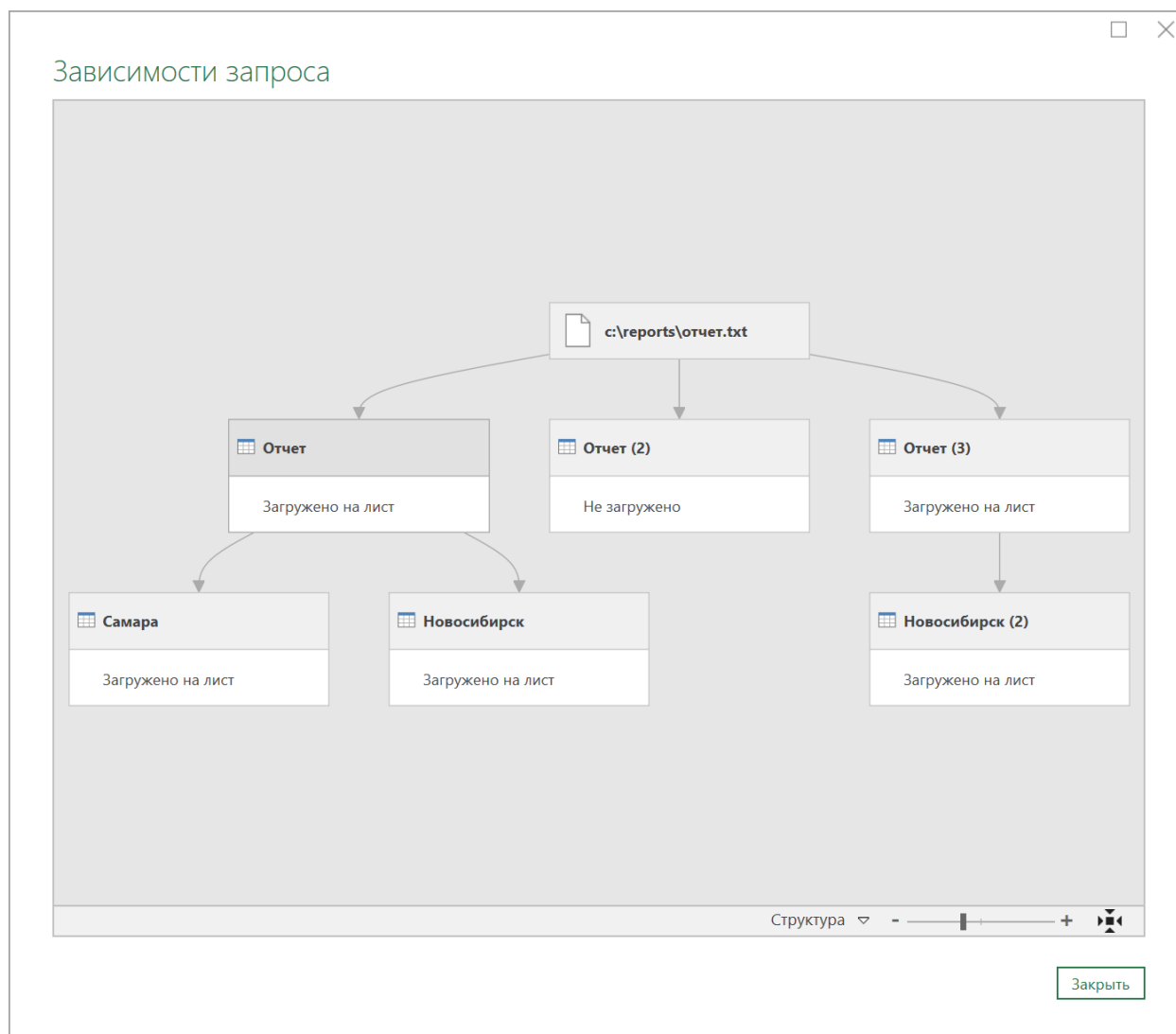
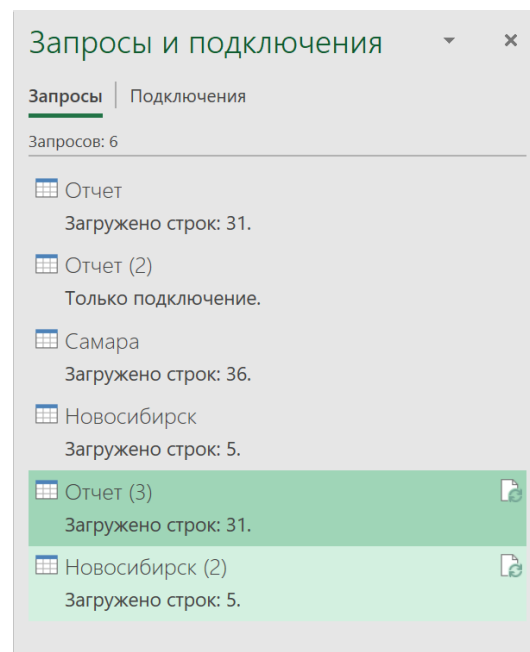
Копирование

Команда **Копировать (Сору)**, находящаяся в том же контекстном меню рядом с **Дублировать** и **Ссылка**, работает немного иначе.

Если применить её к обычному запросу, который никак не связан с другими, то мы получим просто его копию, т.е. **Копировать** сработает как **Дублировать**. Но если запрос имеет связи, то при копировании и последующей вставке мы получим копию не только его, но и всех влияющих на него запросов.

Так, например, если копировать запрос **Новосибирск**, то мы получим не только ожидаемый дубликат **Новосибирск (2)**, но и дубль связанного с ним Отчета – **Отчет (3)**.

Созданные копии опять же абсолютно автономны и никак не связаны в данном случае с оригиналами, но зато связаны друг с другом, что хорошо видно на диаграмме связей:



Делимся запросами с внешним миром

Делитесь своими знаниями: это способ достичь бессмертия.

(Далай Лама XIV)

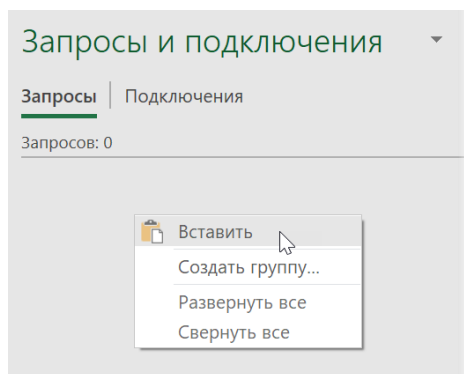
Представьте, что вы сделали запрос, который экономит вам кучу времени и сил, и вы решили поделиться этим запросом с вашими коллегами, чтобы облегчить их жизнь. Или у вас возникла необходимость перенести один из ваших запросов в другую книгу Excel для решения аналогичной задачи. В подобных ситуациях у вас есть несколько вариантов дальнейших действий.

Способ 1. Пересылка файла

Первое, что приходит в голову, – это просто переслать всю книгу Excel с нужным запросом другому пользователю. Правда, если в книге были и другие запросы или данные, которыми вы не хотите делиться, то сначала придется их удалить (сделав предварительно копию файла). При этом всегда есть шанс что-то забыть вычистить, особенно если вы использовали **Модель Данных (Data Model)** и надстройку Power Pivot.

Способ 2. Копирование и вставка запроса в другой файл

Если вам нужно перенести ваш запрос в другой файл, то проще будет скопировать его, щёлкнув по нему правой кнопкой мыши и выбрав команду **Копировать (Copy)**, и вставить его затем в другой (или в новый) файл, также щёлкнув правой кнопкой в правой панели **Запросы и подключения (Queries & Connections)** и выбрав команду **Вставить (Paste)**:



Таким способом можно перенести из одной книги в другую и сразу несколько запросов, выделив их предварительно с клавишей **Ctrl**.

Однако тут кроется и пара отрицательных моментов.

- Если вы откроете новую книгу, то панели с запросами не будет видно, и её надо будет сначала включить кнопкой **Запросы и подключения** на вкладке **Данные (Data → Queries & Connections)**.
- При таком способе нет возможности выбрать, куда и как выводить результаты нового запроса: его поведение также копируется из исходного файла. То есть если в вашей книге запрос выводился как подключение и загружал при этом данные в Модель Данных, то и после вставки в чуждой файл он будет делать то же самое. Это не всегда удобно. Часто приходится потом вручную это корректировать, щёлкая по вставленному запросу правой кнопкой мыши и выбирая команду **Загрузить в... (Load to...)**.

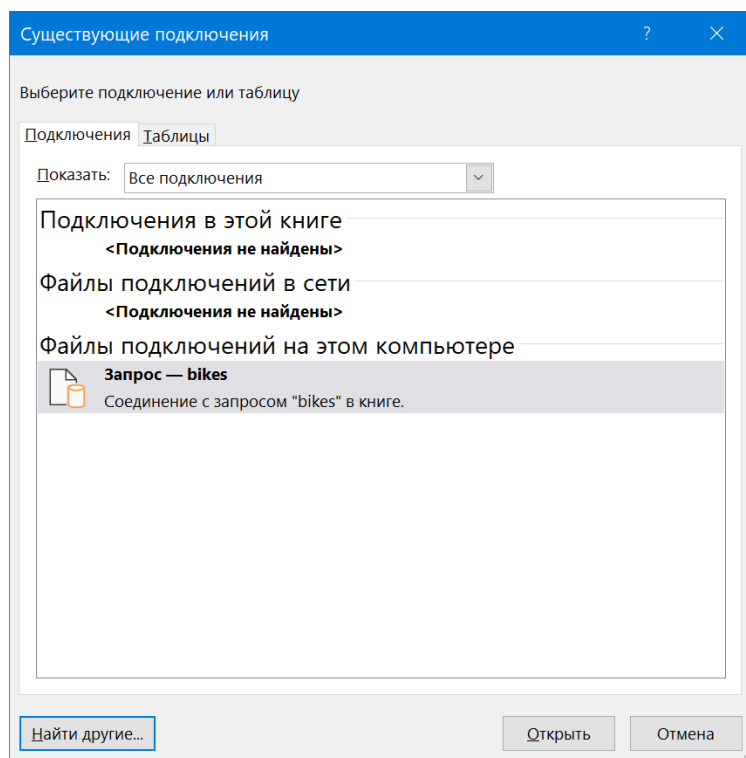
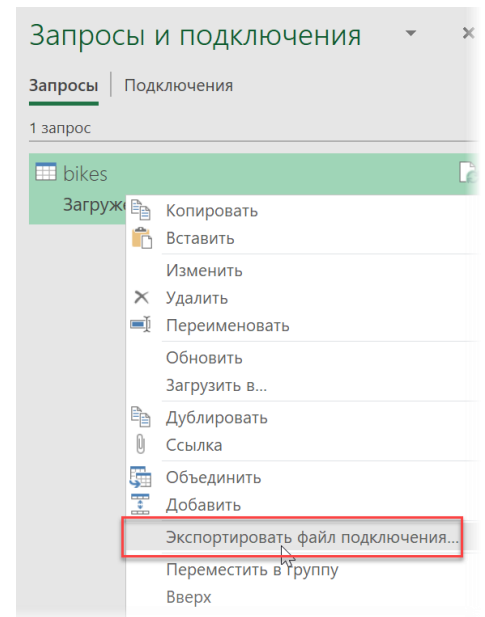
Способ 3. Экспорт файла подключения

Ещё один удобный способ поделиться созданным запросом с внешним миром – это экспортировать все данные запроса (настройки подключения к источнику данных + шаги обработки Power Query) в отдельный файл особого формата – **ODC (Office Database Connection file)**, который потом легко использовать на других компьютерах.

Для этого щёлкните по запросу в правой панели и выберите команду **Экспортировать файл подключения (Export Connection File)**, а затем сохраните получившийся ODC-файл в любую удобную вам папку.

Воспользоваться созданным ODC-файлом очень легко: откройте любой файл, куда вы хотите импортировать свой запрос, и на вкладке **Данные** нажмите кнопку **Существующие подключения (Data → Existing Connections)**.

Если вы сохранили файл подключения в стандартную папку **Мои подключения** (она будет предложена по умолчанию), то ваш запрос будет виден сразу. Если вы сохранили запрос в другом месте, то придется нажать в левом нижнем углу на кнопку **Найти другие (Browse for More...)** и указать файл самостоятельно.



После выбора нужного запроса нажмите кнопку **Открыть (Open)**, и на следующем шаге появится уже знакомое нам окно с выбором места для выгрузки результатов запроса.

Обновление запросов по расписанию

Если ваши запросы настолько тяжёлые и ворочают такими объемами данных, что после нажатия на кнопку **Обновить всё (Refresh All)** вам приходится по несколько минут ждать их обновления, то стоит задуматься о том, как делать это автоматически и по расписанию.

Давайте рассмотрим следующий пример.

Допустим, что у нас есть книга Excel с запросом Power Query, который оптом загружает данные из всех файлов заданной папки, как мы это делали в главе [Массовая загрузка данных](#). При большом количестве исходных файлов время на обновление запроса будет ощутимо расти, поэтому хотелось бы обновлять его автоматически по заданному расписанию. Например, каждый день в 5:00, чтобы к нашему приходу на работу файл был уже готов.

В такой ситуации лучше воспользоваться **Планировщиком Windows** – специально встроенной в любую версию Windows программой, которая умеет по расписанию выполнять заданные действия. По факту вы уже используете его, сами того не зная, ведь ваш ПК регулярно проверяет обновления, качает новые антивирусные базы, синхронизирует облачные папки и т. д. – это всё работа **Планировщика**. Так что нам нужно просто добавить к уже имеющимся задачам ещё одну, которая будет запускать Excel и открывать в нём заданный файл. Затем мы с вами добавим команды VBA для обновления запросов в событие **Workbook_Open** этого файла – и проблема решена.

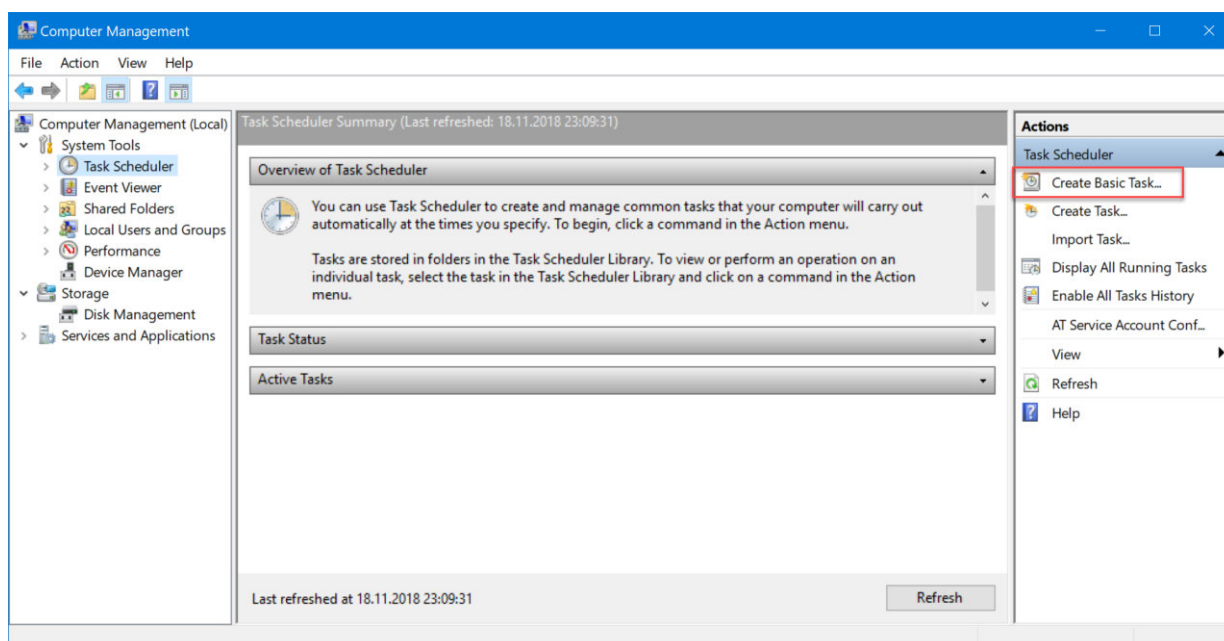
Хочу сразу предупредить, что для работы с **Планировщиком**, возможно, потребуются расширенные пользовательские права, поэтому, если вы не можете найти описанных ниже команд и функций у себя на рабочем компьютере в офисе, обратитесь за помощью к вашим IT-специалистам.

Запускаем Планировщик

Итак, давайте запустим **Планировщик**. Для этого можно выполнить любое из этих действий:

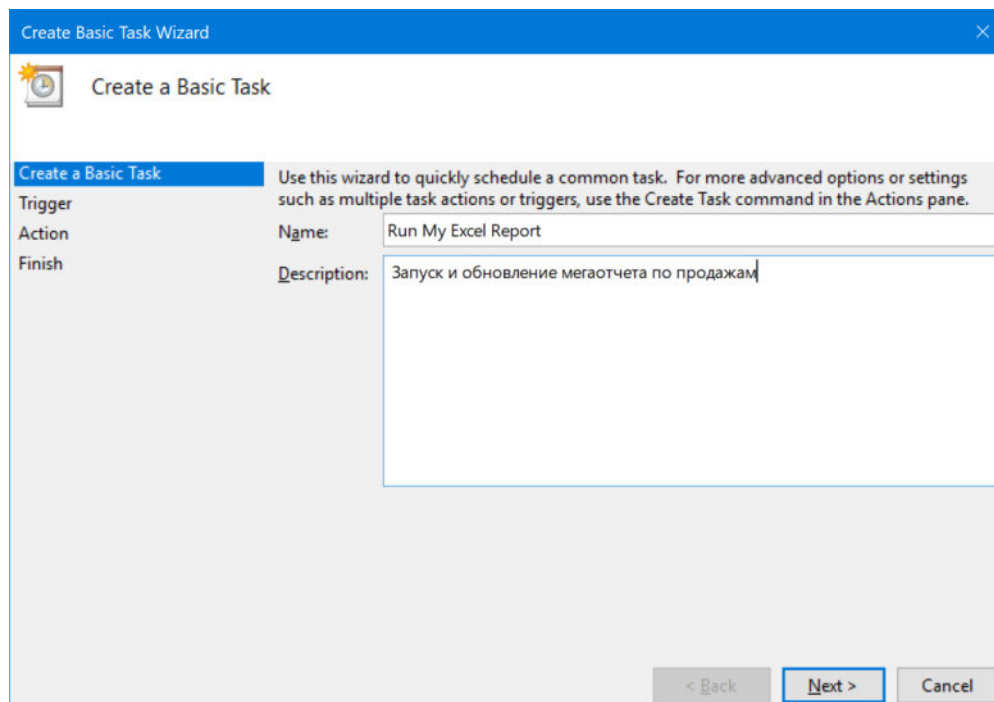
- щелкнуть правой кнопкой мыши по кнопке **Пуск** и выбрать **Управление компьютером (Computer management)**;
- выбрать в панели управления: **Администрирование → Планировщик заданий (Control Panel → Administrative Tools → Task Scheduler)**;
- выбрать в главном меню **Пуск → Стандартные → Служебные → Планировщик заданий**
- нажать сочетание клавиш **Win+R**, ввести **taskschd.msc** и нажать **Enter**.

На экране должно появиться примерно такое окно (у меня англоязычная версия, но если у вас русский Windows, то всё будет по-русски, естественно):

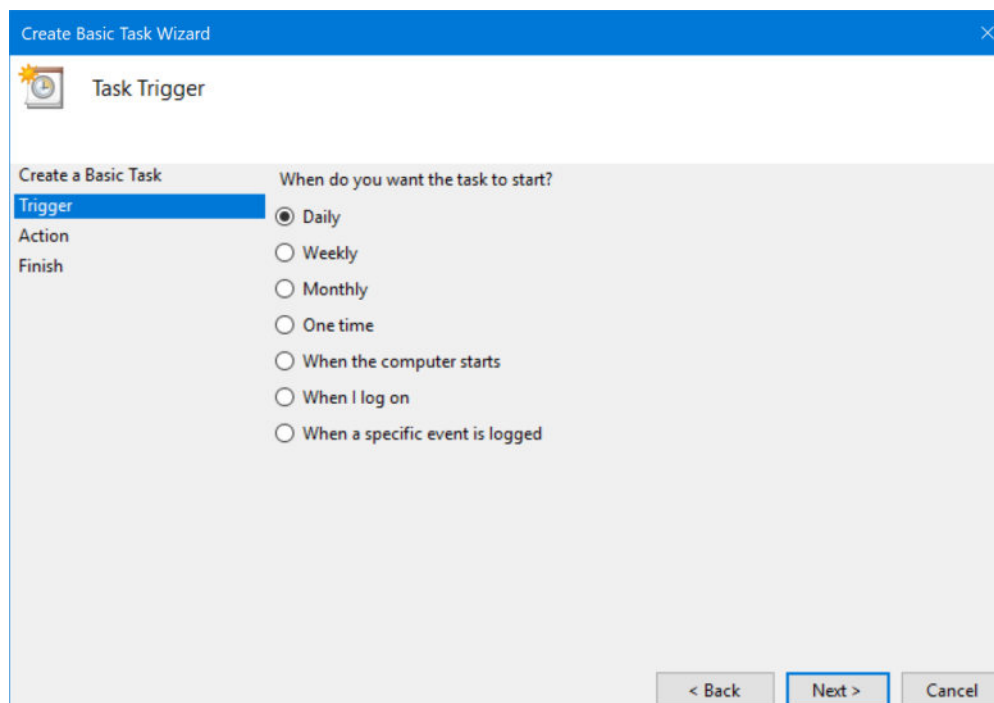


Создаем задачу

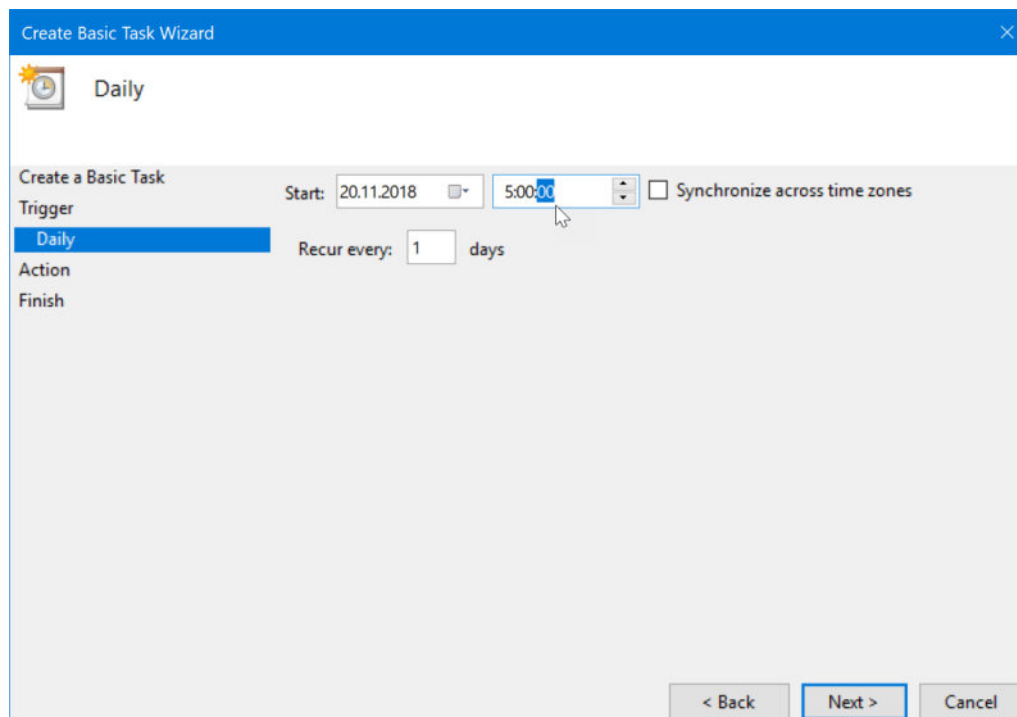
Чтобы создать новую задачу с помощью простого пошагового мастера, нажмем на ссылку **Создать простую задачу** (Create Basic Task) в правой панели. На первом шаге мастера нужно ввести название и описание создаваемой задачи:



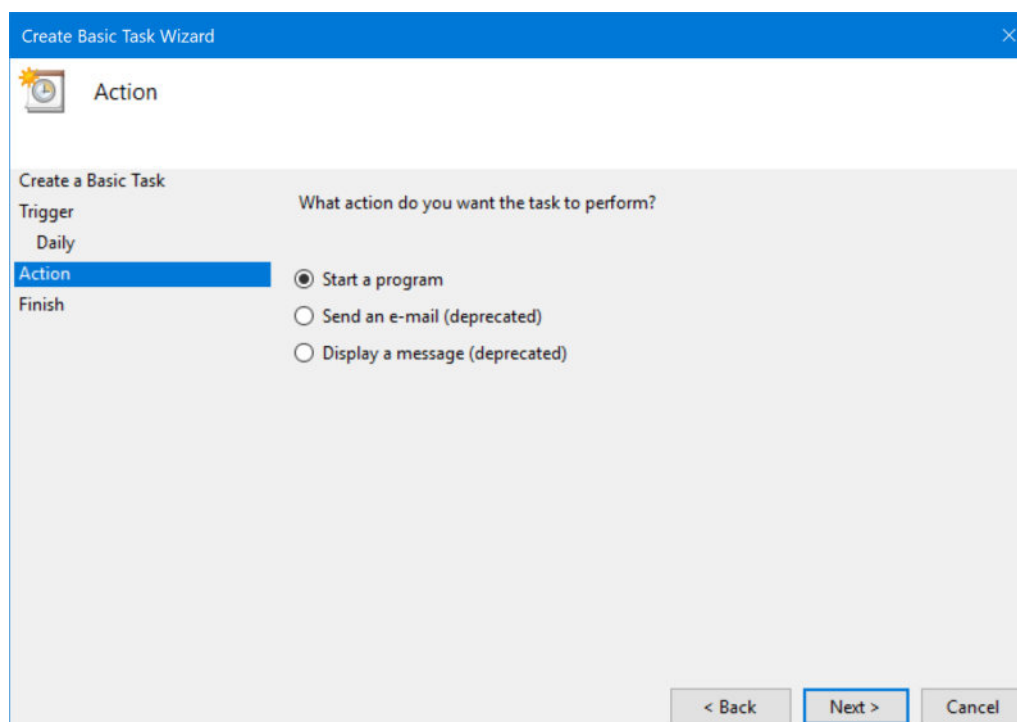
Жмем на кнопку **Далее** (Next) и на следующем шаге выбираем триггер – частоту запуска или событие, которое будет запускать нашу задачу (например, включение компьютера):



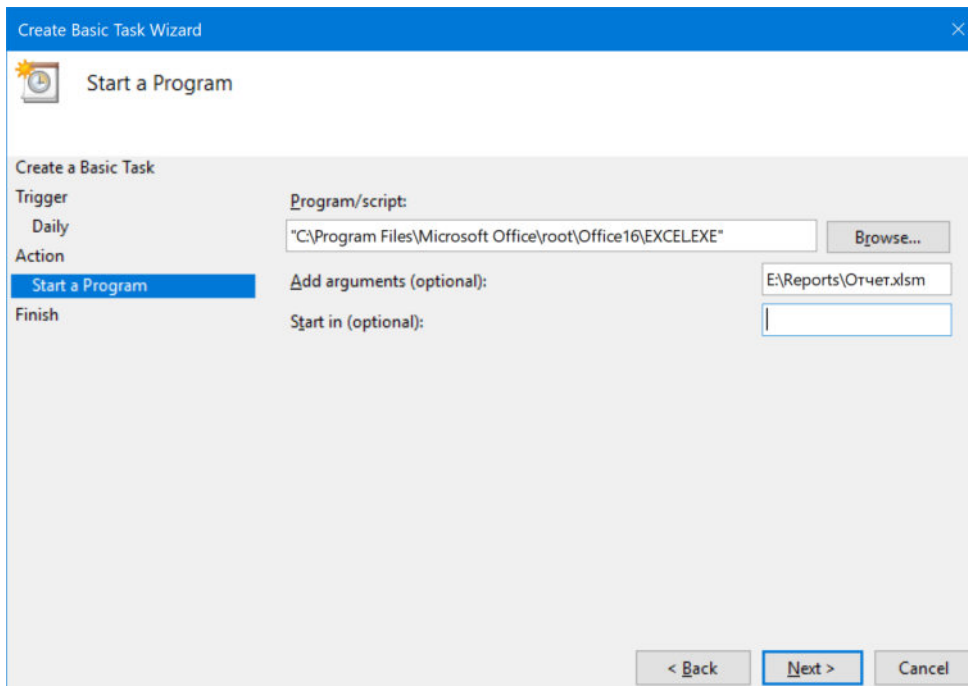
Если вы выбрали **Ежедневно** (Daily), то на следующем шаге нужно будет выбрать конкретное время, дату начала последовательности и шаг (каждый 2-й день, 5-й день и т. д.):



Следующий шаг – выбираем действие – **Запуск программы (Start a program)**:

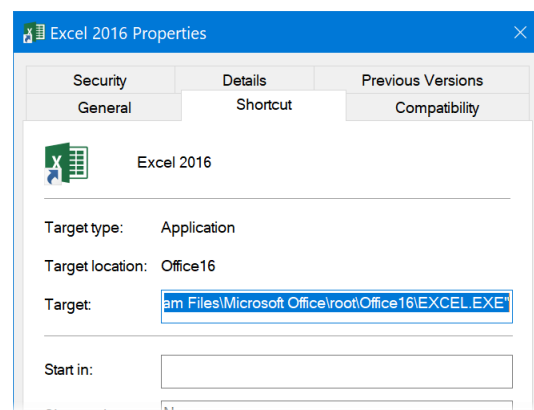
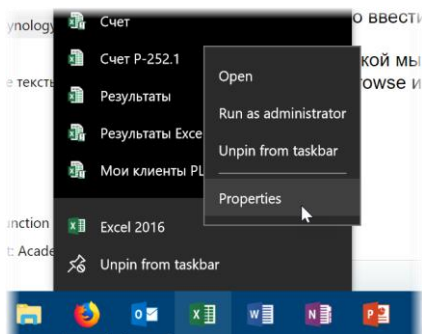


И, наконец, самое интересное – что именно нужно открывать:

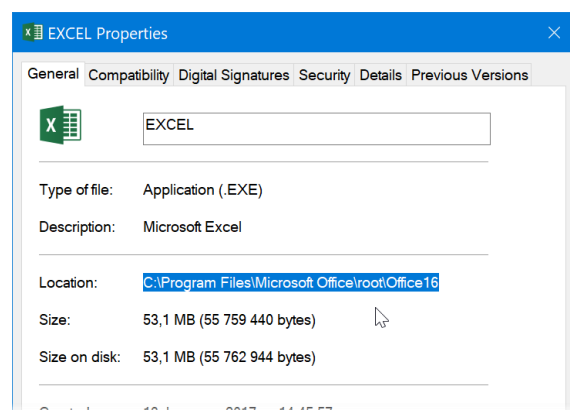
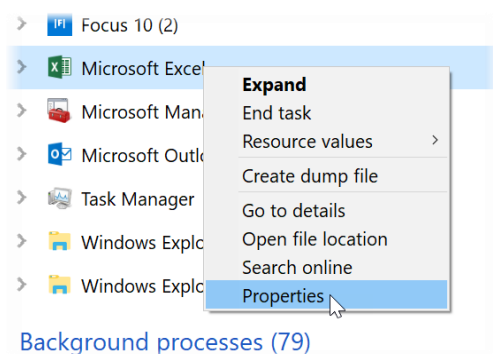


В поле **Программа или сценарий (Program/script)** нужно ввести путь к Microsoft Excel как к программе, т. е. непосредственно к исполняемому файлу Excel. На разных компьютерах с разными версиями Windows и Office этот файл может лежать в разных папках, поэтому вот вам несколько способов, как можно узнать его местоположение.

- Щелкнуть правой кнопкой мыши по иконке (ярлычку) запуска Excel на рабочем столе или в панели задач и выбрать команду **Свойства (Properties)**, а затем в открывшемся окне скопировать путь из строки **Target:**

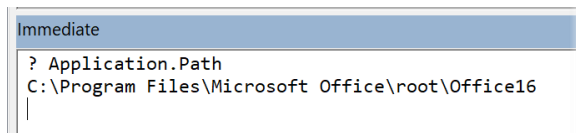


- Открыть любую книгу Excel, затем открыть **Диспетчер задач (Task Manager)** нажатием **Ctrl+Alt+Del** и, щелкнув правой кнопкой мыши по строке **Microsoft Excel**, выбрать команду **Свойства (Properties)**. В открывшемся окне можно скопировать путь, **не забыв потом дописать к нему обратный слеш и EXCEL.EXE в конце:**



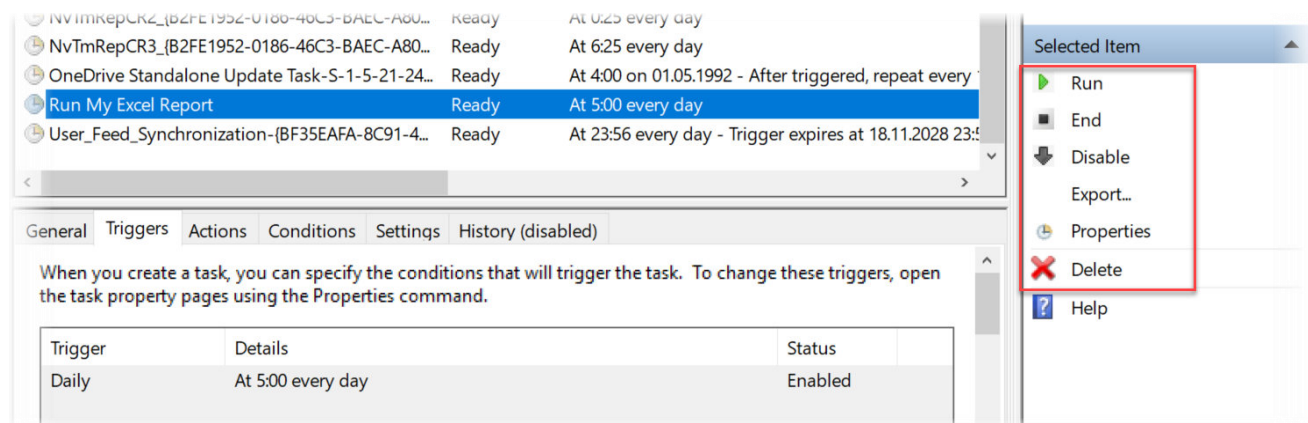
- Открыть Excel, открыть редактор Visual Basic сочетанием клавиш **Alt+F11**, открыть панель **Immediate** сочетанием **Ctrl+G**, ввести в неё команду **? Application.Path** и нажать на **Enter**

Скопировать получившийся путь, **не забыв потом дописать к нему обратный слэш и EXCEL.EXE в конце.**



В поле **Добавить аргументы (необязательно)** (Add arguments (optional)) нужно вставить полный путь к книге с макросом, которую мы хотим открыть.

Когда всё ввели, то жмем **Далее** и затем **Готово (Finish)**. Задача должна добавиться в общий список:



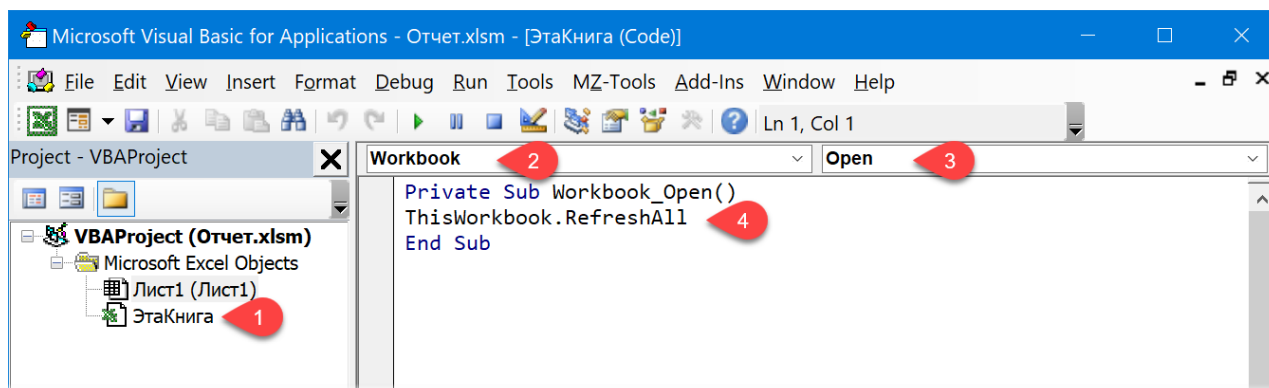
Управление созданной задачей удобно осуществлять с помощью кнопок справа. Здесь можно протестировать задачу, запустив её кнопкой **Выполнить (Run)**, не дожидаясь наступления заданного срока. Можно временно деактивировать задачу, нажав на **Отключить (Disable)**, чтобы она перестала выполняться на время, например, вашего отпуска. Ну, и изменить параметры (даты, время, имя файла) тоже всегда можно через кнопку **Свойства (Properties)**.

Добавляем макрос на открытие файла

Теперь осталось повесить в нашей книге запуск макроса обновления запросов Power Query на событие открытия файла. Для этого откроем книгу и перейдем в редактор Visual Basic с помощью сочетания клавиш **Alt+F11** или кнопки **Visual Basic** на вкладке **Разработчик (Developer)**. В открывшемся окне в левом верхнем углу нужно найти наш файл на дереве и двойным щелчком мыши открыть модуль **ЭтаКнига (ThisWorkbook)**.

Если у вас в редакторе Visual Basic не видно этого окна, то его можно открыть через меню **View → Project Explorer**.

В появившемся окне модуля добавим обработчик события открытия книги, выбрав его из выпадающих списков в верхней части **Workbook** и **Open** соответственно:



На экране должна появиться заготовка процедуры **Workbook_Open**, куда между строчками **Private Sub** и **End Sub** и нужно вставить те команды на VBA, которые должны автоматически выполняться при открытии этой книги Excel, когда её по расписанию откроет Планировщик.

Для обновления одним махом всех внешних запросов к данным, запросов Power Query и сводных таблиц можно использовать команду:

```
ThisWorkbook.RefreshAll
```

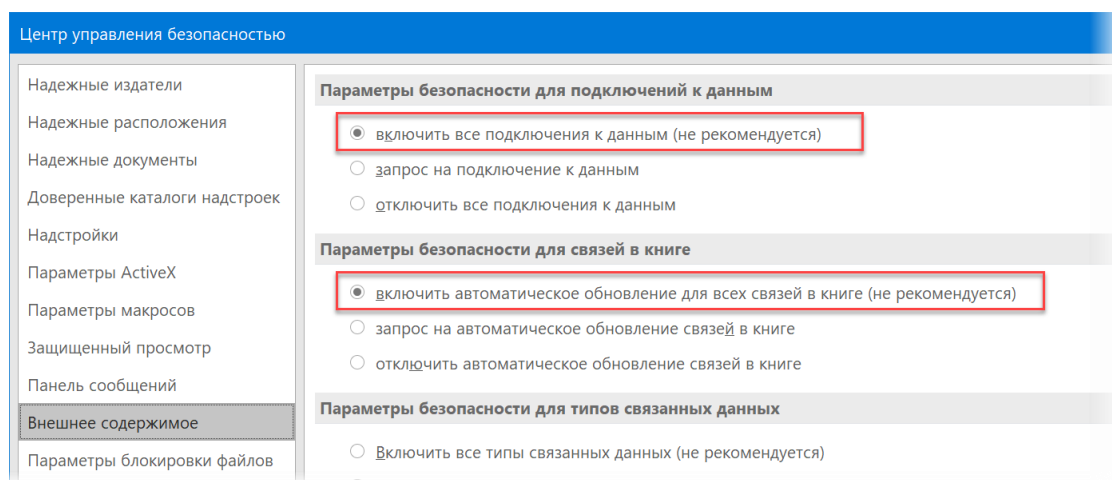
По сути, это равносильно нажатию кнопки **Обновить всё** на вкладке **Данные** (Data → Refresh All).

Если вам нужно, чтобы макрос запускал обновление только при открытии файла Планировщиком в 5 утра, а не каждый раз, когда вы тоже его открываете потом в течение рабочего дня, то имеет смысл добавить проверку на время, например:

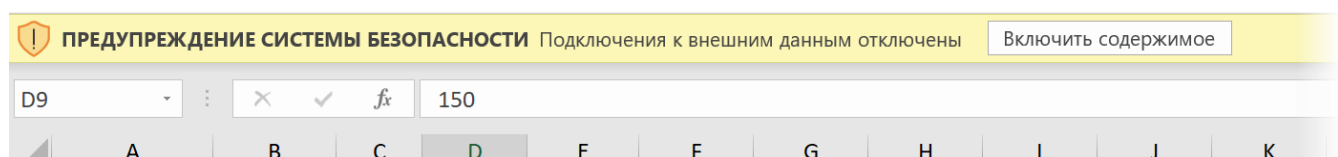
```
If Format(Now, "hh:mm") = "05:00" Then ThisWorkbook.RefreshAll
```

Отключаем защиту

Остался последний штрих. Необходимо разрешить по умолчанию подключения к внешним данным и обновление связей через **Файл → Параметры → Центр управления безопасностью → Параметры центра управления безопасностью → Внешнее содержимое** (File → Options → Trust Center → Trust Center Options → External Content),



Если этого не сделать заранее, то при открытии книги Планировщиком появится стандартное предупреждение и Excel, не продолжая и ничего не обновляя, будет ждать от вас благословения в виде нажатия на кнопку **Включить содержимое** (Enable content):



Вот и всё. Не забудьте сохранить книгу в формате с поддержкой макросов (xlsm или xlsb), и можно смело закрывать Excel и отправляться домой, оставив компьютер включенным. В заданный момент (даже если ПК заблокирован) Планировщик запустит Excel и откроет в нём заданный файл, а наш макрос выполнит обновление всех запросов. А вы будете нежиться в постели, пока ваш тяжелый отчёт автоматически пересчитывается. Красота!

Power Query и VBA

*Лично я вижу в этом перст судьбы — шли по лесу и встретили программиста.
(Аркадий и Борис Стругацкие, «Понедельник начинается в субботу»)*

Начиная с версии 2016 в Excel была добавлена поддержка управления запросами Power Query через макросы на VBA. Основную роль тут играют коллекции **ThisWorkbook.Queries** и **ThisWorkbook.Connections**, отвечающие за работу с запросами и подключения к данным соответственно. Встроенный в Excel 2016 макро-рекордер тоже научился записывать действия с запросами в виде готового кода на Visual Basic, что существенно облегчает жизнь непрограммистам.

Всё это позволяет автоматизировать рутинные действия пользователя в Power Query и значительно ускорить повседневную обработку данных. Давайте разберём основные сценарии такой автоматизации.

Удаление запросов макросом

Давайте начнем с простого: ломать не строить. Если вам нужно удалить определённый запрос в текущей книге, то это легко можно сделать макросом вида:

```
Sub Delete_One_Query()  
    ThisWorkbook.Queries("Запрос1").Delete  
End Sub
```

где **Запрос1** – это имя нужного вам запроса.

Если нужно удалить все запросы в текущей книге, то добавится цикл перебора всех элементов в коллекции с последующим удалением:

```
Sub Delete_All_Queries()  
    For Each pq In ThisWorkbook.Queries  
        pq.Delete  
    Next pq  
End Sub
```

Такое автоматическое удаление может пригодиться, например, когда вы собираетесь поделиться копией вашего файла, предварительно вычистив оттуда все запросы, чтобы не пугать других пользователей. Ну, или «замести следы», чтобы не показывать, что вы всё сделали не вручную за неделю, а за десять минут с помощью Power Query. ;)

Обновление запросов макросом

Если вы хотите обновить все запросы в книге, то это легко сделать буквально одной строкой на VBA:

```
Sub Refresh_All_Queries()  
    ThisWorkbook.RefreshAll  
End Sub
```

По сути, это равносильно нажатию кнопки **Обновить всё** на вкладке **Данные** (Data → Refresh All).

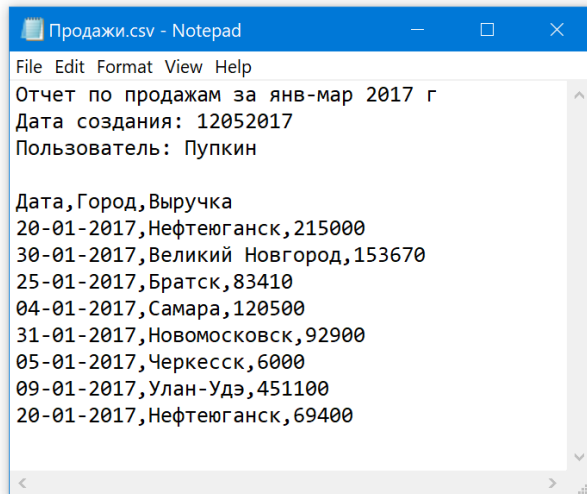
Если же вам необходимо обновить лишь один конкретный запрос, не затрагивая другие, то можно обратиться к нему по имени для персонального обновления:

```
Sub Refresh_One_Query()  
    ThisWorkbook.Connections("Запрос – Заказы").Refresh  
End Sub
```

где **Заказы** – имя нужного нам запроса.

Создание запроса макросом

Предположим, что мы хотим создать запрос макросом совсем с нуля. Будем импортировать простой файл **Продажи.csv** с разделителями-запятыми, который лежит в папке **D:\Отчеты** и выглядит вот так:



В подобной ситуации всегда полезно сначала один раз проделать импорт вручную, чтобы понять, как должен выглядеть исходный M-код запроса. Выберем на вкладке **Данные** → **Получить данные** → **Из файла** → **Из текстового/csv файла** (Data → Get Data → From file → From TXT/CSV) и пройдем короткий путь, состоящий из импорта данных в Power Query, удаления верхних четырех строк и повышения заголовков, чтобы получить причёсанную картинку:

	Дата	Город	Выручка
1	20.01.2017	Нефтеюганск	215000
2	30.01.2017	Великий Новгород	153670
3	25.01.2017	Братск	83410
4	04.01.2017	Самара	120500
5	31.01.2017	Новомосковск	92900
6	05.01.2017	Черкесск	6000
7	09.01.2017	Улан-Удэ	451100
8	20.01.2017	Нефтеюганск	69400

Теперь на вкладке **Просмотр** (View) воспользуемся кнопкой **Расширенный редактор** (Advanced Editor), чтобы посмотреть исходный код запроса на языке M:



Теперь, когда у нас есть общее понимание того, как должен выглядеть код, для написания макроса, который создает подобный запрос, можно использовать следующую заготовку на VBA:

```
Sub Macro_Template()
    qname = "Мой запрос"      'имя создаваемого запроса
    ActiveWorkbook.Queries.Add Name:=qname, Formula:="Тут должен быть М-код запроса"
End Sub
```

Обратите внимание, что код запроса нельзя просто скопировать из окна **Расширенного редактора** Power Query и вставить между кавычками после параметра Formula в VBA. Тут действуют следующие правила синтаксиса Visual Basic.

- Текст запроса должен быть склеен из фрагментов с использованием символов сцепки **&**, перед каждым и после каждого из которых должны стоять пробелы.
- Новая строка (т. е. имитация нажатия на клавишу **Enter**) делается приклеиванием спецсимвола с кодом 10 с помощью функции **Chr(10)**.
- Кавычки в исходном коде М-запроса (например, путь к файлу или названия заголовков столбцов) должны быть удвоенными.

```
Sub New_Query_By_Macro()

ActiveWorkbook.Queries.Add Name:="Продажи", Formula:= _
"let" & Chr(10) & _
"Источник = Csv.Document(File.Contents("D:\Выгрузки\Продажи.csv"),[Delimiter="","","" & _
    "Columns=3, Encoding=65001, QuoteStyle=QuoteStyle.None])," & Chr(10) & _
"УдаленныеВерхниеСтроки = Table.Skip(Источник,4)," & Chr(10) & _
"ПовышенныеЗаголовки = Table.PromoteHeaders(УдаленныеВерхниеСтроки, [PromoteAllScalars=true])," & _
    Chr(10) & _
"ИзмененныйТип = Table.TransformColumnTypes(ПовышенныеЗаголовки,{{"Дата", type date}," & _
    "{"Город", type text}, {"Выручка", Int64.Type}})" & Chr(10) & _
"in" & Chr(10) & _
"ИзмененныйТип"

End Sub
```

Завершая разбор этой темы, хотелось бы отметить следующее.

- Начиная с версии Excel 2016 макро-рекордер умеет записывать все описанные выше команды в VBA-код макросов автоматически. Так что можно просто включить запись на вкладке **Разработчик → Запись макроса (Developer → Record Macro)** и вручную проделать всю процедуру создания запроса и обработки данных. Полученный на выходе код будет чуть менее наглядным и замусоренным «лишними» командами, но вполне рабочим. Если же у вас пока Excel 2010–2013, то у вас только один вариант – писать код вручную.
- По умолчанию Power Query даёт имена шагам с пробелами, берёт их в кавычки и добавляет символ решётки в начале (например, **#"Повышенные Заголовки"**). Так как кавычки надо удваивать, то их количество в коде начнёт быстро расти и создавать проблемы с отладкой. Если переименовать шаги, убрав из них пробелы (**ПовышенныеЗаголовки**), то кавычки и решётки будут тоже не нужны, и внешний вид кода существенно упростится (см. предыдущий макрос).

Загрузка «умных» таблиц в Power Query макросом

Загрузка исходных данных в виде «умных» таблиц – один из основных механизмов работы в Power Query. Для загрузки одной таблицы, однако, придется проделать следующие действия:

1. Поставить активную ячейку в таблицу.

2. Нажать на вкладке **Данные** кнопку **Из таблицы/диапазона** (Data → From Table/Range).
3. Дождаться появления окна редактора запросов и выбрать в нём **Главная → Закрывать и загрузить → Закрывать и загрузить в...** (Home → Close&Load → Close&Load to...).
4. Дождаться появления окна вариантов загрузки и выбрать в нём **Только создать подключение** (Create only connection) и нажать **ОК**.

На все эти операции на среднестатистическом компьютере уходит примерно 10–15 секунд. А теперь представьте, что у вас десяток-другой таких таблиц, каждую из которых надо загрузить как отдельный запрос в Power Query для дальнейшей обработки. Делать это всё вручную долго, скучно, да и вероятность ошибки повышается с каждой итерацией. Для массового создания таких запросов к «умным» таблицам можно использовать следующие макросы.

Макрос загрузки всех «умных» таблиц с активного листа:

```
Sub Load_All_Tables_from_Active_Sheet_To_PowerQuery()
    Dim st As ListObject      'переменная для хранения ссылки на очередную умную таблицу

    'перебираем все умные таблицы на текущем листе
    For Each st In ActiveSheet.ListObjects
        'удаляем запрос с таким именем, если он уже существует
        On Error Resume Next
        ActiveWorkbook.Queries(st.Name).Delete
        On Error GoTo 0

        'создаем запрос с именем умной таблицы
        ActiveWorkbook.Queries.Add Name:=st.Name, _
            Formula:="let Источник = Excel.CurrentWorkbook(){[Name="" & _
                st.Name & """]}[Content] in Источник"
    Next st
End Sub
```

Если нужно загрузить в Power Query все умные таблицы в книге, то к предыдущему макросу добавится цикл перебора листов:

```
Sub Load_All_Tables_from_Workbook_To_PowerQuery()
    Dim st As ListObject
    Dim ws As Worksheet      'переменная для хранения ссылки на очередной лист

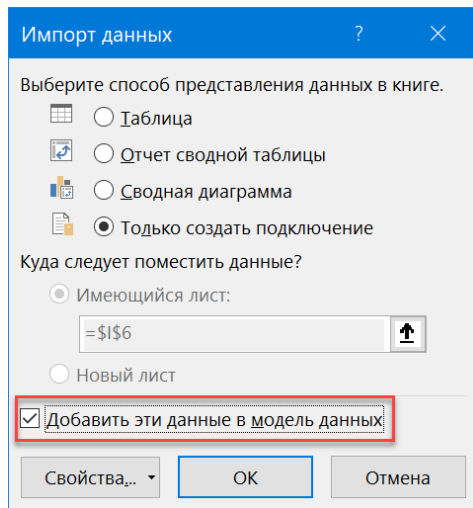
    For Each ws In ActiveWorkbook.Worksheets
        For Each st In ws.ListObjects
            On Error Resume Next
            ActiveWorkbook.Queries(st.Name).Delete
            On Error GoTo 0
            ActiveWorkbook.Queries.Add Name:=st.Name, _
                Formula:="let Источник = Excel.CurrentWorkbook(){[Name="" & _
                    st.Name & """]}[Content] in Источник"
        Next st
    Next ws
End Sub
```

Загрузка запросов Power Query в Модель Данных Power Pivot макросом

Если у вас есть запрос, результаты которого нужно выгрузить не на лист, а загрузить в Модель Данных Power Pivot, то вручную это потребует следующих действий.

Щелкнуть по запросу правой кнопкой мыши в окне запросов и подключений (справа) и выбрать команду **Загрузить в...** (Load to...).

В появившемся окне вариантов выгрузки включить флажок **Добавить эти данные в модель данных** (Add this data to Data Model):



Чтобы проделать это не вручную, а макросом, потребуется создать для нашего запроса новое подключение с загрузкой в Power Pivot:

```
Sub Load_One_Query_To_DataModel()
    qname = "Клиенты"           'имя запроса, который хотим загрузить в Модель Данных'

    ThisWorkbook.Connections.Add2 "Запрос - " & qname, _
        "Соединение с запросом '" & qname & "' в книге.", _
        "OLEDB;Provider=Microsoft.Mashup.OleDb.1;Data Source=$Workbook$;Location=" & qname _
        , """" & qname & """"", 6, True, False
End Sub
```

Если же нужно загрузить в Модель Данных все запросы из текущей книги, то добавится цикл для их перебора:

```
Sub Load_All_Queries_To_DataModel()
    Dim pq As WorkbookQuery    'переменная для хранения ссылки на очередной запрос'

    For Each pq In ActiveWorkbook.Queries
        ThisWorkbook.Connections.Add2 "Запрос - " & pq.Name, _
            "Соединение с запросом '" & pq.Name & "' в книге.", _
            "OLEDB;Provider=Microsoft.Mashup.OleDb.1;Data Source=$Workbook$;Location=" & pq.Name _
            , """" & pq.Name & """"", 6, True, False
    Next pq
End Sub
```

При этом надо иметь в виду, что при повторном запуске этих макросов в Модели Данных будет создан дубликат таблицы с тем же именем и порядковым номером (1,2,3...).

Язык M

До сих пор мы создавали запросы в основном через интерфейс, т. е. пользуясь только кнопками на ленте и вкладках редактора. Теперь давайте попробуем зайти с другой стороны и заглянуть за кулисы Power Query, обратив фокус на внутренний язык M, который заложен в основу всего процесса.

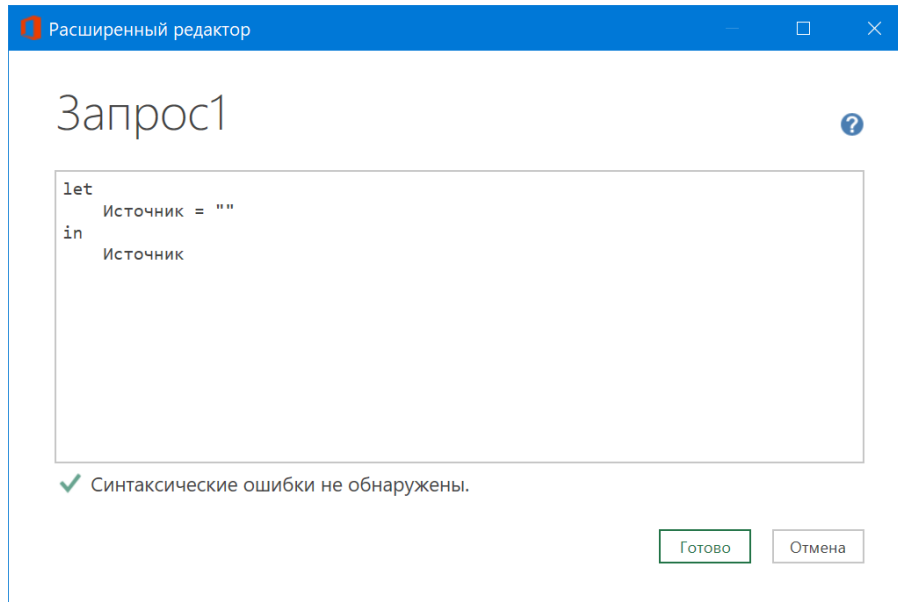
В этой главе мы:

- разберём **основы синтаксиса языка M**, его базовые программные конструкции, ключевые слова и команды;
- пробежимся по всем **типам данных**, которые есть в Power Query;
- будем **писать код запросов с нуля**, как заправские программисты;
- научимся создавать **пользовательские функции** на языке M, если вам вдруг не хватает встроенных;
- выясним, как можно **ссылаться на отдельные объекты** (строки, столбцы, ячейки) в запросах Power Query.



Основы синтаксиса языка M

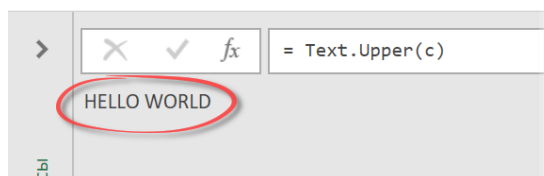
Выберем на вкладке **Данные** команду **Получить данные** → **Из других источников** → **Пустой запрос** (Data → Get Data → From Other Sources → Blank Query). В открывшемся окне редактора Power Query нажмем кнопку **Расширенный редактор** на вкладке **Просмотр** (View → Advanced Editor), чтобы увидеть исходный код нашего пустого пока запроса:



Введём сюда следующий код:

```
let
    a = "hello",
    b = "world",
    c = a & " " & b,
    d = Text.Upper(c)
in
    d
```

Как легко сообразить, механика этого запроса проста: мы склеиваем два слова через пробел и преобразуем получившийся текст в верхний регистр. После нажатия на кнопку **Готово** в окне предварительного просмотра отобразится результат:



Поздравляю, вы только что написали свой первый запрос с нуля – напрямую на языке M! Теперь давайте копнём чуть глубже и разберём основы синтаксиса этого языка и его фундаментальные принципы.

Выражения

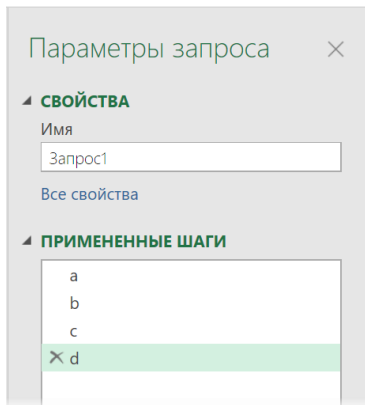
Основой любого запроса являются **выражения** (expressions) – текстовые строки, где в левой половине стоит имя переменной (в нашем примере это **a**, **b**, **c** и **d**), а в правой – присваиваемое ей значение. Значение может быть как константой, например:

```
a = "hello"
```

... так и функцией, например:

```
d = Text.Upper(c)
```


Имена переменных отображаются в правой панели редактора как названия применённых шагов:



Если в именах переменных есть пробелы, то их нужно заключать в кавычки и ставить перед именем «решётку», например:

```
let
    #"Первое слово" = "hello",
    #"Второе слово" = "world",
    c = #"Первое слово" & " " & #"Второе слово",
    d = Text.Upper(c)
in
    d
```

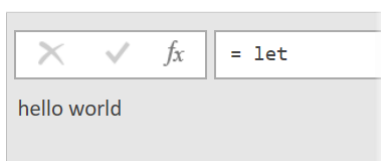
Имена переменных обязательно должны быть уникальными, иначе мы получим ошибку и наш запрос не будет выполнен.

Оператор let

Конструкция **let ... in** используется для группировки нескольких выражений и возвращает в качестве результата значение той переменной, которая указана после **in**. В большинстве случаев нам нужно получить то, что получается на выходе после выполнения последнего шага, поэтому после **in** часто стоит название последней переменной (в нашем случае это **d**). Однако же это не является обязательным требованием, и таким образом вполне можно вывести содержимое любых других переменных. Например, запрос:

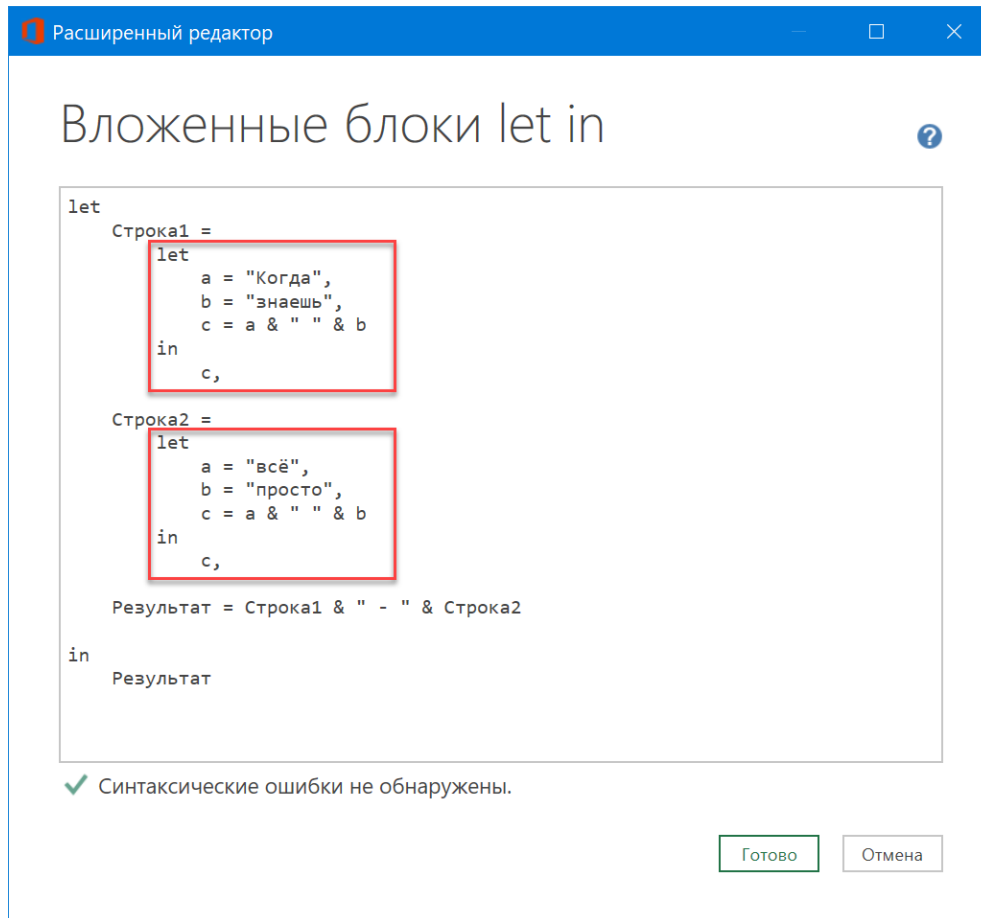
```
let
    a = "hello",
    b = "world",
    c = a & " " & b,
    d = Text.Upper(c)
in
    c
```

...вернёт в качестве результата значение третьей переменной (**c**), т. е. склейку ещё до преобразования в прописные буквы:

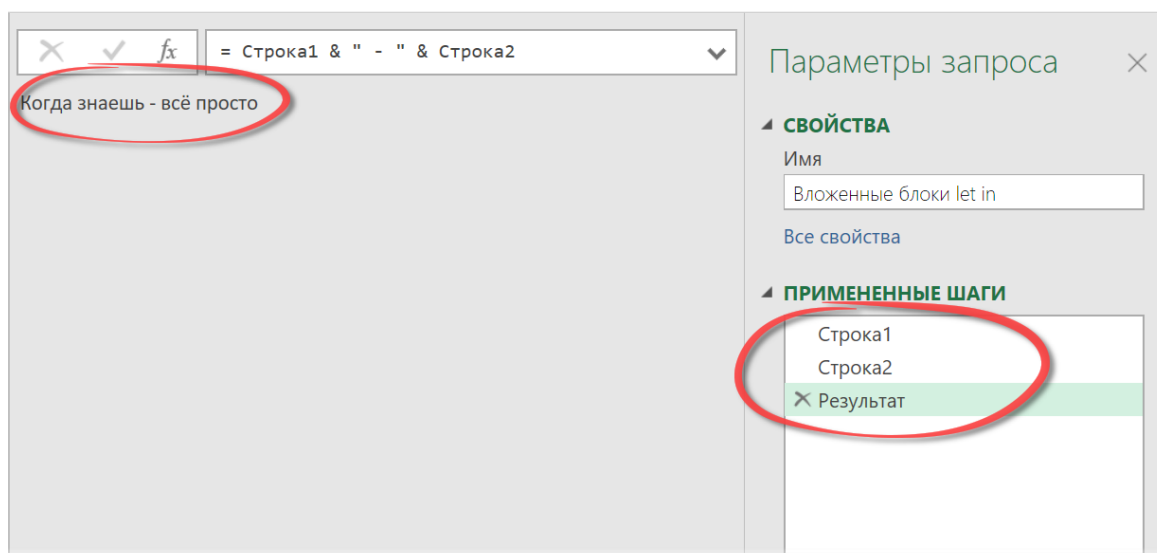


Если в блоке **let ... in** содержится несколько выражений, то они должны разделяться запятыми, т. е. необходимо ставить запятую после каждой строки в блоке, кроме последней.

Также блоки **let ... in** могут быть вложенными друг в друга, что позволяет делать при необходимости многоуровневые иерархические структуры в сложных запросах:



После нажатия на **Готово** такой запрос выдаст моё любимое:



Как вы можете заметить, имена переменных внутри разных блоков **let ... in** запроса могут повторяться. Это не будет проблемой, т. к. в языке M область действия переменных ограничена их блоком.

Такой подход бывает очень удобен, когда вы собираете свой запрос из нескольких других и копируете куски кода из других запросов в свой. Чтобы не переименовывать чужие переменные, можно просто заключить каждый скопированный кусок кода в свой блок **let ... in**, и проблема повторяющихся имён переменных будет элегантно решена.

Есть, однако, и минусы: в панели справа в таком случае отображаются уже не все шаги нашего запроса, а только верхний уровень созданной иерархии блоков.

Комментарии

*Всегда пишите код так, будто сопровождать его будет склонный к насилию психопат, который знает, где вы живете.
(Мартин Голдинг)*

И, кстати, об удобстве чтения и отладки написанного M-кода. Как и в других языках программирования, здесь есть возможность оставлять в коде примечания и комментарии с пояснениями к тем или иным фрагментам. Однострочные комментарии создаются с помощью двух подряд знаков дроби (`//`). Такое примечание может быть как единственным в строке, так и стоять справа от уже имеющегося выражения. Всё, что в строке правее этих символов, воспринимается как комментарий:

```
let
    nds = 0.2,          //это новый НДС
    price = 100,
    //вычисляем цену с НДС
    listprice = price * nds
in
    listprice
```

Если же необходимо сделать многострочное примечание, то оно обрамляется символами `/*` и `*/` соответственно:

```
/* этот запрос вычисляет
площадь круга заданного радиуса */
let
    r = 5,
    area = r * r * 3.14
in
    area
```

Кроме, собственно, создания комментариев можно использовать эти спецсимволы для временной нейтрализации кусков кода во время отладки сложных запросов.

Последовательность выполнения

Чтобы до конца прояснить ситуацию, необходимо упомянуть ещё один нюанс. На самом деле последовательность шагов-выражений между операторами `let` и `in` не играет никакой роли. Например, результатом вот такого запроса:

```
let
    b = "world",
    d = Text.Upper(c),
    a = "hello",
    c = a & " " & b
in
    d
```

... будет всё то же самое **HELLO WORLD**.

Это выглядит немного странно и отличается от большинства других языков программирования, где команды выполняются именно в том порядке, в котором они написаны. Дело в том, что технически для Power Query наши команды в запросе – это просто набор переменных, каждой из которых присваивается какое-то значение.

При этом совершенно не важно, в какой именно строке кода это присвоение происходит. В случае такой перестановки, как и с вложенными блоками **let ... in**, мы, к сожалению, уже не увидим наглядной последовательности шагов в правой панели, но работе запроса это не мешает.

Программа начинает «раскручивать» наш запрос с конца – от той переменной, что указана после **in** (в нашем случае это **d**). Если для её вычисления нужна другая переменная (у нас это **c**), то алгоритм сначала пытается вычислить её. Но, чтобы получить это значение, нужно сначала вычислить **a** и **b**, поэтому происходит переход на соответствующие строки, где эти переменные задаются, и так далее, пока Power Query не дойдёт до начала этой цепочки.

Не стоит, однако, злоупотреблять этим свойством и перемешивать выражения без крайней надобности: хотя на выполнении запроса это и не отразится, но читать и разбирать такой код будет очень тяжело.

Логические ветвления с if ... then ... else

Аналогом экселевской функции **ЕСЛИ** (IF) в Power Query является конструкция **if ... then ... else**, используемая для проверки заданных условий и выполнения затем различных действий в зависимости от результатов проверки, т. е. организации ветвлений для обработки данных по разным сценариям.

Синтаксис этой конструкции в языке M прост:

```
=if (условие) then Выражение1 else Выражение2
```

Например:

```
=if [Баланс]>1000 then «Хорошо» else «Не очень»
```

Можно вкладывать такие конструкции друг в друга для проверки нескольких условий, причем очень рекомендуется в подобных случаях разделять такие «матрёшки» на несколько строк в коде для наглядности, например:

```
=if (условие1) then
    Выражение1
else if (условие2) then
    Выражение2
else
    Выражение3
```

В условиях можно использовать символы сравнения >, <, >=, <=, <> и комбинировать сразу несколько критериев, используя операторы **and**, **or**, **not** и скобки для определения последовательности проверок.

Если вы используете конструкцию **if ... then ... else** внутри оператора **let ... in**, то не забудьте присвоить проверочную конструкцию какой-либо переменной, т. е. код:

```
let
    if (условие) then
        Result = 1
    else
        Result = 0
in
    Result
```

... выдаст ошибку и работать не будет. Нужно его переписать следующим образом:

```
let
    Result =
        if (условие) then
            1
        else
            0
in
    Result
```

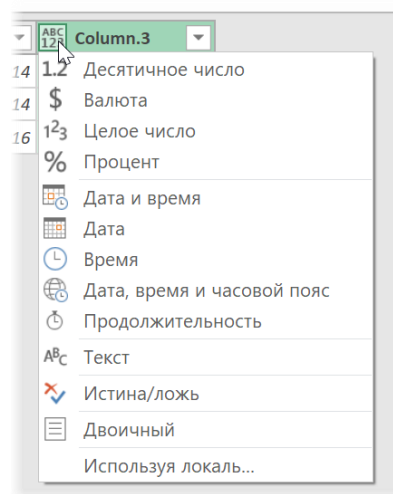
Простые типы данных

Всегда начинайте со структуры данных.
(Джон Кармак, автор игры «DOOM»)

Основой любого языка программирования является набор поддерживаемых *типов данных*, т.е. объектов, которыми мы можем оперировать в рамках наших программ (запросов). В этой главе мы разберём все основные типы, которые есть в языке M в Power Query. У каждого типа есть свои допустимые наборы значений, характерные синтаксические конструкции, наборы операторов и функции, которые можно к нему применить.

Все типы условно делятся на две большие группы – *простые (primitive)* и *структурированные (structured)*.

Простыми типами называют базовые типы данных, применяемые к одиночным значениям: числовой, текстовый, дата, время и т.д. Давайте рассмотрим их подробнее.



Числовой (number)

Этот простой тип обозначается кодовым словом **number** и используется для числовых и арифметических операций. Значения такого типа можно задавать в разном виде:

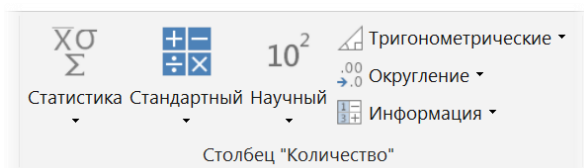
- как целые числа, например: **1, -2, 100**;
- как десятичные дроби, например: **7.5, 9.99** (между целой и дробной частью ставится точка или запятая, в зависимости от текущих региональных настроек вашего компьютера);
- в экспоненциальном виде: **2.5e6** (т.е. $2.5 \times 10^6 = 2\,500\,000$);
- в шестнадцатеричном формате, например **0xfb** (это 251 в привычной нам системе счисления).

К числовым данным применимы знакомые нам по Excel операторы сравнения **>**, **<**, **>=**, **<=**, **<>** и базовые арифметические действия **+**, **-**, **/** и *****.

Язык M содержит отдельную категорию с большим количеством числовых функций для обработки данных такого типа. Среди них можно выделить:

- вычислительные функции: **Number.Power** (возведение в степень), **Number.Sqrt** (квадратный корень), **Number.Abs** (значение по модулю) и т.д.;
- функции округления: **Number.Round**, **Number.RoundUp**, **Number.RoundDown** и т.д.;
- проверочные функции: **Number.IsEven**, **Number.IsOdd** (проверка на чётность-нечётность);
- тригонометрические функции: **Number.Sin**, **Number.Cos**, **Number.Tan** и т.д.

Почти все эти функции доступны также через интерфейс пользователя в редакторе Power Query на вкладке **Преобразование (Transform)**:



Полный список всех функций и их описания можно найти тут <https://docs.microsoft.com/en-us/powerquery-m/number-functions>.

Текстовый (text)


Это прямой родственник текстового формата из Microsoft Excel – тип данных, обозначаемый в M-коде как **text** и предназначенный для хранения текста, или, другими словами, набора символов. Несколько текстовых строк можно, как и в Excel, склеивать в одну общую строку с помощью оператора конкатенации **&** (амперсанд), например:

```
= "Маша" & " ела " & "кашу"
```

При этом нельзя подобным образом сцеплять данные разных типов. Например, выражение:

```
= "Зона " & 51
```

... вернет ошибку,

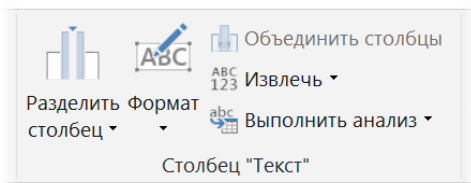
 Expression.Error: Не удается применить оператор & к типам Text и Number.

...т. к. недопустимо склеивать текст и числа. Правильным решением будет использование специальной функции **Text.From** для преобразования типа из числового в текстовый перед склейкой:

```
= "Зона " & Text.From(51)
```

Power Query имеет большое количество M-функций для обработки текста: тут есть как аналоги экселевских привычных **ЛЕВСИМВ** (LEFT), **ПРАВСИМВ** (RIGHT), **ПСТР** (MID), **ДЛСТР** (LEN), **ПОИСК** (FIND) и т. д., так и уникальные, доступные только в языке M функции. Полный список можно найти на странице <https://docs.microsoft.com/en-us/powerquery-m/text-functions>.

Некоторые функции доступны без программирования прямо из пользовательского интерфейса через команды группы **Столбец «Текст»** с вкладки **Преобразование** (Transform):



Продолжительность (duration)

Продолжительность – это особый тип данных, представляющий собой числовое значение длительности (продолжительности) некоего процесса или события, т. е. разницу между двумя значениями даты-времени на временной оси. Задавать такие значения можно с помощью специального выражения **#duration(дней, часов, минут, секунд)**, например:

```
=#duration(2, 5, 30, 0)
```

Также можно задавать длительность текстовой строкой вида "d.hh:mm:ss", используя функцию **Duration.FromText**, например:

```
=Duration.FromText("2.05:30:00")
```

В обоих случаях это будет значить «2 дня 5 часов 30 минут 0 секунд» соответственно.

Наименьшим интервалом длительности в Power Query принята 0.1 микросекунда (0.0000001 сек.), или *тик* (*tick*).

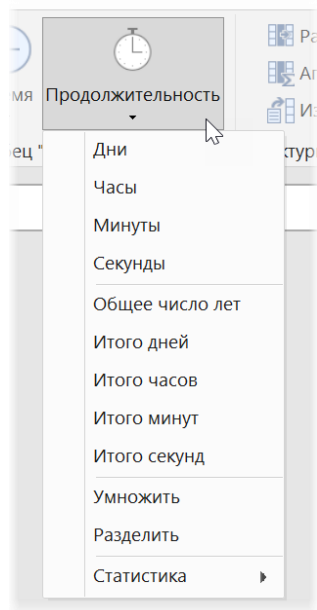
Значения длительности могут быть как положительными, так и отрицательными, т. е. отображающими сдвиг в прошлое относительно начального момента.

Значения этого типа данных можно прибавлять и вычитать из даты и времени (см. следующий пункт), сравнивать между собой с помощью операторов >, <, >=, <=, <> и применять к ним базовые арифметические операции. Например:

```
=#duration(2, 10, 30, 0)/2
```

выдаст в качестве результата значение вдвое меньшей продолжительности, т. е. 1.05:00:15:00 (1 день 5 часов 15 минут 0 секунд).

Большинство функций для работы с этим типом данных можно найти в выпадающем списке **Продолжительность (Duration)** на вкладке **Преобразование (Transform)**. Полный же список всегда доступен на сайте Microsoft по ссылке <https://docs.microsoft.com/en-us/powerquery-m/duration-functions>.



Дата (date)

Power Query умеет работать с датами в интервале 01.01.0001 - 31.12.9999. В языке M значения типа даты создаются с помощью выражения **#date(год, месяц, день)**. Например:

```
=#date(1977, 3, 18)
```

... или из текстовой строки с помощью функции **Date.FromText**:

```
=Date.FromText("18.03.1977")
```

...или:

```
=Date.FromText("19770318")
```

Что во всех случаях вернёт нам 18 марта 1977 года соответственно.

Как и числа, даты поддерживают операторы **>**, **<**, **>=**, **<=**, **<>**, если нужно сравнить, например, какая из двух заданных дат раньше другой, и т. п. Также можно вычитать даты, если нужно получить разницу между ними в типе **Продолжительность (Duration)**:

	Старт	Финиш
1	12.07.2014	15.07.2014
2	27.03.2014	18.04.2014
3	07.11.2016	28.11.2016

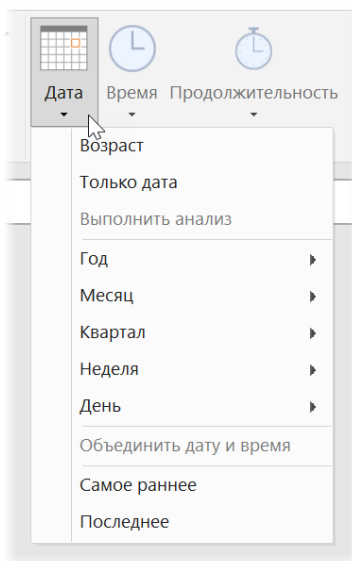
Настраиваемый столбец

Имя нового столбца

Пользовательская формула столбца:

Power Query имеет в своём распоряжении весьма приличный список M-функций для выполнения различных операции над данными типа даты. Большинство из них опять же доступны через интерфейс пользователя на

вкладке **Преобразование** → **Дата** (Transform → Date), и мы уже разбирали их работу в главе [Преобразование дат](#):



Однако некоторые функции в этом списке отсутствуют, и использовать их можно только в M-коде, например:

- **Date.AddDays**(исходная_дата, N)¹ – сдвигает исходную дату на N дней в будущее или прошлое (если N отрицательное). Совершенно аналогично работают функции сдвига на N месяцев, кварталов, недель или лет: **Date.AddMonths**, **Date.AddQuarters**, **Date.AddWeeks** и **Date.AddYears**;
- **Date.IsInCurrentWeek**(дата) – проверяет, попадает ли заданная дата в текущую неделю, и выдает на выходе логическую истину или ложь. Аналогично работают функции проверки вхождения в текущий месяц, квартал или год: **Date.IsInCurrentMonth**, **Date.IsInCurrentQuarter**, **Date.IsInCurrentYear**.

Полный список всех функций можно найти на сайте Microsoft <https://docs.microsoft.com/en-us/powerquery-m/date-functions>.

Время (time)

Как мы уже упоминали, наименьшим временным интервалом в Power Query является 0.1 микросекунда, или *тик*. Внутреннее представление времени представляет собой количество таких тиков начиная с полуночи. Соответственно, минимальным и максимальным значениями времени являются **0:00:00** и **23:59:9999999**.

Как и даты, значения в типе времени задаются с помощью специального выражения **#time(час, мин, сек)**, например:

```
=#time(15, 45, 58)
```

или с помощью функции преобразования из текстовой строки:

```
=Time.FromText("15:45:58")
```

что вернёт значение 15:45:58 или 2:45:58 PM в 12-часовом AM/PM формате.

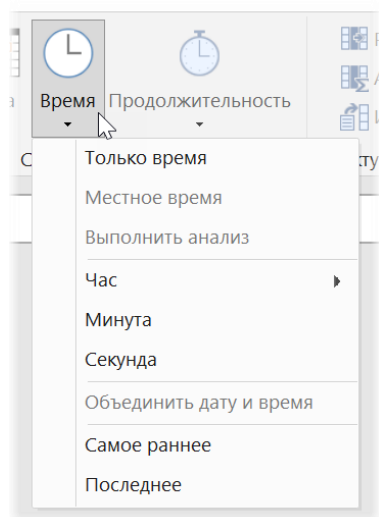
Как и дата, время тоже поддерживает логические операторы сравнения **>**, **<**, **>=**, **<=**, **<>** и базовые математические действия (сложение, вычитание) со значениями типа **Продолжительность** (*Duration*). Например:

```
=#time(13, 10, 50) + #duration(0, 4, 5, 3)
```

выдаст 17:15:53.

Встроенных M-функций для работы с временем не очень много, и почти все они доступны через выпадающий список **Время** (Time) на вкладке **Преобразование** (Transform):

¹ См. главу [Сдвиг даты на N периодов](#).



Логический тип (logical)

Значения этого типа используются при выполнении логических (булевых) операций и могут принимать значения только **TRUE** или **FALSE**. Обычно они являются результатом операций сравнения двух значений с помощью $>$, $<$, $>=$, $<=$, $<>$ или результатом логических манипуляций типа **Выражение1 and Выражение2** или **Выражение1 or Выражение2** и т. п. Для определения переменных такого типа используется специальное слово **logical**.

Тип null

Представьте, что вы смотрите на лист Microsoft Excel и видите на нём пустую ячейку. Действительно ли в ней ничего нет? Не факт! Есть масса вариантов того, что там может быть на самом деле: текст «белым на белом», невидимый глазу пробел, формула, выдающая в результате пустую строку «», и т. п.

В Power Query для таких случаев есть особый тип данных (и значение) – **null**, который используется для обозначения отсутствия какого-либо значения.

Интересной и неочевидной особенностью этого типа данных является то, что при сравнении null с другими типами данных с помощью операторов $>$, $<$, $>=$, $<=$, $<>$ на выходе мы тоже получим **null**, а не ожидаемое обычно логическое **TRUE** или **FALSE**. На практике это означает, что если в наших данных есть null, то эти значения нужно «отлавливать» отдельно.

Предположим, что у нас есть вот такая загруженная в Power Query таблица данных с вкраплениями **null**:

	ABC 123	Баланс
1		94
2		-60
3		null
4		31
5		-70
6		12

Допустим, нам нужно вывести слово *Хорошо*, если баланс положительный, и *Плохо*, если отрицательный. Если создать вычисляемый столбец через **Добавление столбца → Настраиваемый столбец** (Add Column → Custom Column) и использовать в нём формулу:

```
=if [Баланс]>0 then "Хорошо" else "Плохо"
```

... то мы получим ошибку на тех ячейках, где значения баланса были **null**:

	123 Баланс	ABC 123 Проверка
1	94	Хорошо
2	-60	Плохо
3	null	Error
4	31	Хорошо
5	-70	Плохо
6	12	Хорошо

Чтобы ошибка не возникала, нужна дополнительная проверка на *null* с помощью конструкции **if ... then ... else**, которую мы уже упоминали:

```
if [Баланс] = null then
    "Пусто"
else
    if [Баланс]>0 then
        "Хорошо"
    else
        "Плохо"
```

Это выдаст уже более приятную картину:

	123 Баланс	ABC 123 Проверка
1	94	Хорошо
2	-60	Плохо
3	null	Пусто
4	31	Хорошо
5	-70	Плохо
6	12	Хорошо

Структурированные типы данных

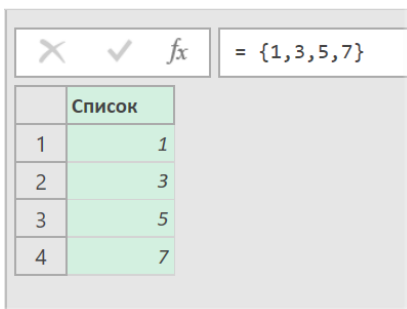
В отличие от разобранных выше простых типов *структурированные* или *комплексные* типы состоят не из одного, а из множества значений. К таким типам относятся списки (Lists), записи (Records) и таблицы (Tables).

Список (list)

Тип данных *список* – это одномерный перечисляемый набор элементов. Проще всего представить себе список как таблицу, состоящую из единственного столбца с данными. Элементами списка могут быть совершенно любые значения: числа, текст, даты, вложенные списки или таблицы и т. д.

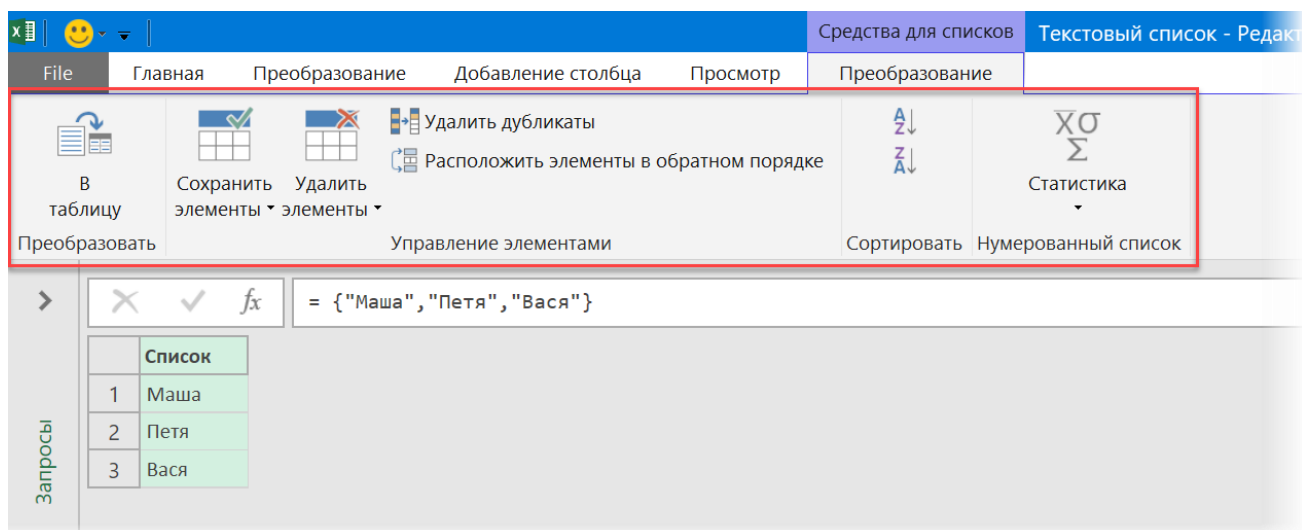
Создание списков

Для создания списков в языке M можно использовать фигурные скобки, перечисляя в них элементы списка через запятую, например:



	Список
1	1
2	3
3	5
4	7

или:



	Список
1	Маша
2	Петя
3	Вася

Обратите внимание, что при работе со списком на ленте редактора Power Query появляется новая контекстная вкладка **Средства для списков** (Tools for lists), содержащая базовые инструменты для обработки списков: удаление ненужных элементов, дубликатов, разворот списка в обратном порядке, сортировку и т. д.

Чаще всего в работе с Power Query встречаются списки, где все элементы одного типа, но в общем случае это совершенно не обязательно: легко можно сделать список из разношёрстных элементов, например:

```
= {1, 5, "Маша", null, true}
```

Если нужно создать список из большого количества последовательных элементов-чисел (с шагом 1), то можно не прописывать каждое число вручную, а использовать конструкцию:

```
= {1:12}
```

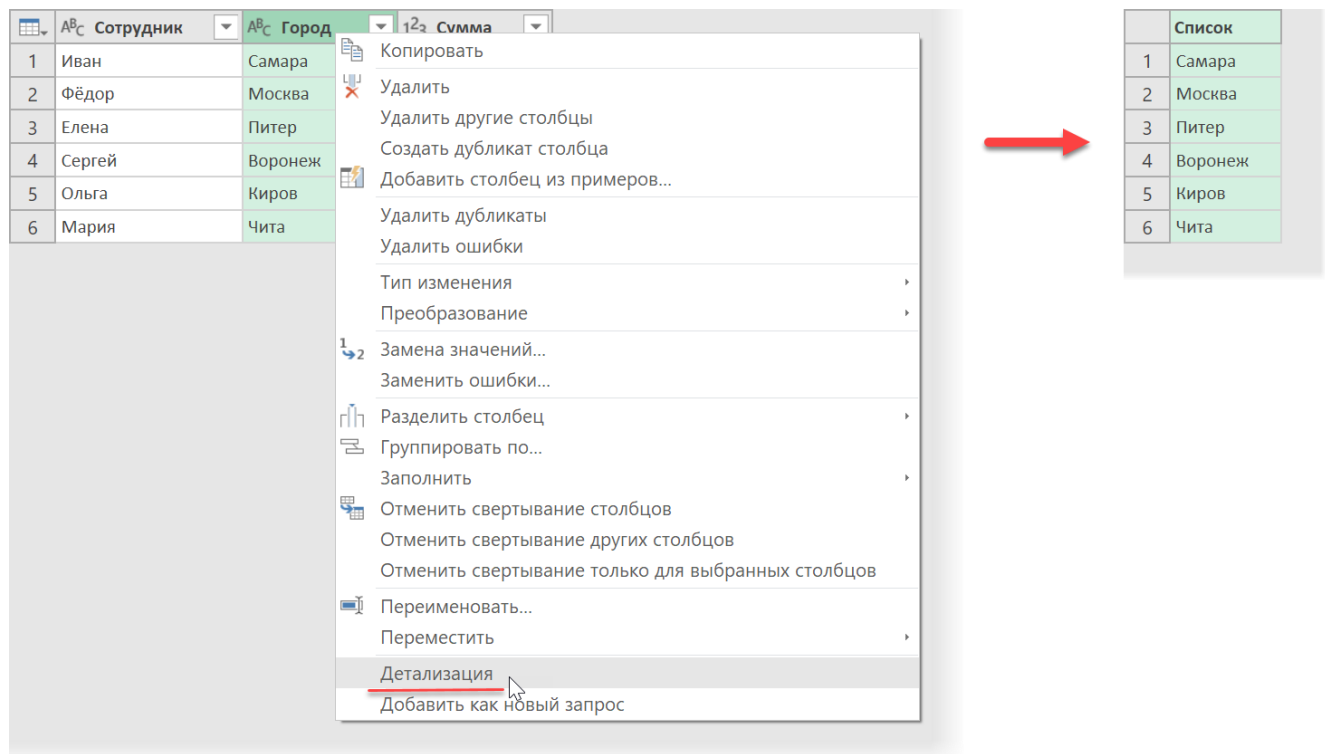
Тот же результат даст применение двух последовательных точек для задания интервала «от и до»:

```
= {1..12}
```

или использование встроенной M-функции **List.Numbers**:

```
=List.Numbers(1,12)
```

Если вы работаете в редакторе Power Query с таблицей, то можно легко извлечь из неё любой столбец в виде списка. Для этого нужно щёлкнуть правой кнопкой мыши по заголовку требуемого столбца и выбрать команду **Детализация (Drill Down)**:



Извлечение элементов списка

Если вам нужно обратиться к отдельному элементу списка, то можно использовать его индекс (порядковый номер, считая от нуля), добавленный после списка в фигурных скобках:

```
={"Москва", "Самара", "Воронеж", "Новосибирск"}{1}
```

... выдаст в качестве результата *Самару*.

Также можно извлечь первый и последний элементы любого списка с помощью встроенных M-функций **List.First** и **List.Last**, например:

```
=List.Last({"Москва", "Самара", "Воронеж", "Новосибирск"})
```

выдаст нам последний *Новосибирск*.

Сравнение списков

Списки можно сравнивать между собой с помощью стандартных операторов = и <>, например:

```
={3,5,7}={3,5,7}
```

выдаст нам *true*.

Обратите внимание, что для равенства двух списков в них должны быть и одинаковые элементы, и одинаковая их последовательность, например:

```
={3,5,7}={3,7,5}
```

Это уже не одинаковые списки, и мы получим в результате сравнения *false*.

Соединение списков

Списки можно соединять (сцеплять) друг с другом при помощи стандартного оператора текстовой конкатенации (&). Например, вот такая команда:

```
={"красный", "оранжевый"} & {"желтый", "зеленый"}
```

... выдаст в качестве результата объединенный список:

```
={"красный", "оранжевый", "желтый", "зеленый"}
```

M-функции для работы со списками

Язык M содержит большое количество функций для обработки и трансформации списков, причем большинство из них недоступны через пользовательский интерфейс и могут использоваться только при прямом редактировании M-кода. Полный список всех функций с описаниями всегда можно найти в документации по Power Query на сайте Microsoft по ссылке <https://docs.microsoft.com/en-us/powerquery-m/list-functions>.

Вот некоторые самые полезные из них:

- **List.Count** – подсчитывает количество элементов в списке.
- **List.IsEmpty** – проверяет, является ли список пустым, и выдаёт true или false соответственно.
- **List.Select** – используется для извлечения из списка элементов, удовлетворяющих заданному условию.
- **List.FirstN**, **List.LastN** – возвращает N первых/последних элементов списка.
- **List.Combine** – является аналогом оператора конкатенации (&) и склеивает несколько списков в один.
- **List.Transform** – производит над каждым элементом списка заданное действие, т. е. применяет к нему указанную функцию.
- **List.Contains** – проверяет, входит ли указанный элемент в список, и выдает true или false соответственно.
- **List.Sort** – выдает исходный список в отсортированном виде.
- **List.Difference** – сравнивает два списка между собой и возвращает элементы первого списка, которых нет во втором списке.
- **List.Min**, **List.Max** – извлекает наименьшее/наибольшее значение из списка.
- **List.Sum**, **List.Average** – вычисляет сумму / среднее арифметическое всех чисел-элементов списка.
- **List.Numers**, **List.Dates** – генерирует список из последовательных чисел или дат.

Запись (record)

Запись – это тип данных, представляющий из себя набор из одной или нескольких пар «ключ-значение». Проще всего объяснить суть этого типа, если попросить вас представить таблицу, состоящую из одной строки. У каждого элемента в такой таблице будет имя столбца в шапке (ключ) и значение (содержимое ячейки).

Создание записей

Для создания записей в языке M используется конструкция с квадратными скобками, в которых через запятую перечисляются те самые пары «ключ=значение», например:

The screenshot shows the Power Query Editor interface. At the top, there is a ribbon with tabs: File, Главная, Преобразование, Добавление столбца, Просмотр, Средства для записей, and Запрос1 - Редактор Power Query. Below the ribbon, there is a sidebar with a 'Преобразовать' button. The main area displays a formula bar with the following record creation formula: `= [Артикул=100500, Наименование="Джинсы", Цена=5000, Дата продажи=#date(2019,01,27)]`. Below the formula bar, there is a table with the following data:

Артикул	100500
Наименование	Джинсы
Цена	5000
Дата продажи	27.01.2019

В повседневной работе с данными в Power Query записи встречаются в основном в двух ситуациях: при разборе вложенных структурированных данных (XML- или JSON-файлов) или при выполнении операций с отдельными

строками в таблицах. Как и в случае со списками (Lists), любую строку из таблицы можно извлечь в виде записи, если добавить после названия таблицы (шага) номер нужной нам строки (считая с нуля!) в фигурных скобках. Так, например, если у нас есть запрос **Продажи**, возвращающий вот такую таблицу в качестве результата:

	Сотрудник	Город	Сумма
1	Иван	Самара	81560
2	Фёдор	Москва	42190
3	Елена	Питер	25770
4	Сергей	Воронеж	22370
5	Ольга	Киров	59320
6	Мария	Чита	61980

... то команда:

```
=Продажи{3}
```

... даст нам четвёртую строку из этой таблицы в виде записи:

Сотрудник	Сергей
Город	Воронеж
Сумма	22370

Если вы не знаете номер строки, то можно извлечь запись-строку из таблицы по значению (уникальному!) какого-либо поля. Например:

```
=Товары{[Сотрудник="Сергей"]}
```

... даст нам всё ту же запись «Сергей-Воронеж-22370», что и в предыдущем случае.

Однако *Сергей* в этом случае обязательно должен быть уникальным значением в столбце *Сотрудник*, иначе мы получим сообщение об ошибке. В этом случае лучше использовать функцию **Table.SelectRows**, чтобы извлечь все строки, удовлетворяющие заданному условию:

```
=Table.SelectRows(Продажи, each ([Сотрудник] = "Сергей"))
```

Извлечение элементов записи

Если нам нужно обратиться в коде к конкретному значению какого-то поля в заданной записи, то к нашей команде нужно будет добавить имя поля (столбца) в квадратных скобках, т. е.:

```
=Продажи{3}[Город]
```

... вернёт нам *Воронеж*.

Примерно того же эффекта можно добиться, если щёлкнуть по нужной ячейке правой кнопкой мыши и выбрать команду **Детализация** (Drill Down):

The image shows a table with columns 'Сотрудник', 'Город', and 'Сумма'. The row for 'Сергей' (row 4) is highlighted. A right-click context menu is open over the 'Воронеж' cell, with 'Детализация' selected. An arrow points to a formula bar showing the result of the drill-down operation: '= Источник{3}[Город]' and the value 'Воронеж'.

(**Источник** (Source) в данном случае – это имя шага, с которого мы берём исходную таблицу в запросе).

Изменение элементов записи

Для изменения значений внутри записи используется уже знакомый нам оператор конкатенации & (амперсанд). Например, если мы хотим изменить значение цены в существующей записи:

```
= [Артикул=100500, Наименование="Джинсы", Цена=5000, Дата продажи=#date(2019,01,27)]
```

то нам потребуется формула вида:

= [Артикул=100500, Наименование="Джинсы", Цена=5000, Дата продажи=#date(2019,01,27)] & [Цена=7000]	
Артикул	100500
Наименование	Джинсы
Цена	7000
Дата продажи	27.01.2019

Если же подобным образом приклеить несуществующую в записи пару «ключ=значение», то она добавится к исходным данным как новое поле:

= [Артикул=100500, Наименование="Джинсы", Цена=5000, Дата продажи=#date(2019,01,27)] & [Менеджер="Иван"]	
Артикул	100500
Наименование	Джинсы
Цена	5000
Дата продажи	27.01.2019
Менеджер	Иван

Сравнение записей

При необходимости записи можно сравнивать между собой с помощью операторов = и <>. Причем в отличие от списков последовательность элементов записи, т. е. пар «ключ=значение», не играет роли, поэтому формула:

```
= [Товар="Пончики", Цена=100] = [Цена=100, Товар="Пончики"]
```

... выдаст *true*.

M-функции для обработки записей

К сожалению, интерфейс Power Query практически не содержит команд для управления записями (одна-единственная кнопка для конвертации записи в таблицу не в счёт). Зато напрямую через программирование на языке M мы можем манипулировать ими очень гибко с помощью встроенных M-функций. Среди них стоит упомянуть:

- **Record.FieldCount** – выдаёт количество полей в записи;
- **Record.HasFields** – проверяет, пуста ли запись или содержит данные, и выдаёт логическое *false* или *true*;
- **Record.AddField** – добавляет к существующей записи новую пару «ключ=значение», т. е. действует как &;
- **Record.RemoveField** – как легко догадаться, это обратная по смыслу команда, удаляющая ненужное поле из записи;
- **Record.FromList** – создает запись на основе заданного списка элементов и т. д.

Полный список M-функций для работы с записями всегда можно найти тут <https://docs.microsoft.com/en-us/powerquery-m/record-functions>.

Таблица (table)

Табличный тип данных – это то, с чем мы сталкиваемся в работе с Power Query чаще всего. Обычно мы получаем таблицы в качестве результатов запроса к внешним файлам, базам данных или большинству других

источников. Однако при желании можно создавать небольшие таблицы на лету прямо в редакторе М-кода с помощью команды **#table**:

The screenshot shows the M-code editor with the formula: `= #table({"Товар", "Цена"}, {"Хлеб", 50}, {"Соль", 30}, {"Спички", 10})`. Below the formula, a table is displayed with the following data:

	Товар	Цена
1	Хлеб	50
2	Соль	30
3	Спички	10

Первый аргумент здесь – это список с названиями столбцов, а второй – это список из списков (обратите внимание на вложенные фигурные скобки), каждый из которых представляет собой данные каждой отдельной строки.

Естественно, пользоваться таким способом для создания таблиц, прямо скажем, не очень удобно, но в некоторых случаях или в целях демонстрации (как в справке по Power Query на сайте Microsoft, например) он вполне адекватен.

Ссылки на элементы таблицы

Что делать, если нам необходимо сослаться на отдельные элементы таблицы: заданные строки, столбцы или ячейки? Предположим, что у нас есть запрос с именем **Продажи** или одноименная переменная (выглядит как шаг в правой панели), возвращающая в качестве результата вот такую таблицу:

	Сотрудник	Город	Сумма
1	Иван	Самара	81560
2	Фёдор	Москва	42190
3	Елена	Питер	25770
4	Сергей	Воронеж	22370
5	Ольга	Киров	59320
6	Мария	Чита	61980

Как мы уже упоминали ранее, чтобы извлечь из этой таблицы строку, мы должны использовать выражение:

```
=Продажи{3}
```

Это даст нам четвёртую (нумерация строк начинается от нуля!) строку из этой таблицы в виде записи:

The screenshot shows the M-code editor with the formula: `= Продажи{3}`. Below the formula, a table is displayed with the following data:

Сотрудник	Сергей
Город	Воронеж
Сумма	22370

Если нам нужно сослаться на столбец, то его имя добавляется к названию таблицы в квадратных скобках. Например:

```
=Продажи[Город]
```

... даст нам содержимое столбца **Город** в виде списка:

	Список
1	Самара
2	Москва
3	Питер
4	Воронеж
5	Киров
6	Чита

Сочетая эти два подхода, можно легко адресоваться и к любой ячейке в таблице. Например, если нам нужен третий город, то это реализуется в М-коде как:

```
=Продажи[Город]{2}
```

... или:

```
=Продажи{2}[Город]
```

Оба эти выражения дадут один результат (*Питер*), но первое сначала берёт из таблицы **Продажи** столбец **Город**, а потом извлекает из него третье значение, а второе выражение сначала берёт из таблицы третью строку (запись), а потом выдает из неё содержимое поля **Город**.

М-функции для обработки таблиц

В языке М есть почти сотня функций для выполнения различных операций с таблицами. Полный список всегда можно найти тут <https://docs.microsoft.com/en-us/powerquery-m/table-functions>.

Вот наиболее полезные из них:

- **Table.RowCount**, **Table.ColumnCount** – возвращают количество строк и столбцов в указанной таблице, т. е. могут использоваться для определения её размеров;
- **Table.IsEmpty** – проверяет, пуста ли таблица, и возвращает *true* или *false*;
- **Table.FromColumns**, **Table.FromRows**, **Table.FromList**, **Table.FromRecords** – создают таблицу из заданных исходных фрагментов (столбцов, строк, списков и т. д.);
- **Table.First**, **Table.Last**, **Table.FirstN**, **Table.LastN** – выдает первые или последние строки из таблицы;
- **Table.Range** – извлекает заданное количество строк из середины таблицы (т. е. начиная с определенного номера строки);
- **Table.SelectRows** – извлекает из исходной таблицы строки, удовлетворяющие заданному условию, и формирует из них новую таблицу. Обычно, когда мы фильтруем в Power Query наши данные, то именно эта функция и выполняет всю работу;
- **Table.InsertRows**, **Table.RemoveRows** – вставляет или удаляет строки из исходной таблицы;
- **Table.Combine** – склеивает несколько таблиц в одну друг под другом (см. главу [Добавление \(Append\)](#));
- **Table.HasColumns** – проверяет, есть ли в таблице столбец/столбцы с заданным именем, и возвращает *true* или *false*. Очень удобно её использовать для отбора только нужных таблиц при массовой загрузке всех таблиц из файла или папки;
- **Table.TransformColumnNames** – применяет к названиям столбцов в таблице заданную функцию (например, исправляет их регистр или убирает лишние пробелы);
- **Table.Unpivot** и **Table.UnpivotOtherColumns** – разворачивает кросс-таблицу в плоскую, см. главу [Отмена свёртывания](#);
- **Table.Distinct** – удаляет из таблицы все дубликаты;
- и т. д.

Большинство из этих функций доступны через интерфейс пользователя и имеют аналоги в виде кнопок на ленте на вкладках **Главная (Home)** и **Преобразование (Transform)**, но некоторые функции можно использовать только в М-коде.

Справка по встроенным функциям

В предыдущих главах мы упомянули уже много различных функций для манипуляций с текстом, числами, датами, списками, таблицами и т. д. К сожалению, Power Query пока что не выдает подсказок при вводе функций и их аргументов (как Excel). Предполагаю, что, скорее всего, это будет реализовано в ближайшем будущем (и частично уже появилось в Microsoft Power BI), но на данный момент такой удобной user-friendly функциональности у нас нет.

Что же делать, если вы забыли точное название функции или её аргументы?

Можно, конечно, обратиться к сайту Microsoft, который мы уже упоминали и где содержится подробная справка по всем функциям языка M: <https://docs.microsoft.com/en-us/powerquery-m/power-query-m-function-reference>. Если же у вас в данный момент нет доступа к интернету, то поможет следующий трюк.

1. Создадим новый пустой запрос через **Данные → Получить данные → Из других источников → Пустой запрос** (Data → Get Data → From Other Sources → Blank Query).
2. В открывшемся окне дадим нашему запросу любое имя (например, *Справка*).
3. В строке формул введём команду `=#shared` и нажмём на клавишу **Enter**.

На экране появится список всех встроенных функций языка M. Щелчок мышью в белый фон ячейки рядом со словом **Function** отобразит в нижней части окна подробную справку по интересующей вас функции:

The screenshot shows the Power Query interface. In the 'Queries' pane on the left, a query named 'Справка' is selected. The formula bar at the top contains '= #shared'. A list of text functions is displayed, with 'Text.Proper' highlighted. Below the list, the detailed documentation for 'Text.Proper' is shown, including its syntax, description, and an example.

Function Name	Type
Text.Clean	Function
Text.PositionOf	Function
Text.PositionOfAny	Function
Text.Lower	Function
Text.Upper	Function
Text.Proper	Function
Text.Split	Function
Text.SplitAny	Function
Text.Combine	Function
Text.Repeat	Function
Text.Replace	Function
Text.ReplaceRange	Function
Text.Insert	Function
Text.Remove	Function
Text.RemoveRange	Function
Text.Reverse	Function
Text.Select	Function

function (*text* as nullable text, *optional culture* as nullable text) as nullable text

Возвращает результат перевода в верхний регистр только первой буквы каждого слова в текстовом значении *text*. Все остальные буквы возвращаются в нижнем регистре.

Пример: Использовать `Text.Proper` для простого предложения.

Использование:

```
Text.Proper("the QUICK BrOwN fOx jUmPs oVER tHe LAzy DoG")
```

Выходные данные:

```
"The Quick Brown Fox Jumps Over The Lazy Dog"
```

Если щёлкнуть мышью в само слово **Function**, то мы «провалимся» в подробности (появится дополнительный шаг **Навигация** в правой панели) и сможем протестировать функцию на любых входных данных.

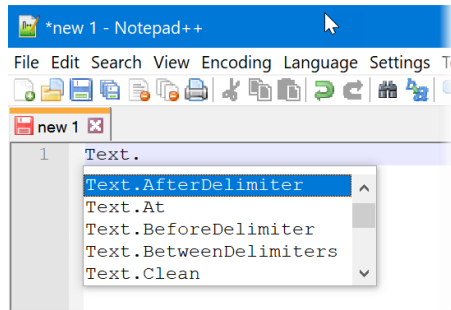
Чтобы удобнее было искать нужные функции, можно преобразовать полученный список в таблицу с помощью кнопки **В таблицу** (To table) на вкладке **Преобразовать** (Transform). После этого в шапке появятся привычные фильтры, которыми можно будет воспользоваться для быстрого поиска требуемых функций.

Созданный запрос *Справка* можно обычным образом сохранить как подключение и смело обращаться к нему в будущем, если у вас возникает потребность в получении подробностей по той или иной M-функции.

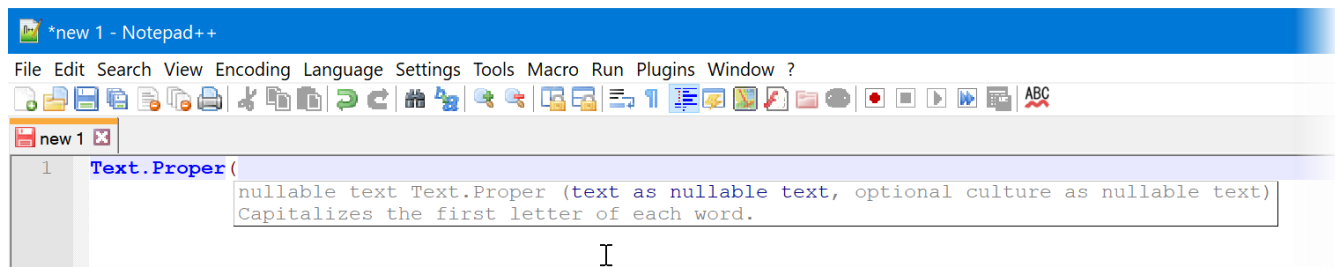
Редактор M-кода Notepad++ с подсветкой синтаксиса

Продолжая развивать тему, начатую в прошлой главе, хотелось бы упомянуть ещё один удобный инструмент для ввода и редактирования M-кода – это бесплатный **текстовый редактор Notepad++**. Это очень мощный, удобный и гибкий инструмент, и плюсы у него не только в названии, поверьте.

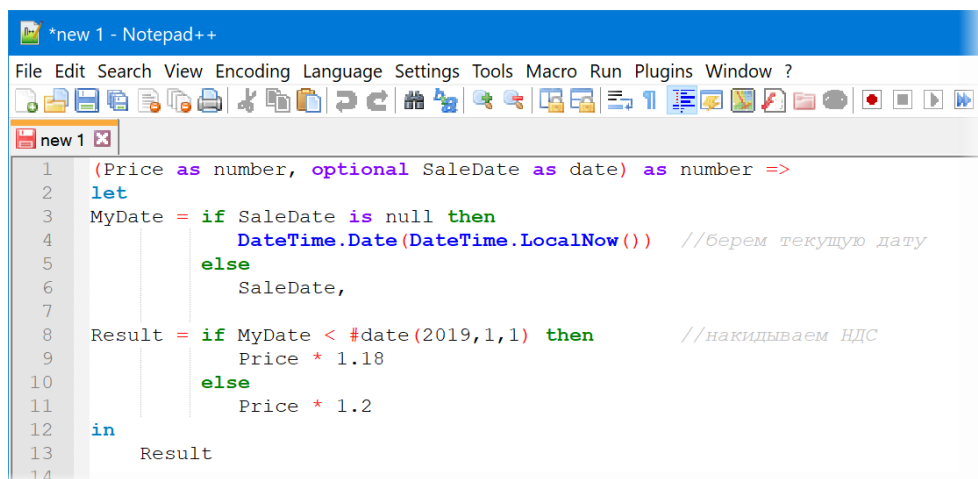
Он умеет показывать всплывающие подсказки по первым буквам для всех встроенных функций Power Query:



Отображает подсказку по аргументам любой функции и её краткое описание:



Поддерживает цветовую подсветку синтаксиса:

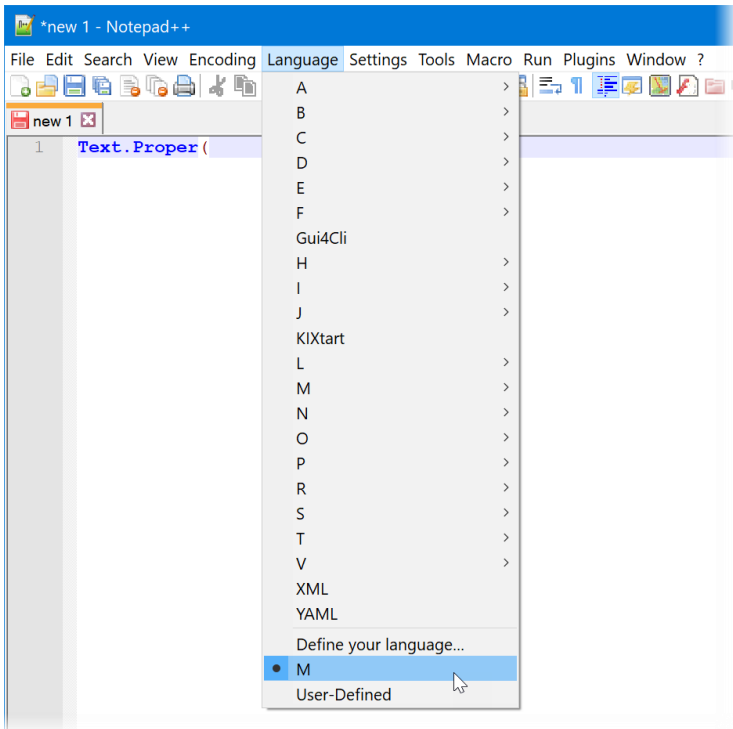


Короче говоря, в нём есть всё то, чего не хватает любому нормальному пользователю при работе с M-кодом во встроенном предельно спартанском **Расширенном редакторе (Advanced Editor)**.

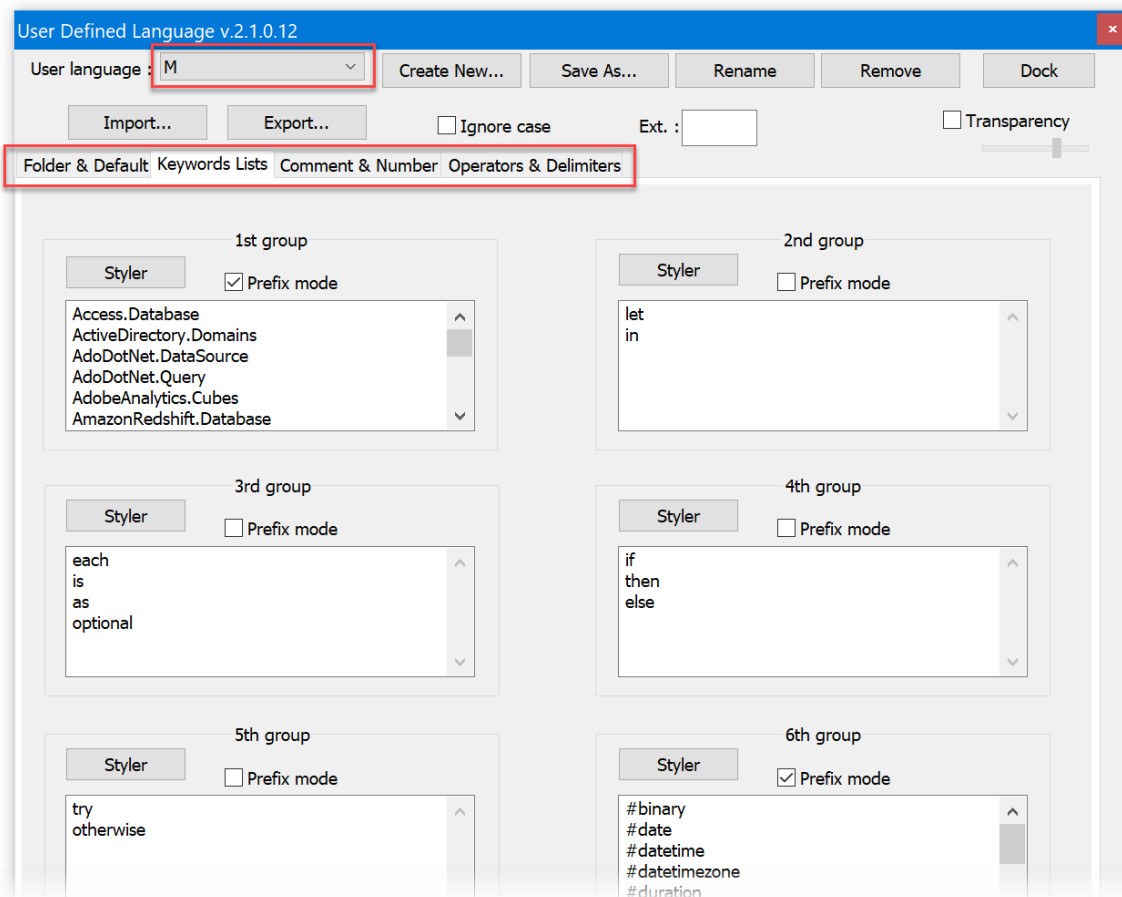
Чтобы воспользоваться всеми вышеперечисленными радостями, вам нужно:

1. Скачать и установить последнюю версию Notepad++ с сайта <https://notepad-plus-plus.org/>.
2. Скопировать файл **M.xml**, содержащий подсказки для функций языка M, из папки с примерами к этой книге в папку **C:\Program Files (x86)\Notepad++\autoCompletion**.
3. Запустить Notepad++ и выбрать в меню **Синтаксисы** → **Задать свой синтаксис** (Language → Define your language) и, нажав кнопку **Импорт** (Import), указать файл, содержащий информацию о цветовой подсветке синтаксиса (файл **M Language Notepad Plus Markup.xml** из папки с примерами к этой книге).
4. Перезапустить Notepad++.

Теперь в меню **Синтаксисы** (Language) должен появиться наш язык M, который можно выбрать, и наслаждаться затем всеми «благами цивилизации»:



При желании вы можете легко внести правки в цветовую схему: поменять цвета на свои или добавить ещё какие-то дополнительные ключевые слова для подсветки. Это можно сделать через то же самое меню **Синтаксиса** → **Задать свой синтаксис** (Language → Define your language), выбрав затем в верхней части окна из выпадающего списка наш язык M:



После этого можно «погулять» по вкладкам **Ключевые слова** (Keyword Lists), **Комментарии и числа** (Comments & Number) и **Операторы и разделители** (Operators & Delimiters) и задать свои параметры форматирования (шрифт, цвет, начертание и т. д.) для каждой группы с помощью соответствующих кнопок **Стиль** (Styler).

Пользовательские функции

*Все началось с того, что я огляделся по сторонам и, не увидев автомобиля своей мечты, решил сконструировать его сам.
(Фердинанд Порше)*

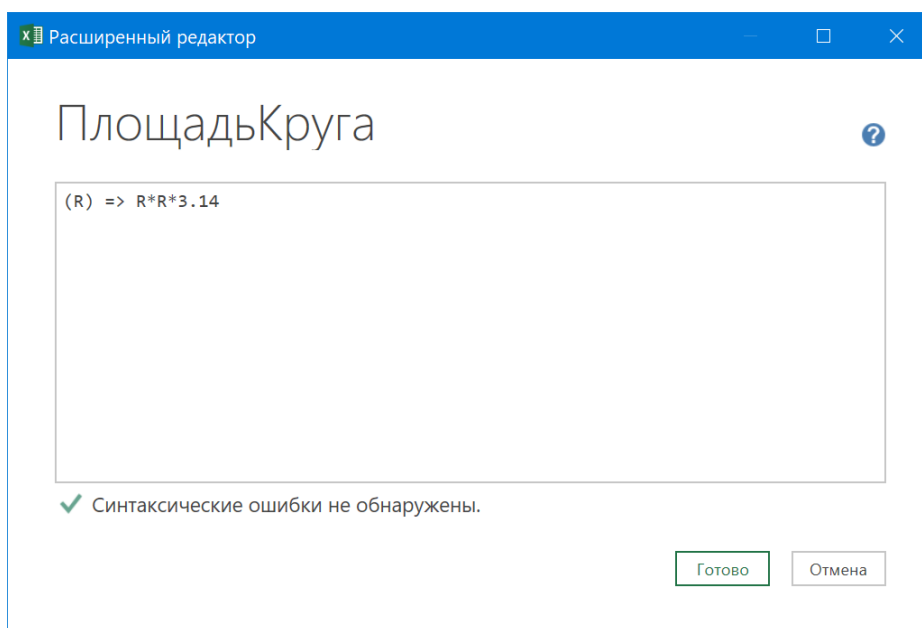
Помимо использования встроенных функций для обработки данных, язык M, как и любой другой язык программирования, позволяет нам создавать собственные функции. Это критически важный навык в сложных задачах, которые редко обходятся без такой необходимости. Если помните, то мы уже встречали подобное в главе [Удаление лишних пробелов и SuperTrim](#). Теперь давайте рассмотрим эту технику более детально.

Создание простой функции

Предположим, что нам нужно создать свою пользовательскую функцию для расчёта площади круга по заданному радиусу. Для этого:

4. Создадим новый пустой запрос через **Данные → Получить данные → Из других источников → Пустой запрос** (Data → Get Data → From Other Sources → Blank Query).
5. В открывшемся окне дадим нашему запросу подходящее имя (например, *ПлощадьКруга*).
6. Перейдём в **Расширенный редактор** на вкладке **Просмотр** (View → Advanced Editor).
7. Введём туда следующий код:

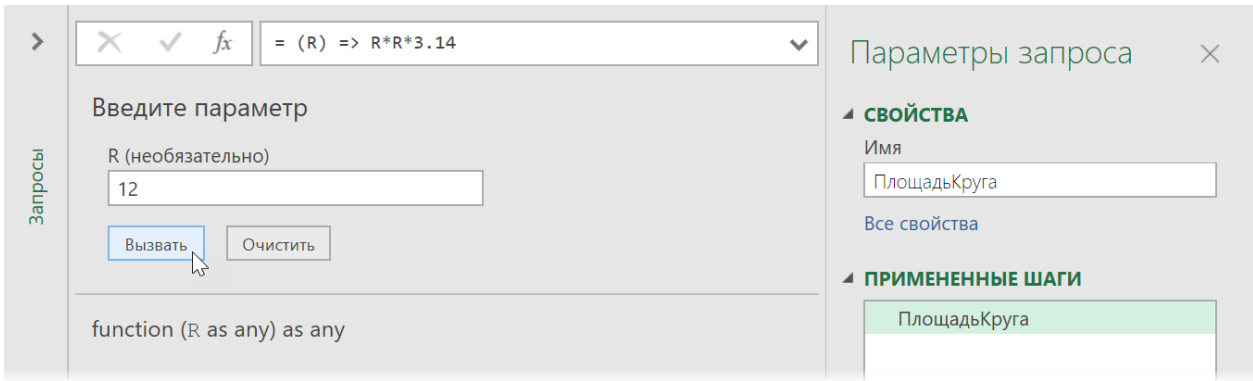
```
(R) => R*R*3.14
```



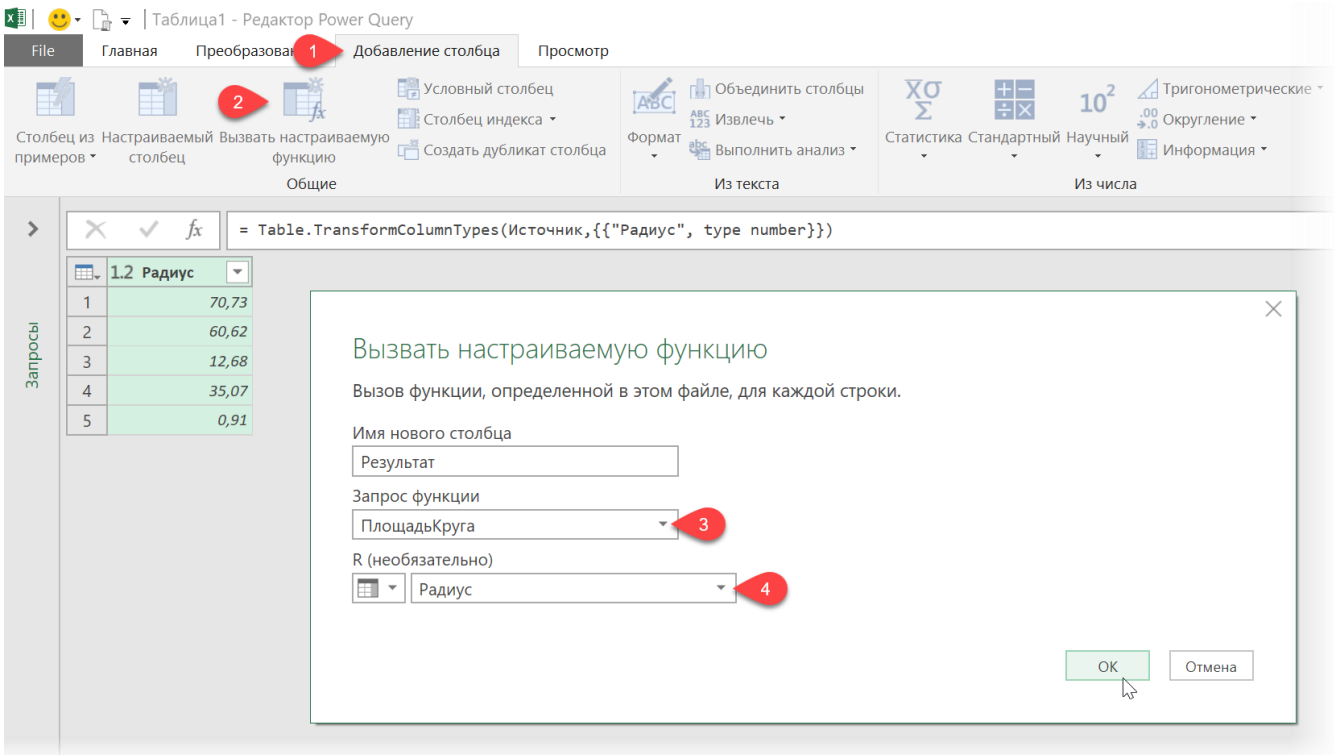
Технически это означает, что мы создаём функцию, где будет один аргумент в виде переменной R неопределённого типа, и на выходе эта функция должна возвращать результат выражения $R \cdot R \cdot 3.14$.

Вызов пользовательской функции

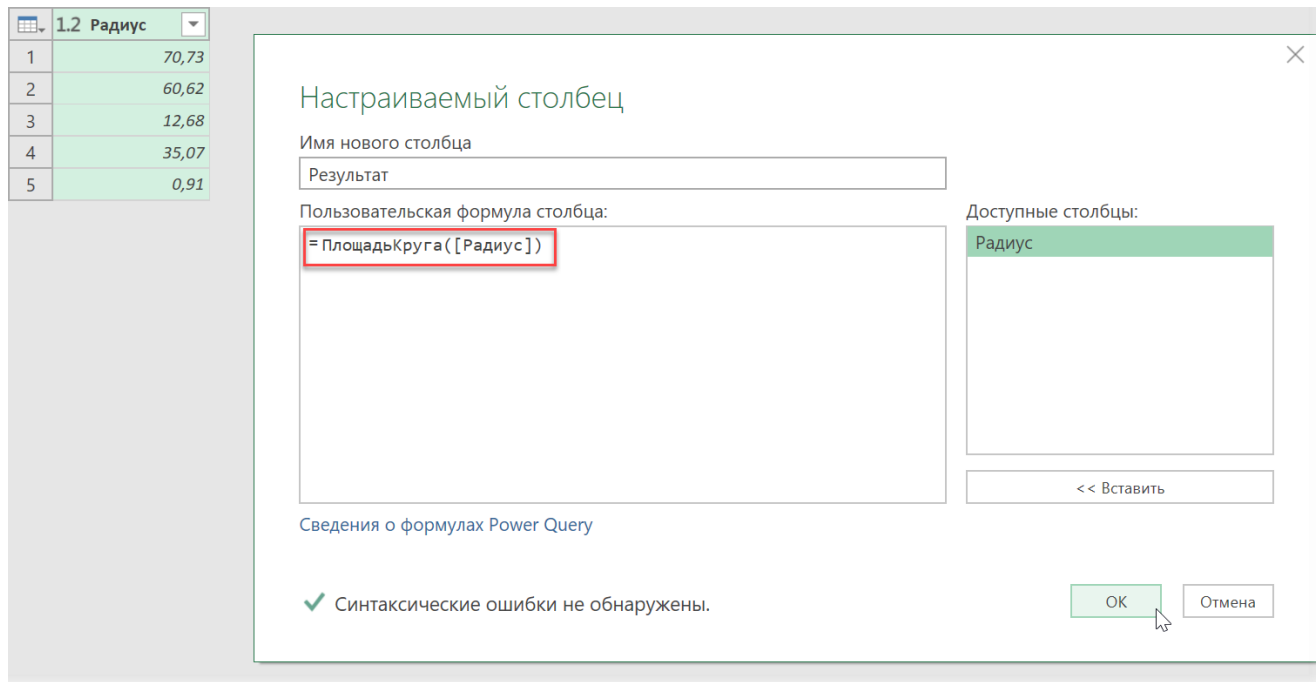
После нажатия на **Готово** в предыдущем окне функцию можно опробовать, как мы уже это делали в предыдущей главе, введя в соответствующее поле значение входного параметра (радиуса круга) и нажав на кнопку **Вызвать** (Invoke):



Созданную функцию теперь можно использовать в любых запросах в этой книге. Вызвать её можно, используя кнопку **Вызвать настраиваемую функцию** на вкладке **Добавление столбца** (Add Column → Invoke Custom Function) и выбрав затем нашу функцию и её аргументы из удобного выпадающего списка:



Также можно напрямую вписать имя функции в формулу при создании вычисляемого столбца через **Добавление столбца** → **Настраиваемый столбец** (Add Column → Custom Column):



И, наконец, можно просто сделать двойной щелчок левой кнопкой мыши по названию нашей функции в панели **Запросы и подключения** (Queries & Connections) в правой части окна Microsoft Excel.

Типы данных для аргументов и результата

При создании более-менее серьёзных функций лучше уточнять типы данных задаваемых аргументов и возвращаемого функцией значения-результата. Да и самих аргументов запросто может быть больше, чем один. Допустим, мы хотим сделать пользовательскую функцию, которая будет в качестве аргумента принимать дату и цену товара, а на выходе выдавать цену уже с НДС (автоматически определяя ставку 18% или 20% в зависимости от даты сделки). Реализовать такое можно следующим кодом:

```
(SaleDate as date, Price as number) as number =>
if SaleDate<#date(2019,1,1) then
    Price*1.18
else
    Price*1.2
```

Обратите внимание на следующее.

- Здесь объявляется два входных аргумента – переменная *SaleDate* типа даты и переменная *Price* числового типа.
- Уточняется, что наша функция на выходе возвращает числовое значение (об этом говорит **as number** в конце первой строки).

Также отметьте, что текст кода функции можно смело разносить на разные строки и делать в нём отступы для наглядности, как мы это уже делали ранее. Если в процессе вычисления результатов нужно выполнить несколько операций, то их можно заключать в стандартный блок **let...in**, например:

```
(SaleDate as date, Price as number) as number =>
let
    NDS_New = 0.2,
    NDS_Old = 0.18,
    Result = if SaleDate<#date(2019,1,1) then
        Price*(1+NDS_Old)
    else
```



```

        Price*(1+NDS_New)
in
    Result

```

Необязательные аргументы

Некоторые аргументы можно задать как необязательные, для этого перед ними ставится ключевое слово **optional**. Например, дата в приведенной выше функции расчёта НДС может быть не обязательной. Если она не задана (т. е. равна *null*), то мы подставляем вместо неё текущую с помощью функции **DateTime.LocalNow**:

```

(PPrice as number, optional SaleDate as date) as number =>
let
    MyDate = if SaleDate is null then
        DateTime.Date(DateTime.LocalNow())
    else
        SaleDate,
    Result = if MyDate < #date(2019,1,1) then
        PPrice*1.18
    else
        PPrice*1.2
in
    Result

```

Важно отметить, что необязательные аргументы должны быть перечислены последними, после обязательных. Поэтому в первой строке в скобках аргументы переставлены местами: сначала идёт *Price*, а только потом опциональная *SaleDate*.

Функция внутри запроса

На самом деле для создания функции совершенно не обязательно создавать отдельный пустой запрос, как мы это делали в первом примере этой главы. Технически возможно прописать пользовательскую функцию прямо внутри запроса Power Query, т. е. в пределах блока **let ... in**, например:

```

let
    ДлинаОкружности = (R) => 2*R*3.14,
    Size = 10,
    Result = ДлинаОкружности(Size)
in
    Result

```

Здесь мы определяем функцию **ДлинаОкружности** в первой (не считая **let**) строке запроса и затем тут же вызываем её в третьей для вычисления по заданному радиусу.

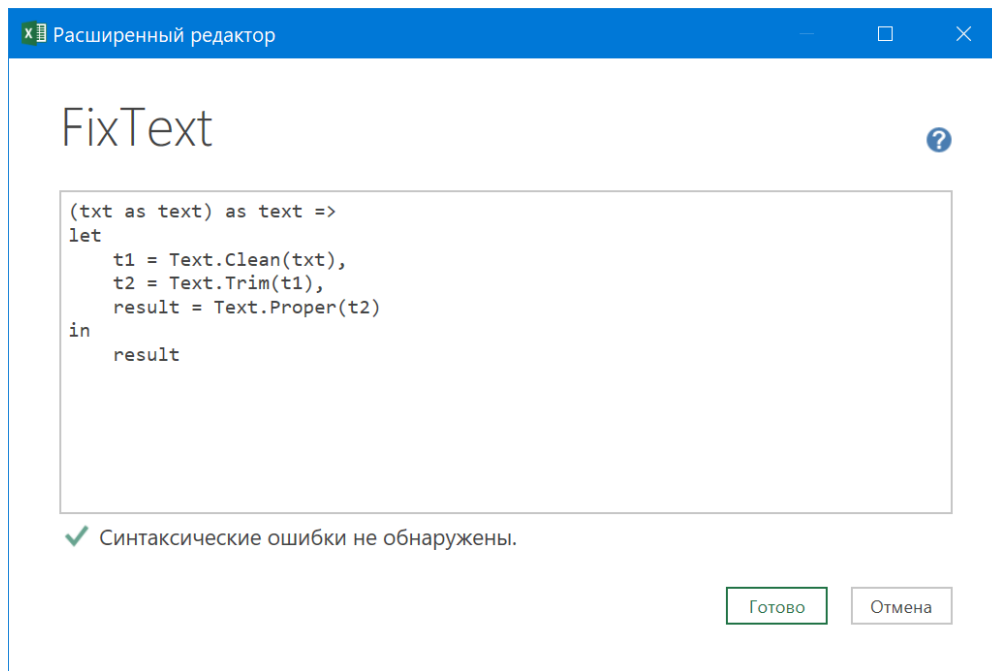
При таком способе, однако, надо помнить, что все переменные и функции в блоке **let ... in** являются локальными, т. е. воспользоваться такой вложенной функцией извне (из других запросов в этом файле) уже не получится.

Функция как аргумент для другой функции

Power Query вполне спокойно позволяет передавать название одной функции как аргумент для другой. Чтобы проще было объяснить этот немного странный на первый взгляд момент, представьте себе следующую ситуацию.

Предположим, что у вас есть загруженная в Power Query из какого-либо внешнего источника обновляемая таблица, где постоянно наблюдаются проблемы с названиями столбцов: в них «гуляет» регистр (прописные и строчные) и периодически встречаются лишние пробелы и непечатаемые символы.

Чтобы исправить текст заголовков, создадим несложную пользовательскую функцию (я назвал её **FixText**):



Как легко сообразить, здесь сначала из текста удаляются непечатаемые символы функцией **Text.Clean**, затем убираются лишние пробелы с помощью **Text.Trim**, и, наконец, в получившемся тексте исправляется регистр – первая буква каждого слова делается заглавной функцией **Text.Proper**.

Теперь предположим, что мы загрузили в Power Query таблицу с данными и хотим применить созданную функцию к названию каждого столбца:

	ABC 123 город	ABC 123 ТОВАР	ABC 123 мЕнЕджеР	ABC 123 дата ПОСТАВКИ
1	Москва	Хлеб	Иван	24.04.2015 0:00:00
2	Самара	Спички	Сергей	23.05.2012 0:00:00
3	Киров	Сало	Марина	21.11.2012 0:00:00

Это легко можно сделать, дописав к исходному коду запроса:

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Таблица2"]}[Content]
in
    Источник
```

ещё один шаг (переменную **Красота**) со встроенной функцией **Table.TransformColumnNames**, которая как раз и получает имя нашей пользовательской функции **FixText** как аргумент и применяет затем её к заголовку каждого столбца:

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Таблица2"]}[Content],
    Красота = Table.TransformColumnNames(Источник, FixText)
in
    Красота
```

Обратите внимание, что имя функции пишется без кавычек (не как текст). На выходе получаем исправленные заголовки столбцов. Совсем другое дело:

	ABC 123	Город	ABC 123	Товар	ABC 123	Менеджер	ABC 123	Дата Поставки
1		Москва		Хлеб		Иван		24.04.2015 0:00:00
2		Самара		Спички		Сергей		23.05.2012 0:00:00
3		Киров		Сало		Марина		21.11.2012 0:00:00

Рекурсия

В отдельных случаях бывает необходимо использовать рекурсию – вызов функции из неё же самой. Это позволяет реализовать цепочку трансформаций, где на каждом шаге данные берутся с предыдущей итерации и используются для расчёта текущих значений. Простейший пример рекурсии в повседневной жизни – матрёшка. Классические примеры рекурсии в программировании:

- Расчёт **чисел Фибоначчи**, т. е. последовательности чисел, которая начинается с 0 и 1, а потом каждое последующее число является суммой двух предыдущих, т. е. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144 и т. д. Числа Фибоначчи играют важную роль в природе, биологии, технике, дизайне, астрономии, биржевом техническом анализе и т. д.
- Алгоритм **обхода дерева с заранее неизвестным количеством ветвей**. Классический пример – макрос обхода всех вложенных папок и вывода списка всех файлов, которые в них размещены¹.
- Вычисление **факториала**, т. е. результата перемножения всей последовательности чисел от 1 до N. Например, факториал пяти (обозначается 5!) = 1*2*3*4*5 = 120. Факториал 0 принимается равным 1.

Давайте рассмотрим, как рекурсия реализуется на языке M на примере факториала.

Создадим новый пустой запрос и введём туда следующий код:

```
let
    Factorial =
        (n)=>
            if (n = 0) then
                1
            else
                n * @Factorial(n-1)
in
    Factorial
```

Логика тут следующая.

- Если в качестве аргумента n будет число 0, то функция вернёт 1.
- В противном случае функция должна умножить n на значение факториала для (n-1), т. е. на произведение всей предыдущей последовательности чисел. Но, чтобы её вычислить, нужно опять вызвать эту же функцию.
- Для рекурсивного вызова функции внутри неё же перед её именем добавляется @.

В заключение хотелось бы добавить, что сильно увлекаться подобными техниками не стоит, особенно на больших объемах данных. Во-первых, время обновления подобных запросов в разы дольше. Во-вторых, любая рекурсия требует от Power Query вычислять и хранить все предыдущие состояния, что с большим аппетитом скушает ресурсы вашего процессора и свободную память.

¹ См. статью и макрос на <https://www.planetaexcel.ru/techniques/12/45/>.

Ключевое слово `each`

Ключевое слово **`each`** весьма часто встречается в исходных кодах запросов языка M (особенно при фильтрации) и предназначено для быстрого и элегантного создания и вызова небольших функций на лету.

Чтобы понять логику работы **`each`**, рассмотрим вот такую несложную функцию, которая накидывает на исходную цену 20% НДС:

```
(Price) => Price*1.2
```

Поскольку, формально говоря, имя функции может быть любым, то это же самое выражение можно переписать, заменив имя аргумента *Price* на нижнее подчеркивание:

```
(_) => *_1.2
```

И тут на сцену выходит наше слово **`each`**, которое успешно заменяет первую часть этого выражения, делая его ещё компактнее (на языке программистов подобное упрощение забавно называется *синтаксическим сахаром*):

```
each *_1.2
```

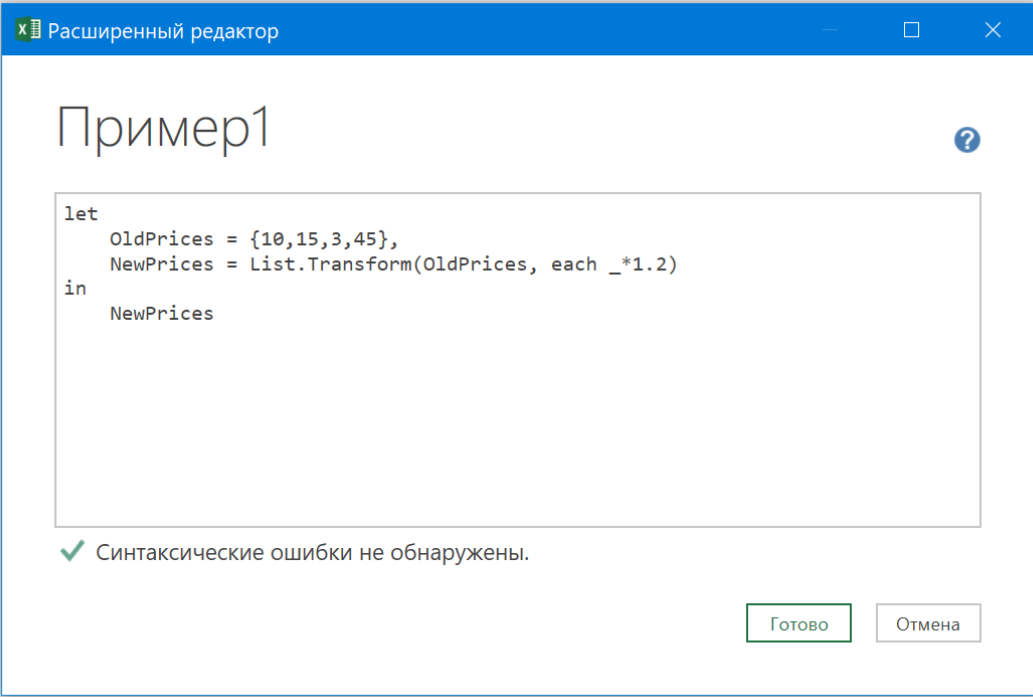
Обычно подобные выражения используют при групповой обработке сложных типов данных: переборе элементов в списках, строк в таблицах и т. д.

Давайте рассмотрим несколько примеров для понятности.

Пример 1. Арифметические операции с элементами списка

Предположим, что у нас есть список с ценами, и мы хотим накинуть на каждую сумму 20% налога. Код на языке M будет выглядеть как:

	Список
1	12
2	18
3	3,6
4	54



The screenshot shows a window titled "Расширенный редактор" (Advanced Editor) with the following M code:

```
let
    OldPrices = {10,15,3,45},
    NewPrices = List.Transform(OldPrices, each *_1.2)
in
    NewPrices
```

Below the code, a green checkmark indicates: "Синтаксические ошибки не обнаружены." (No syntax errors detected). At the bottom right, there are buttons for "Готово" (Done) and "Отмена" (Cancel).

В данном случае используется функция **`List.Transform`**, первым аргументом которой является исходный список, а вторым – наша функция, которая применяется к каждому его элементу.

Пример 2. Обработка текстового списка

Допустим, нам нужно применить к каждому элементу текстового списка функцию преобразования регистра **`Text.Proper`**, чтобы сделать первую букву каждого слова прописной. На языке M это выглядело бы как:

	Список
1	Саша
2	Маша
3	Даша

Расширенный редактор

Пример2

```

let
    MyList = {"саша", "МАША", "даша"},
    Result = List.Transform(MyList, each Text.Proper(_))
in
    Result

```

Пример 3. Фильтрация строк в таблице

Предположим, что мы загрузили в Power Query таблицу с данными по продажам на шаге **Источник**:

	Товар	Менеджер	Количество	Дата заказа	Дата отгрузки
1	Яблоко	Сергей	65	17.07.2011 0:00:00	19.07.2011 0:00:00
2	Картофель	Анастасия	87	03.01.2012 0:00:00	08.01.2012 0:00:00
3	Яблоко	Анастасия	64	20.02.2015 0:00:00	02.03.2015 0:00:00
4	Картофель	Мария	73	28.08.2011 0:00:00	28.08.2011 0:00:00
5	Дыня	Мария	27	06.11.2017 0:00:00	08.11.2017 0:00:00
6	Яблоко	Анна	39	15.07.2011 0:00:00	19.07.2011 0:00:00
7	Брокколи	Мария	20	01.03.2011 0:00:00	10.03.2011 0:00:00

Если отфильтровать, например, все сделки менеджера **Анны**, то в строке формул мы увидим, как уже знакомое нам слово **each** применяется для проверки всех имен из столбца **Менеджер** внутри функции **Table.SelectRows**:

	Товар	Менеджер	Количество	Дата заказа	Дата отгрузки
1	Яблоко	Анна	39	15.07.2011 0:00:00	19.07.2011 0:00:00
2	Апельсин	Анна	98	30.07.2014 0:00:00	31.07.2014 0:00:00
3	Земляника	Анна	28	12.03.2014 0:00:00	22.03.2014 0:00:00
4	Дыня	Анна	75	15.09.2017 0:00:00	16.09.2017 0:00:00
5	Апельсин	Анна	34	29.05.2010 0:00:00	04.06.2010 0:00:00
6	Просо	Анна	33	14.10.2012 0:00:00	18.10.2012 0:00:00
7	Лосось	Анна	80	13.02.2011 0:00:00	14.02.2011 0:00:00
8	Земляника	Анна	61	28.03.2010 0:00:00	30.03.2010 0:00:00
9	Дыня	Анна	39	08.04.2012 0:00:00	12.04.2012 0:00:00

Обратите внимание, что в этом случае уже не используется символ нижнего подчёркивания, т.к. идёт обращение к полю (столбцу) в таблице или записи.

По сути, приведенная выше формула:

```
= Table.SelectRows(Источник, each ([Менеджер] = "Анна"))
```

была бы равносильна созданию отдельной пользовательской функции (например, с именем **AnnaOrNot**), которая получала бы в качестве входящего аргумента строку из таблицы (т. е. запись – переменная **my_record**), извлекала бы из нее содержимое поля **Менеджер** и проверяла бы затем, **Анна** это или нет, выдавая на выходе логическую истину (*true*) или ложь (*false*). А потом имя этой функции можно было бы использовать в качестве второго аргумента функции фильтрации **Table.SelectRows**:

```
let
```

```
    Источник = Excel.CurrentWorkbook(){[Name="Продажи"]}[Content],
```

```

AnnaOrNot = (my_record)=>
    let
        value = my_record[Менеджер],
        result = (value = "Анна")
    in
        result,

Продажи_Анны = Table.SelectRows(Источник, AnnaOrNot)
in
    Продажи_Анны

```

Первый вариант с **each** гораздо компактнее, поэтому он используется чаще всего, и именно его будет записывать в M-коде редактор Power Query, когда мы фильтруем таблицу вручную, используя кнопки в шапке таблицы.

Пример 4. Сложные фильтры

Понимание принципов работы **each** позволяет компактно реализовать любые нестандартные сценарии при фильтрации, которые невозможно сделать обычным образом через интерфейс. Например, чтобы отфильтровать все строки, где вторая буква в имени менеджера «а», нужно будет лишь указать:

```
= Table.SelectRows(Источник, each (Text.Range([Менеджер],1,1)="a"))
```

Здесь функция **Text.Range** извлекает второй символ (нумерация начинается с нуля) имени и проверяет, равен ли он «а».

Или представьте себе ситуацию, когда вам нужно отфильтровать только те товары, которые написаны начиная с маленькой (строчной) буквы. Поможет конструкция:

```
= Table.SelectRows(Источник, each (Text.Start([Товар],1) = Text.Lower(Text.Start([Товар],1))))
```

Здесь функция **Text.Start** извлекает из названия товара первую букву, а затем проверяется, равна ли она той же букве, но преобразованной в строчные функцией **Text.Lower**.

Подобным образом также удобно выполнять фильтрацию, когда нужно как-то сравнивать между собой значения в одной строке. Например, если нужно вывести все продажи, где дата отгрузки не отличается от даты заказа, то это будет:

```
=Table.SelectRows(Источник, each ([Дата заказа] = [Дата отгрузки]))
```

И конечно же, можно комбинировать несколько условий при проверке, соединяя их соответствующим образом операторами **or**, **and** и **not**. Например, если нам нужно найти все строки, где менеджером была Анна и шла продажа апельсинов, то это будет выглядеть как:

```
=Table.SelectRows(Источник, each ([Менеджер] = "Анна" and [Товар]="Апельсин"))
```

Обработка ошибок в запросах

Если вы не учитесь на своих ошибках, нет смысла их делать.

(Лоренс Питер)

Рассмотрим простой пример.

Предположим, что у нас есть внешний файл, куда один или несколько других пользователей вносят данные по сделкам: наименования, цены и количества проданных товаров. Представим, что мы загрузили в Power Query данные из этого файла. Допустим также, что нам нужно подсчитать стоимость по каждой строке-сделке, что, как вы понимаете, легко реализуется добавлением ещё одного столбца через **Добавление столбца** → **Настраиваемый столбец** (Add Column → Custom Column) с формулой:

№	Товар	Цена	Количество	Стоимость
1	Яблоки	120	3	360
2	Апельсины	220	4	880
3	Сливы	80	2	160
4	Груши	150	3	450

Каждый день данные в исходном файле добавляются-изменяются, вы обновляете запрос, таблица пересчитывается. Всё просто, понятно и хорошо работает. Можно расслабиться.

Или нет?

А что будет, если пользователь, вводящий данные в файл, откуда мы загружаем информацию, допустит ошибку или «проявит инициативу»? Переименует один из столбцов в исходных данных – «Количество» в «Кол-во», например? Или использует не тот символ-разделитель в дробных числах? Или напишет «два» вместо «2», переименует исходный файл или ещё что похуже? Тут возможны два варианта развития событий.

Если возникшая ошибка локализуется в одной ячейке и позволяет продолжить выполнение запроса, то Power Query не остановит обновление и досчитает запрос, но выведет слово **Error** в соответствующих ячейках. При щелчке по фону таких ячеек в нижней части окна появится описание возникшей проблемы:

№	Товар	Цена	Количество	Стоимость
1	Яблоки	120	3	360
2	Апельсины	220	4	880
3	Сливы	80	два	Error
4	Груши	150	3	450

DataFormat.Error: Не удалось преобразовать в число.
Сведения:
два

Если действовать через интерфейс пользователя, то отреагировать на такие **Error** можно несколькими способами:

- Ничего не делать. После выгрузки на лист ячейки с **Error** будут пустыми. Однако оставленные ошибки могут начать порождать другие ошибки (как это произошло в столбце **Стоимость**).
- Удалить строки с ошибками, щёлкнув правой кнопкой мыши по заголовку соответствующего столбца и выбрав в контекстном меню команду **Удалить ошибки (Remove Errors)**. Это, однако, может привести к незаметной потере данных в будущем, которую весьма сложно обнаружить.
- Заменить ошибки на какое-либо заданное значение, щёлкнув по заголовку столбца правой кнопкой мыши и выбрав **Заменить ошибки (Replace Errors)**.

В большинстве случаев последний вариант является наиболее предпочтительным. В Microsoft Excel для подобных случаев существует функция **ЕСЛИОШИБКА¹ (IFERROR)**, а в языке M есть её аналог – программная конструкция **try ... otherwise** со следующим синтаксисом:

try (проверяемое выражение) **otherwise** (что выводим вместо ошибки)

В неё можно «завернуть» потенциально проблемные выражения в M-коде, чтобы перехватить ошибки и вывести вместо них что-то другое. Например, в исходном коде нашего запроса в выражении, где создаётся вычисляемый столбец с суммой:

```
= Table.AddColumn("#Измененный тип", "Стоимость", each [Цена]*[Количество])
```

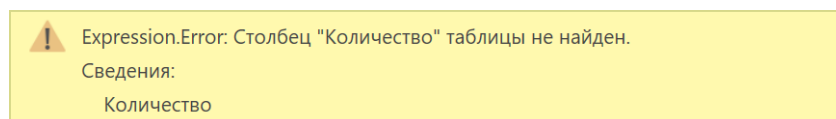
можно добавить перехват ошибки как:

```
= Table.AddColumn("#Измененный тип", "Стоимость", each try [Цена]*[Количество] otherwise 0)
```

Тогда картина на выходе будет уже другая:

	ABC 123	Товар	1.2 Цена	123 Количество	Стоимость
1		Яблоки	120	3	360
2		Апельсины	220	4	880
3		Сливы	80	Error	0
4		Груши	150	3	450

Если же возникшая ошибка слишком серьёзна (например, исходный файл с данными был удалён, были переименованы столбцы и т. д.), то Power Query отреагирует более радикальным образом: выполнение запроса будет прервано, и на экране появится сообщение об ошибке:



Проблема в том, что зачастую одна незначительная ошибка останавливает пересчёт целой цепочки зависимых от неё запросов, полностью обездвиживая весь ваш проект.

Иногда в подобных ситуациях тоже можно использовать конструкцию **try ... otherwise**, но в большинстве случаев здесь уже требуется вмешательство пользователя или другие, более сложные подходы (параметризация, проверка структуры таблиц и т. д.), которые мы подробно рассмотрим в следующих главах.

¹ Подробнее об этой функции см. статью <https://www.planetaexcel.ru/techniques/25/2521/>.

Тонкости деления

*Делю на ноль. Недорого.
(Репетитор по мат. анализу)*

В начальной школе нас всех учили, что на ноль делить нельзя. Чуть позже, в старших классах, выясняется, что на ноль делить всё-таки можно, но в результате мы получим бесконечность. Ещё позже, в вузе, оказывается, что бесконечности бывают разные, и с ними вполне можно производить массу различных действий (складывать, вычитать и т. д.) Нет предела совершенству!

И если Excel в этих вопросах занимает «школьную» позицию и при попытке поделить на ноль выдаёт ошибку **#ДЕЛ/0!** (**#DIV/0!**), то Power Query более продвинут, и всё уже не так банально.

Рассмотрим простой пример. Предположим, что у нас есть два столбца с выручкой и количеством проданного товара, и мы добавляем ещё один вычисляемый столбец через **Добавление столбца** → **Настраиваемый столбец** (**Add Column** → **Custom Column**), где хотим получить цену, т. е. делим первую колонку на вторую:

	1 ² 3 Выручка	ABC 123 Количество	ABC 123 Цена
1	300	5	60
2	1000	0	Infinity
3	-1000	0	-Infinity
4	700	null	null
5	null	100	null
6	null	null	null
7	500	два	Error
8	0	0	NaN
9	0	5	0

Настраиваемый столбец

Имя нового столбца

Цена

Пользовательская формула столбца:

= [Выручка]/[Количество]

Как видите, результат получается далеко не однозначный. Давайте разберём получившуюся картину построчно.

Ошибки (Error)

Ошибка **Error** в 7-й строке, возникающая из-за того, что мы делим число на текст, – уже знакомое нам по прошлой главе дело. Перехватить её можно опять же уже знакомой нам конструкцией **try ... otherwise**, заменив, например, на ноль:

```
=try [Выручка]/[Количество] otherwise 0
```

Пусто (Null)

Строки 4–6 наглядно показывают, что если в числителе и/или знаменателе нашей формулы окажется **null** (т. е. пустая ячейка), то в любом случае в результате мы получаем тоже **null**, как бы неочевидно это не казалось. Проверить и перехватить это дело можно с помощью конструкции:

```
=if [Выручка]/[Количество]=null then 0 else [Выручка]/[Количество]
```

... или:

```
=if [Выручка]=null or [Количество]=null then 0 else [Выручка]/[Количество]
```

Бесконечности (Infinity)

Во второй и третьей строчках хорошо видно, что при делении чисел на ноль мы получаем особый тип результатов – специальные числовые константы положительной и отрицательной бесконечности. Отловить их через **try** уже не получится, т. к. формально это не ошибка. Так что для перехвата и замены подобных бесконечностей придется использовать конструкцию с **if** и со специальными числовыми функциями:

```
if [Выручка]/[Количество] = Number.PositiveInfinity or
   [Выручка]/[Количество] = Number.NegativeInfinity
then 0 else [Выручка]/[Количество]
```

Нечисло (NaN)

Если мы делим ноль на ноль, как в 8-й строке, то получаем в качестве результата особый тип числовых констант – неопределенность, или «нечисло» (NaN = Not a Number). Для обнаружения таких значений в языке M есть специальная функция **Number.IsNaN**, которую можно использовать в связке с **if**:

```
=if Number.IsNaN([Выручка]/[Количество]) then 0 else [Выручка]/[Количество]
```

Универсальный подход

При желании можно соединить всё описанное выше в одну формулу, которая будет проверять все возможные варианты развития событий:

```
if [Выручка]/[Количество] = null or
   [Выручка]/[Количество] = Number.PositiveInfinity or
   [Выручка]/[Количество] = Number.NegativeInfinity or
   Number.IsNaN([Выручка]/[Количество])
then
    0
else
    [Выручка]/[Количество]
```

Если подобное придется делать часто, то имеет смысл создать универсальную пользовательскую функцию для беспрепятственного деления двух любых аргументов:

```
(x,y) =>
if x/y = null or
   x/y = Number.PositiveInfinity or
   x/y = Number.NegativeInfinity or
   Number.IsNaN(x/y)
then
    0
else
    x/y
```

Абсолютные и относительные ссылки в запросах

Думаю, вы уже давно прочувствовали, что принципы работы с данными в Power Query весьма ощутимо отличаются от работы с данными на листе Microsoft Excel. Если в Excel мы думаем и действуем в терминах отдельных ячеек, то Power Query оперирует в основном целыми столбцами.

Однако же могут возникнуть ситуации, когда нам придется и в Power Query ссылаться на отдельные ячейки в наших данных. Причем, как и в Excel, ссылки могут потребоваться абсолютные и относительные. И без редактирования M-кода напрямую здесь уже не обойтись.

Ссылка на конкретную ячейку в столбце

Предположим, что у нас есть вот такая несложная таблица с данными продаж по нескольким автомобильным брендам:

	ABC 123	Бренд	ABC 123	Количество
1		Fiat		54
2		Volkswagen		16
3		Skoda		35
4		Mini		63
5		Hyundai		42
6		Renault		57
7		Subaru		25
8		KIA		79
9		Rover		60
10		BMW		29
11		Honda		52
12		Mercedes		83

Допустим, наша задача состоит в том, чтобы взять *Skoda* за точку отсчета и вычислить разницу между её продажами и всеми остальными марками. Чтобы это реализовать, нам придётся вспомнить пару моментов, которых мы уже касались в главе [Ссылки на элементы таблицы](#).

Во-первых, чтобы сослаться на конкретную ячейку таблицы, используется конструкция:

```
=ИмяТаблицы[ИмяСтолбца]{НомерСтроки}
```

или:

```
=ИмяТаблицы{НомерСтроки}[ИмяСтолбца]
```

Во-вторых, нумерация строк в Power Query начинается с нуля.

Таким образом, если мы загрузили таблицу на шаге с именем, например, **Источник**, то ссылка на ячейку с количеством проданных автомобилей *Skoda* может выглядеть как:

```
=Источник{2}[Количество]
```

И отлчия каждого бренда от *Skoda* легко вычислить, добавив пользовательский столбец через [Добавление столбца](#) → [Настраиваемый столбец](#) (Add Column → Custom Column) с формулой:

	ABC 123	Бренд	ABC 123	Количество	ABC 123	Отличие от Шкоды
1		Fiat		54		-19
2		Volkswagen		16		19
3		Skoda		35		0
4		Mini		63		-28
5		Hyundai		42		-7
6		Renault		57		-22
7		Subaru		25		10
8		KIA		79		-44

Настраиваемый столбец

Имя нового столбца
Отличие от Шкоды

Пользовательская формула столбца:

```
=Источник{2}[Количество]-[Количество]
```

Аналогично, если нам потребуется, например, рассчитать отличие в процентах каждого бренда по отношению к *Skoda*, то это легко сделать формулой:

```
= [Количество]/Источник{2}[Количество] - 1
```

Ссылка на предыдущую/следующую ячейку

Чуть хитрее придётся поступить, если нам нужно получить ссылку на предыдущую или следующую ячейку в определенном столбце таблицы. Предположим, что мы загрузили в Power Query данные по курсу доллара за несколько дней:

	Дата	1.2 Курс
1	01.03.2017	57,9627
2	02.03.2017	58,3776
3	03.03.2017	58,4067
4	04.03.2017	58,9099
5	07.03.2017	58,337
6	08.03.2017	58,263
7	09.03.2017	58,263
8	10.03.2017	58,8318
9	11.03.2017	59,2174
10	14.03.2017	59,1327

Наша задача – вычислить разницу курсов между текущим и предыдущим днем, т. е. наглядно показать динамику курса.

Сначала добавим к нашей таблице столбец индекса через вкладку **Добавление столбца** → **Столбец индекса** → **От 0** (Add Column → Index Column → From 0). Как легко сообразить, числа в этом столбце – это, по сути, и есть номера строк, которые нам нужны, чтобы адресоваться к ячейкам.

Теперь добавим вычисляемый столбец с формулой:

The screenshot shows the Power Query Editor interface. The formula bar at the top contains the following M formula:

```
= Table.AddColumn("#Добавлен индекс", "Отличие от предыдущего", each [Курс] - #"Добавлен индекс"{[Индекс]-1}[Курс])
```

The main table area shows the following columns: **Дата**, **1.2 Курс**, **1.2 Индекс**, and **ABC 123 Отличие от предыдущего**. The data rows are as follows:

1	01.03.2017	57,9627	0	Error
2	02.03.2017	58,3776	1	0,4149
3	03.03.2017	58,4067	2	0,0291
4	04.03.2017	58,9099	3	0,5032
5	07.03.2017	58,337	4	-0,5729
6	08.03.2017	58,263	5	-0,074
7	09.03.2017	58,263	6	0
8	10.03.2017			
9	11.03.2017			
10	14.03.2017			
11	15.03.2017			
12	16.03.2017			
13	17.03.2017			
14	18.03.2017			
15	21.03.2017			
16	22.03.2017			
17	23.03.2017			

The 'Настраиваемый столбец' (Custom Column) dialog box is open, showing the following fields:

- Имя нового столбца:
- Пользовательская формула столбца:
- Доступные столбцы:
 - Дата
 - Курс
 - Индекс

```
= [Курс] - #"Добавлен индекс"{[Индекс]-1}[Курс]
```

Здесь **#"Добавлен индекс"** – это таблица с предыдущего шага с добавленным столбцом индекса, а выражение **{[Индекс] - 1}** как раз и представляет собой ссылку на строку с предыдущим номером из столбца **Курс**.

Единственное, что стоит добавить, – это проверку на номер строки, чтобы избавиться от ошибки **Error** в первой ячейке получившегося столбца. Это легко сделать с помощью **if**:

```
if [Индекс]=0 then
    0
else
    [Курс]-#"Добавлен индекс"{[Индекс]-1}[Курс]
```

Или же можно обойтись чуть более компактной конструкцией с **try**:

```
=try [Курс]-#"Добавлен индекс"{[Индекс]-1}[Курс] otherwise 0
```

Легко сообразить, что в случае, когда нам нужна ссылка не на предыдущую, а на следующую ячейку, придется всего лишь прибавлять, а не вычитать единицу из номера строки в столбце **Индекс**.

Для полноты картины к таблице с результатами запроса можно добавить правило условного форматирования со значками через **Главная** → **Условное форматирование** → **Создать правило** (Home → Conditional formatting → Create Rule), наглядно отобразив динамику курса с помощью зеленых и красных стрелок:

Значок	Значение	Тип
	если значение равно > 0	Число
	если <= 0 и >= 0	Число
	если < 0	

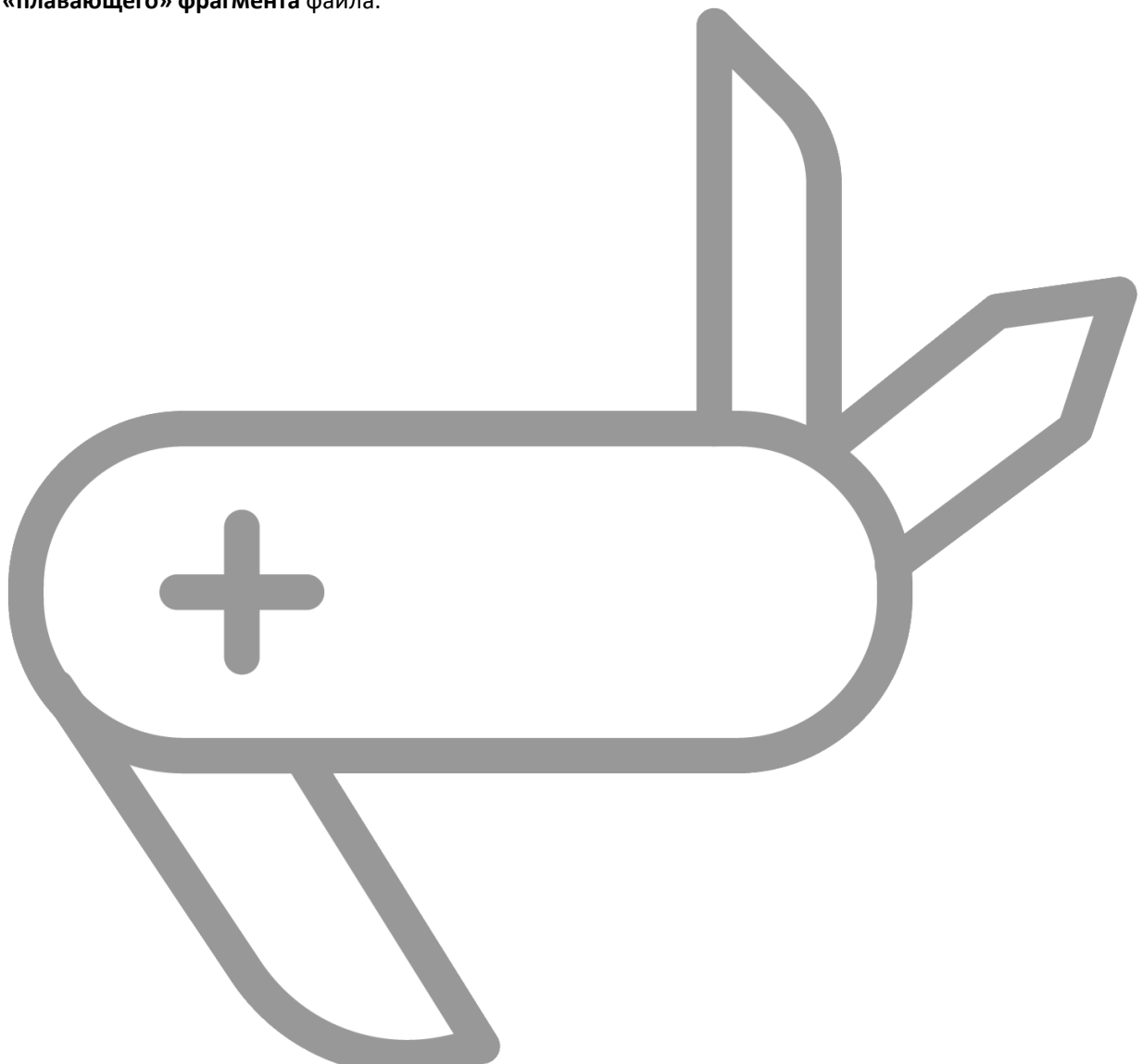
В заключение хочется добавить, что абсолютными ссылками в Power Query лучше не увлекаться без крайней нужды. На пересчёт подобных формул уходит много ресурсов, и при больших объемах данных запросы с такими формулами могут ощутимо подтормаживать.

Параметризация запросов

Параметризация – это возможность заменить некоторые жёстко прописанные в запросе константы (условия фильтрации, путь к данным и т. п.) на параметры – переменные, которые берутся, например, из ячеек листа или ещё откуда-нибудь. Это позволит сделать ваши запросы более универсальными и легче делиться ими с другими пользователями.

В этой главе мы:

- научимся **добавлять простые и сложные параметры** к вашим запросам;
- **параметризовать путь к исходным данным** в вашем запросе, чтобы он без ошибок выполнялся на любых компьютерах;
- рассмотрим **параметризацию веб-запроса**, чтобы суметь загрузить из интернета несколько страниц сразу;
- используем параметризацию, чтобы загружать данные из заранее неизвестного по положению и размерам «плавающего» фрагмента файла.



Добавление простых параметров к запросу

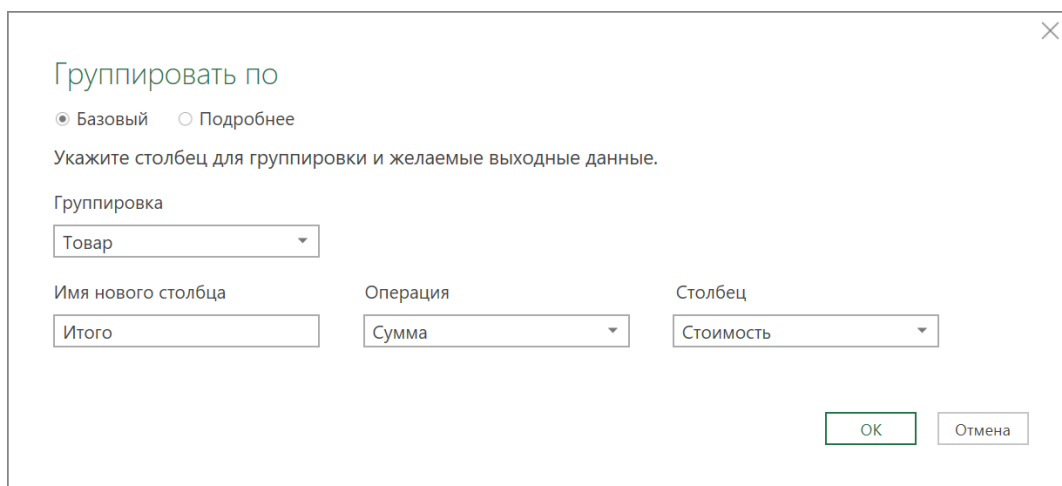
Разберём сначала простой случай параметризации на следующем примере. Предположим, что у нас есть таблица с продажами вот такого вида:

	A	B	C
1	Товар	Дата	Стоимость
2	Клюква	23.09.2016	20300
3	Черника	27.09.2018	62200
4	Клюква	30.09.2018	44500
5	Картофель	12.05.2017	67100
6	Чеснок	06.05.2015	9700
7	Тунец	06.09.2018	94500
8	Капуста	28.07.2016	71300
9	Рис	07.12.2015	41500
10	Капуста	07.07.2016	47800
11	Грибы	11.07.2017	21600
12	Груша	15.12.2017	90200
13	Пшеница	08.01.2015	8600
14	Тунец	09.07.2018	200
15	Лосось	11.09.2018	28200
16	Оливки	19.11.2017	4100

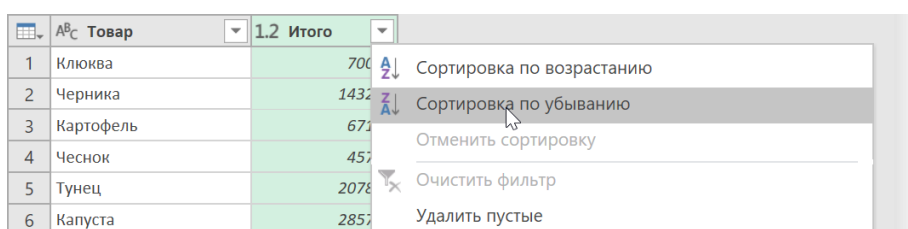
Нам нужно подсчитать суммарные итоги по каждому товару, отсортировать получившиеся результаты по убыванию и оставить только топ-5 лучших продуктов.

Если вы читали в этой книге главу про группировку данных, то подобная задача не будет для вас сложной. После загрузки таблицы в Power Query нужно:

1. На вкладке **Преобразование** нажать на кнопку **Группировать по** (Transform → Group By) и задать группировку по столбцу **Товар** с выводом суммы по колонке **Стоимость**:



2. Отсортировать получившуюся таблицу по убыванию сумм с помощью кнопки фильтра в шапке:



3. Оставить только пять верхних строк, воспользовавшись командой **Главная → Сохранить строки → Сохранить верхние строки** (Home → Keep Rows → Keep Top Rows) и ввести затем количество строк, которое мы хотим оставить.

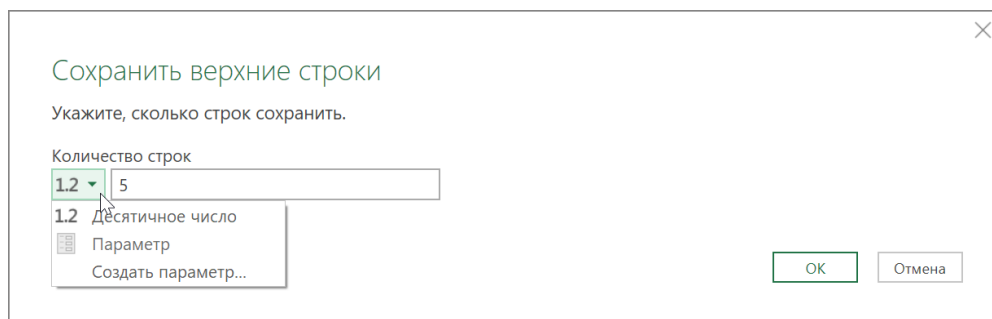
В итоге получим топ-5 товаров с максимальными продажами:

	Товар	Итого
1	Брюссельская капуста	499900
2	Ямс	365000
3	Кукуруза	343200
4	Квиноа	312800
5	Персик	302500

Пока всё просто, так?

Теперь слегка усложним задачу и представим, что вводные, которые мы получаем от руководителя, клиента или заказчика, постоянно меняются. Сегодня нас просят сделать топ-5, завтра будет нужен уже топ-10, послезавтра – топ-3 и т. д. То есть количество строк, которое нужно оставить на последнем шаге, может быть переменной величиной.

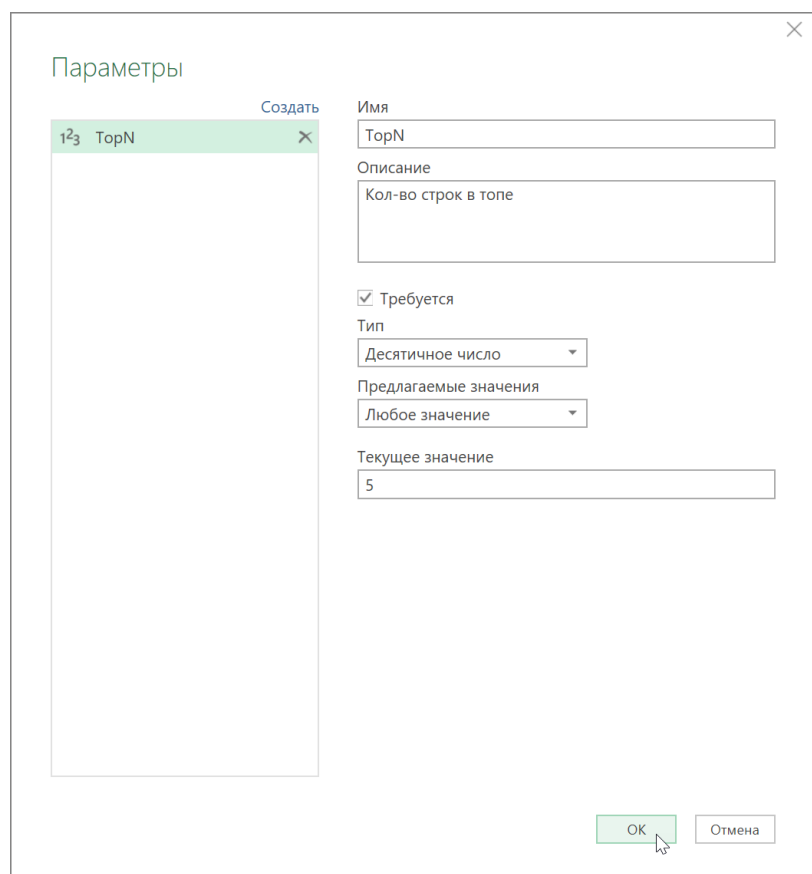
Для такого случая в Power Query есть возможность задавать большинство значений в диалоговых окнах как параметры. Об этом, собственно, и сигнализирует выпадающий список слева от поля ввода:



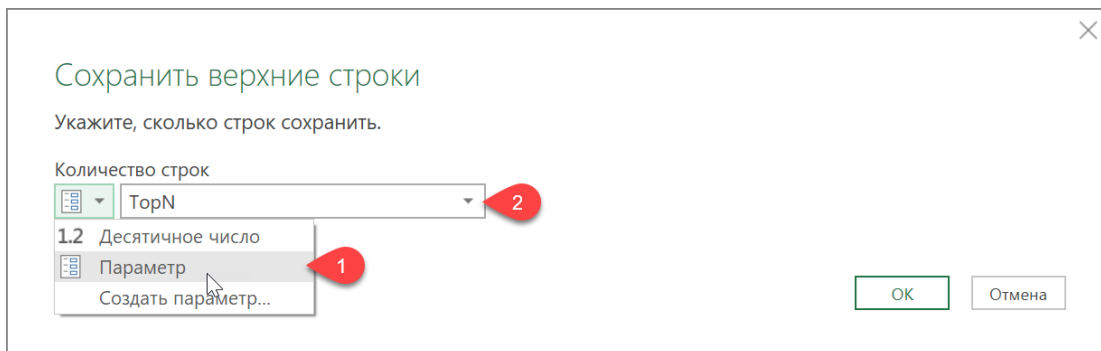
Чтобы использовать эту возможность, параметр сначала нужно создать. Для этого выберем **Главная** → **Управление параметрами** → **Создать параметр** (Home → Manage Parameters → New parameter) и в открывшемся окне введём следующие данные:

- **Имя** параметра (Name) – любое подходящее.
- **Описание** (Description) — по желанию
- Флажок **Требуется** (Required) включаем, если значение этого параметра не может остаться пустым, т. е. его обязательно нужно ввести.
- **Тип** (Type) – тип данных, которые будет хранить параметр (числа, текст, даты и т. д.).
- **Предлагаемые значения** (Suggested values) – значение по умолчанию.
- **Текущее значение** (Current value).

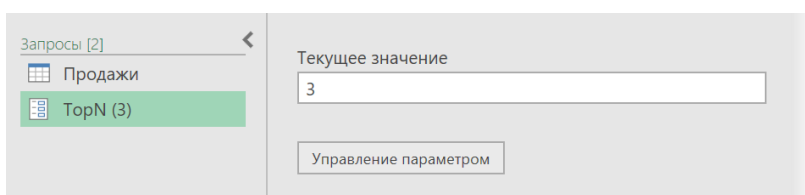
Нажмем **OK**, чтобы создать параметр, и вернемся к нашему исходному запросу.



Если щёлкнуть мышью по значку справа от последнего шага **Сохраненные первые строки (Keep Top Rows)**, то мы вернёмся в предыдущее диалоговое окно и сможем теперь выбрать созданный параметр **TopN** в качестве входного значения для количества оставляемых строк:



После нажатия на **OK** можно поэкспериментировать, задавая разные значения параметра в левом верхнем углу окна в списке запросов:



Это будет приводить к мгновенному автоматическому обновлению запроса и выводу уже не Топ-5, а именно того количества строк, которое нам нужно:

	АВ _С Товар	1.2 Итого
1	Брюссельская капуста	499900
2	Ямс	365000
3	Кукуруза	343200

Плюс такого подхода по сравнению с ручной правкой шага в том, что один и тот же параметр может использоваться в запросе несколько раз. Тогда, единожды поменяв его значение, мы получим изменения сразу во всех фрагментах, где наш параметр встречается.

Минус же в том, что для изменения параметра нам всё равно нужно открывать окно Power Query, что может быть сложновато для неопытных пользователей, которые будут работать с файлом. Обойти это ограничение можно, если брать значение параметра напрямую из какой-нибудь ячейки листа Excel. В следующей главе мы как раз и разберёмся с этим трюком.

Параметризация путей к файлам исходных данных

Если вы начнёте пересылать коллегам файлы с созданными в них запросами в Power Query, то очень скоро столкнётесь с одной узкоспециальной, но весьма надоедливой проблемой, связанной с постоянно ломающимися ссылками на исходные данные.

Суть этих «граблей» в том, что если в своём запросе вы ссылаетесь на внешние файлы или папки, то Power Query жёстко пропишет абсолютный путь к ним в тексте запроса. У вас на компьютере всё работает прекрасно, но если отправить файл с запросом другим пользователям, то их ждёт разочарование, т. к. у них на компьютере путь к исходным данным уже другой, и наш запрос работать не будет.

Что же сделать в такой ситуации? Давайте рассмотрим этот случай подробнее на следующем примере.

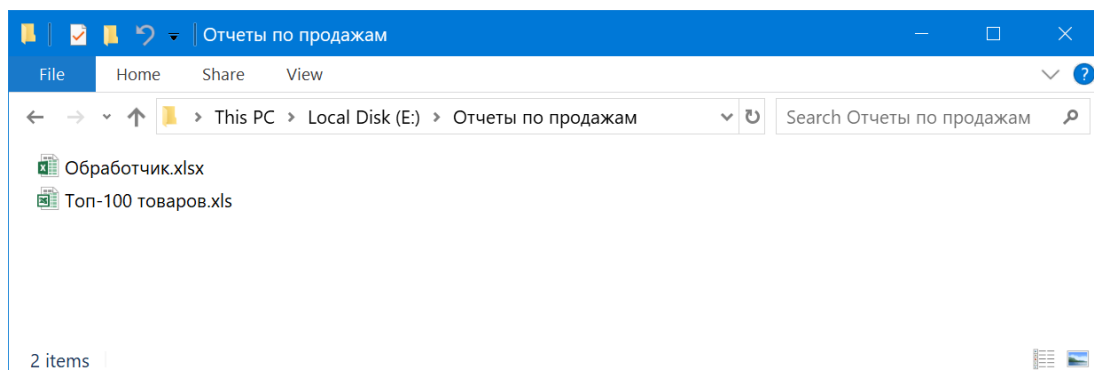
Постановка задачи

Предположим, что у нас в папке **E:\Отчеты по продажам** лежит файл **Топ-100 товаров.xls**, представляющий собой выгрузку из нашей корпоративной базы данных или ERP-системы (1С, SAP и т. п.) Этот файл содержит информацию о наиболее популярных товарных позициях и выглядит внутри примерно так:

№	п/п	Товар	Ед. изм.	Продажи, количество	Выручка, руб.	Закупочная цена, руб.	Розничная цена, руб.	Наценка, %
1	23829	ЯЙЦО КУРИНОЕ 10ШТ С1 ТАГАНРОГ	ШТ	1 298,000	54 544,49	55,00	0,00	-100,00
2	14113	МАККОФЕ 3-В-1 20 Г	ШТ	613,000	6 112,90	10,00	0,00	-100,00
3	24323	КРУАССАН С СГУЩЕНКА ВАР. 0.075КГ	ШТ	511,000	6 637,54	13,00	0,00	-100,00
4	23294	КАРТОФЕЛЬ ЖЕЛТ.	КГ	510,464	13 667,67	32,00	0,00	-100,00
5	24324	КРУАССАН С ШОКОЛОДНОЙ НАЧ. 0.075КГ	ШТ	500,000	6 483,29	13,00	0,00	-100,00
6	24101	БАТОН НАРЕЗНОЙ В НАРЕЗКЕ 0.380Г	ШТ	490,000	10 251,25	21,00	0,00	-100,00
7	24313	БАТОНЧИК АЮТИНСКИЙ В НАР. 0,2 КГ	ШТ	484,000	5 782,20	12,00	0,00	-100,00
8	18178	ЖЕВАТЕЛЬНАЯ РЕЗИНКА LOVE IS.....В АССОРТИМЕНТЕ ШТ	ШТ	376,000	1 124,61	3,00	0,00	-100,00
9	02269	ЛАПША ДОШИРАК КУРИЦА 90Г	ШТ	355,500	11 694,64	33,00	0,00	-100,00
10	05212	СНИКЕРС СУПЕР 95 Г	ШТ	331,000	18 193,34	55,00	0,00	-100,00
11	24051	КУРИЦА ГРИЛЬ	КГ	300,360	53 900,60	180,00	0,00	-100,00
12	21739	ЛАПША АНАКОМ КУРИНЫЙ 60Г	ШТ	287,350	2 822,24	10,00	0,00	-100,00

С ходу очевидно, что работать с ним в Excel в таком виде практически невозможно: будут мешать пустые строки через одну с данными, объединенные ячейки, лишние столбцы, многоуровневая шапка и т. д.

Поэтому рядом с этим файлом в той же папке мы создаём ещё один новый файл **Обработчик.xlsx**, в котором создадим запрос Power Query, который будет загружать страшенькие данные из исходного файла-выгрузки **Топ-100 товаров.xls** и приводить их в порядок:



Создаем запрос к внешнему файлу

Сначала создадим запрос к файлу **Топ-100 товаров.xlsx**, чтобы загрузить из него данные, как мы это уже делали в главе [Загрузка данных из внешней книги Excel](#). Открыв файл **Обработчик.xlsx**, выберем на вкладке **Данные** команду **Получить данные** → **Из файла** → **Из книги Excel** (Data → Get Data → From file → From Excel), затем укажем местоположение исходного файла и нужный нам лист. Выбранные данные загрузятся в редактор Power Query:

№	Артикул	Товар	Ед. изм.	Продажи, количество	Выручка, руб.
1					
2					
3					
4	Дата печати: 28.02.2018				
5					
6					
7					
8	Лучшие товары (100)				
9	Места хранения: Торговый зал ип				
10					
11	№ п/п	Товар	Ед. изм.		
12					
13					
14					
15	1	23829 ЯЙЦО КУРИНОЕ 10ШТ С1 ТАГАНРОГ	ШТ	1298	54544,49
16					
17	2	14113 МАККОФЕ 3-В-1 20 Г	ШТ	613	6112,9
18					
19	3	24323 КРУАССАН С СГУЩЕНКА ВАР. 0.075КГ	ШТ	511	6637,54
20					
21	4	23294 КАРТОФЕЛЬ ЖЕЛТ.	КГ	510,464	13667,67
22					

Приведём их в нормальный вид:

1. Удалим пустые строки через **Главная** → **Удалить строки** → **Удалить пустые строки** (Home → Remove Rows → Remove Empty Rows).
2. Удалим ненужные четыре верхние строки через **Главная** → **Удалить строки** → **Удалить верхние строки** (Home → Remove Rows → Remove Top Rows).
3. Поднимем первую строку в шапку таблицы кнопкой **Использовать первую строку в качестве заголовков** на вкладке **Главная** (Home → Use first row as headers).
4. Отделим пятизначный артикул от названия товара во втором столбце, используя команду **Разделить столбец** на вкладке **Преобразование** (Transform → Split Column).
5. Удалим ненужные столбцы и переименуем заголовки оставшихся для лучшей наглядности.

В итоге у нас должна получиться следующая, гораздо более приятная, картина:

№	Артикул	Товар	Ед. изм.	1.2 Продажи, количество	1.2 Выручка, руб.
1	1	23829 ЯЙЦО КУРИНОЕ 10ШТ С1 ТАГАНРОГ	ШТ	1298	54544,49
2	2	14113 МАККОФЕ 3-В-1 20 Г	ШТ	613	6112,9
3	3	24323 КРУАССАН С СГУЩЕНКА ВАР. 0.075КГ	ШТ	511	6637,54
4	4	23294 КАРТОФЕЛЬ ЖЕЛТ.	КГ	510,464	13667,67
5	5	24324 КРУАССАН С ШОКОЛОДНОЙ НАЧ. 0.075КГ	ШТ	500	6483,29
6	6	24101 БАТОН НАРЕЗНОЙ В НАРЕЗКЕ 0.380Г	ШТ	490	10251,25
7	7	24313 БАТОНЧИК АЮТИНСКИЙ В НАР. 0,2 КГ	ШТ	484	5782,2
8	8	18178 ЖЕВАТЕЛЬНАЯ РЕЗИНКА LOVE IS.....В АССОРТИМЕНТЕ ...	ШТ	376	1124,61
9	9	02269 ЛАПША ДОШИРАК КУРИЦА 90Г	ШТ	355,5	11694,64
10	10	05212 СНИКЕРС СУПЕР 95 Г	ШТ	331	18193,34
11	11	24051 КУРИЦА ГРИЛЬ	КГ	300,36	53900,6
12	12	21739 ЛАПША АНАКОМ КУРИНЫЙ 60Г	ШТ	287,35	2822,24
13	13	24473 ЖЕВ.МАРМ. МАЯМА ЧЕРВЯК	ШТ	282	452,11
14	14	24326 СЛОЙКА С ВИШНЕВОЙ НАЧ.100Г АЮТ	ШТ	279	4451,88
15	15	24299 ГАМБУРГЕР	ШТ	279	12540,95

Осталось эту облагороженную таблицу выгрузить обратно на лист в наш файл **Обработчик.xlsx** командой **Закреть и загрузить** (Home → Close&Load) на вкладке **Главная**:

№п/п	Артикул	Товар	Ед.изм.	Продажи, количество	Выручка, руб.
1	23829	ЯЙЦО КУРИНОЕ 10ШТ С1 ТАГАНРОГ	ШТ	1298	54544,49
2	14113	МАККОФЕ 3-В-1 20 Г	ШТ	613	6112,9
3	24323	КРУАССАН С СГУЩЕНКА ВАР. 0.075КГ	ШТ	511	6637,54
4	23294	КАРТОФЕЛЬ ЖЕЛТ.	КГ	510,464	13667,67
5	24324	КРУАССАН С ШОКОЛОДНОЙ НАЧ. 0.075КГ	ШТ	500	6483,29
6	24101	БАТОН НАРЕЗНОЙ В НАРЕЗКЕ 0.380Г	ШТ	490	10251,25
7	24313	БАТОНЧИК АЮТИНСКИЙ В НАР. 0,2 КГ	ШТ	484	5782,2
8	18178	ЖЕВАТЕЛЬНАЯ РЕЗИНКА LOVE IS.....В АССОРТИМЕ	ШТ	376	1124,61
9	02269	ЛАПША ДОШИРАК КУРИЦА 90Г	ШТ	355,5	11694,64
11	05212	СНИКЕРС СУПЕР 95 Г	ШТ	331	18193,34
12	24051	КУРИЦА ГРИЛЬ	КГ	300,36	53900,6
13	21739	ЛАПША АНАКОМ КУРИНЫЙ 60Г	ШТ	287,35	2822,24
14	24473	ЖЕВ.МАРМ. МАЯМА ЧЕРВЯК	ШТ	282	452,11
15	24326	СЛОЙКА С ВИШНЕВОЙ НАЧ.100Г АЮТ	ШТ	279	4451,88
16	24299	ГАМБУРГЕР	ШТ	279	12540,95
17	24327	СЛОЙКА С ЯБЛОЧ. НАЧ. АЮТА 100Г	ШТ	269	4286,03
18	00080	ЛАПША ДОШИРАК ГОВЯДИНА 90 Г	ШТ	267,018	8809,59
19	24881	ПИЦЦА	ШТ	261	15603,51
20	24328	СЛОЙКА С ТВОРОЖН. НАЧ. 100Г АЮТ	ШТ	261	4164,77
21	01573	БАНАНЫ ВЕС	КГ	260,356	20533,74
22	24142	ФАХИТОС	ШТ	250	18726,22
23	24321	БУЛОЧКА С ВАНИЛ.НАЧ АЮТА 0.090КГ	ШТ	249	3222,64
24	24052	МОЛОКО НОВАЯ ДЕРЕВНЯ ПЭК 2.5%	ШТ	246	10106,54
25	24119	ПУРИ	ШТ	231	6894,35
26	25337	КОЗИНАК ПОДСОЛНЕЧНЫЙ АЗОВ ШТ	ШТ	224	3358,95
27	24320	БУЛОЧКА С МАК.НАЧИНОК 0.090КГ	ШТ	220	2854,5
28	24721	КРУАССАН С КЛУБНИКОЙ 0.075КГ	ШТ	214	2775,99
29	22978	МОЛОКО КУБАНС.БУРЕНКА 2.5% 930Г	ШТ	209	14176,07
30	23753	ПЛЮШКА СДОБНАЯ АЮТА	ШТ	209	2689,42
31	21740	ЛАПША АНАКОМ ГОВЯЖИЙ 60Г	ШТ	200,291	1986,11
32	24566	НАПИТОК МИСЛОМОЛОЧНЫЙ ФРУКТОВ КЛУБНИКА ШТ	ШТ	197	7467,26

Находим путь к файлу в запросе

Теперь давайте посмотрим, как выглядит наш запрос «под капотом», на встроенном в Power Query внутреннем языке M. Для этого вернемся в наш запрос двойным щелчком по нему в правой панели **Запросы и подключения** и на вкладке **Просмотр** выберем **Расширенный редактор** (View → Advanced Editor). В открывшемся окне во второй строке сразу же обнаруживается жёстко прописанный путь к нашему исходному файлу загрузки:

File | Главная | Преобразование | Добавление столбца | **Просмотр** | Параметры

Параметры запроса | Структура | Предварительный просмотр данных | Столбцы | Параметры | **Расширенный редактор** | Зависимости запроса

```
= Table.RenameColumns("#Измeненный тип1",{"Товар#(1f).1", "Артикул"}, {"Товар#(1f).2", "Товар"})
```

Расширенный редактор

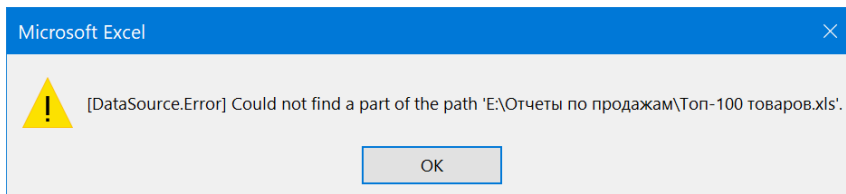
Загрузка ТОП100

```
let
    Источник = Excel.Workbook(File.Contents("E:\Отчеты по продажам\Топ-100 товаров.xls"), null, true),
    Лист2 = Источник{[Name="Лист1"]}[Data],
    #"Удалены пустые строки" = Table.SelectRows(Лист2, each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(_), {"
    #"Удаленные верхние строки" = Table.Skip(#"Удалены пустые строки",4),
    #"Повышенные заголовки" = Table.PromoteHeaders(#"Удаленные верхние строки", [PromoteAllScalars=true]),
    #"Измeненный тип" = Table.TransformColumnTypes(#"Повышенные заголовки",{{"№(1f)n/n", Int64.Type}, {"Товар#(1f)", type text
    #"Другие удаленные столбцы" = Table.SelectColumns(#Измeненный тип",{{"№(1f)n/n", "Товар#(1f)", "Ед.#(1f)изм.", "Продажи, к
    #"Разделить столбец по разделителю" = Table.SplitColumn(#"Другие удаленные столбцы", "Товар#(1f)", Splitter.SplitTextByEach
    #"Измeненный тип1" = Table.TransformColumnTypes(#"Разделить столбец по разделителю",{{"Товар#(1f).1", type text}, {"Товар#(
    #"Переименованные столбцы" = Table.RenameColumns(#Измeненный тип1,{"Товар#(1f).1", "Артикул"}, {"Товар#(1f).2", "Товар"}
in
    #"Переименованные столбцы"
```

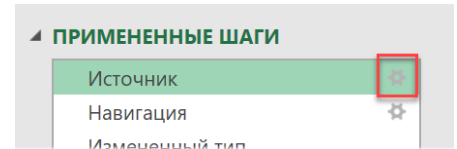
✓ Синтаксические ошибки не обнаружены.

Готово | Отмена

Таким образом, если мы, например, перешлём этот файл нашему коллеге, и он попытается его обновить, то Power Query выдаст ошибку, т. к. не найдет источник данных на компьютере коллеги путь к исходному файлу **Топ-100 товаров.xls** будет уже, скорее всего, другим:



Конечно, можно просто открыть редактор Power Query и, нажав на значок шестерёнки справа от первого шага **Источник (Source)**, заново выбрать местоположение исходного файла, но учитывайте, что это придется сделать не вам, а вашему коллеге, который, вполне возможно, понятия не имеет о существовании какого-то Power Query и никогда в нём не работал. Объяснить, как всё это проделать, да ещё и по телефону – тот ещё квест!

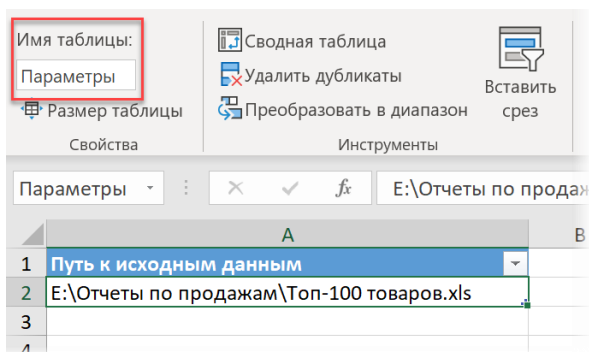


Поэтому лучше в принципе исключить подобную ситуацию. Если мы сможем заменить жёстко прописанный путь к исходному файлу на ссылку на ячейку листа Excel, где нужное местоположение будет заранее введено, то всё будет гораздо проще.

Давайте рассмотрим универсальный способ это реализовать, работающий, насколько мне известно, в любой версии Power Query.

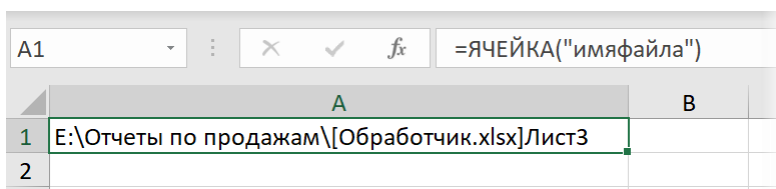
Вводим путь как параметр

Вернёмся в наш файл **Обработчик.xlsx**. Добавим новый пустой лист и сделаем на нём маленькую «умную» таблицу, в единственной ячейке которой будет записан полный путь к нашему файлу исходных данных:



Для создания «умной» таблицы из обычного диапазона можно, как обычно, использовать сочетание клавиш **Ctrl+T** или кнопку **Форматировать как таблицу** на вкладке **Главная (Home → Format as Table)**. Заголовок столбца (ячейка A1) может быть совершенно любым. Также обратите внимание, что для понятности я дал таблице имя *Параметры* на вкладке **Конструктор (Design)**.

Скопировать из Проводника путь или даже ввести его вручную в ячейку A2 не представляет, конечно, особой сложности, но лучше всего минимизировать человеческий фактор и определять путь автоматически. Это можно реализовать с помощью стандартной функции рабочего листа Excel **ЯЧЕЙКА (CELL)**, которая умеет выдавать кучу полезной информации про указанную в качестве аргумента ячейку, в том числе и путь к текущему файлу:



Если предположить, что файл с исходными данными всегда будет находиться в той же папке, что и наш **Обработчик**, то путь, который нам нужен, можно сформировать следующей формулой:

Параметры		A	B	C	D	E	F	G	H	I
1	Путь к исходным данным									
2	E:\Отчеты по продажам\Топ-100 товаров.xls									
3										

```
=ЛЕВСИМВ(ЯЧЕЙКА("имяфайла");НАЙТИ("[";ЯЧЕЙКА("имяфайла"))-1)&"Топ-100 товаров.xls"
```

или в английской версии:

```
=LEFT(CELL("filename");FIND("[";CELL("filename"))-1)&"Топ-100 товаров.xls"
```

...где функция **НАЙТИ** (FIND) ищет позицию открывающей квадратной скобки, а потом функция **ЛЕВСИМВ** (LEFT) берёт из полной ссылки кусок текста до открывающей квадратной скобки (т. е. путь к текущей папке), а затем к нему приклеивается имя и расширение нашего исходного файла с данными.

Остался последний и самый главный штрих – прописать в запросе путь к исходному файлу **Топ-100 товаров.xls**, сославшись на ячейку A2 нашей созданной «умной» таблицы *Параметры*. Для этого вернемся в запрос Power Query и ещё раз откроем **Расширенный редактор** на вкладке **Просмотр** (View → Advanced Editor). Вместо текстовой строки-пути в кавычках **"E:\Отчеты по продажам\Топ-100 товаров.xls"** введём туда вот такую конструкцию:

Расширенный редактор

Загрузка ТОП100

```
let
    Источник = Excel.Workbook(File.Contents(Excel.CurrentWorkbook(){[Name="Параметры"]}[Content]{0}[Путь к исходным данным]), null, true),
    Лист2 = Источник[[Name="Лист1"]][Data],
    #"Удалены пустые строки" = Table.SelectRows(Лист2, each not List.IsEmpty(List.RemoveMatchingItems(Record.FieldValues(_), {"", null}))),
    #"Удаленные верхние строки" = Table.Skip(#"Удалены пустые строки",4),
    #"Повышенные заголовки" = Table.PromoteHeaders(#"Удаленные верхние строки", [PromoteAllScalars=true]),
    #"Измененный тип" = Table.TransformColumnTypes(#"Повышенные заголовки",{{"Товар#(1f)", type text}, {"Column3", type text}},),
    #"Другие удаленные столбцы" = Table.SelectColumns(#"Измененный тип",{{"Товар#(1f)", "Ед.#(1f)изм.", "Продажи, количество", "Выручка", "Артикул"}},),
    #"Разделить столбец по разделителю" = Table.SplitColumn(#"Другие удаленные столбцы", "Товар#(1f)", Splitter.SplitTextByEachDelimiter({" "}, Quoted=false),),
    #"Измененный тип1" = Table.TransformColumnTypes(#"Разделить столбец по разделителю",{{"Товар#(1f).1", type text}, {"Товар#(1f).2", type text}},),
    #"Переименованные столбцы" = Table.RenameColumns(#"Измененный тип1",{{"Товар#(1f).1", "Артикул"}, {"Товар#(1f).2", "Товар"}},)
in
    #"Переименованные столбцы"
```

✓ Синтаксические ошибки не обнаружены.

```
Excel.CurrentWorkbook(){[Name="Параметры"]}[Content]{0}[Путь к исходным данным]
```

Давайте разберемся, из чего она состоит.

- **Excel.CurrentWorkbook()** – это функция языка M для обращения к содержимому текущего файла. Мы уже использовали её в главе [Загрузка данных из текущей книги Excel](#), если помните.
- **{[Name="Параметры"]}[Content]** – это уточняющий параметр к предыдущей функции, указывающий, что мы хотим получить содержимое «умной» таблицы с именем *Параметры*
- **[Путь к исходным данным]** – это имя столбца в таблице *Параметры*, к которому мы обращаемся
- **{0}** – это номер строки в таблице *Параметры*, из которой мы хотим взять данные. Шапка не в счет, и нумерация начинается от нуля, а не от единицы.

Вот и всё, собственно.

Осталось нажать на **Готово** и проверить, как работает наш запрос. Теперь при пересылке всей папки с обоими файлами внутри на другой ПК запрос будет сохранять работоспособность и определять путь к данным автоматически.

Преобразование запроса в функцию на примере веб-запроса курса валют

Очень мощной технологией, позволяющей отвязать алгоритмы обработки от исходных данных, является преобразование запросов в пользовательские функции. Это позволит использовать единожды созданные запросы множество раз, кратно увеличивая ваш КПД.

Представьте себе следующую задачу: у нас есть таблица платежей (покупок, заказов), для каждого из которых необходимо подгрузить с сайта Центрального Банка РФ курс евро на этот день. На сайте ЦБ этот функционал представлен соответствующей страницей, адрес которой и содержит требуемую дату как параметр. Например, для 1 февраля 2019 г. это будет http://cbr.ru/currency_base/daily/?date_req=01.02.2019, а сама веб-страница при этом выглядит так:

База данных по курсам валют

Официальные курсы валют на заданную дату, устанавливаемые ежедневно

Центральный банк Российской Федерации установил с 01.02.2019 следующие курсы иностранных валют к рублю Российской Федерации без обязательств Банка России покупать или продавать указанные валюты по данному курсу

Цифр. код	Букв. код	Единиц	Валюта	Курс
036	AUD	1	Австралийский доллар	47,5608
944	AZN	1	Азербайджанский манат	38,5250
051	AMD	100	Армянских драмов	13,3985
933	BYN	1	Белорусский рубль	30,4060
975	BGN	1	Болгарский лев	38,4389
986	BRL	1	Бразильский реал	17,7361
348	HUF	100	Венгерских форинтов	23,8384
410	KRW	1000	Вон Республики Корея	58,7379

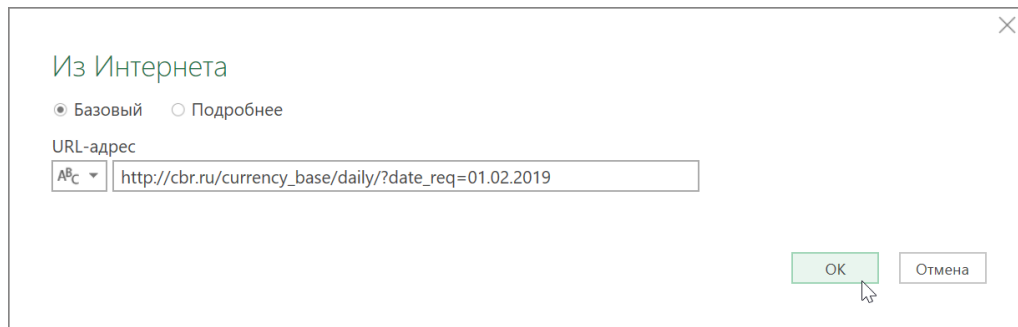
Логика решения здесь будет следующая:

1. Сначала **сделаем простой веб-запрос**, который будет загружать курс евро на любую произвольно выбранную постоянную дату (например, на то же самое 1 февраля).
2. **Преобразуем наш запрос в функцию**, в которой дата будет входным аргументом, т. е. переменной.
3. **Применим созданную функцию ко всем датам**, чтобы получить курс для каждой из них.

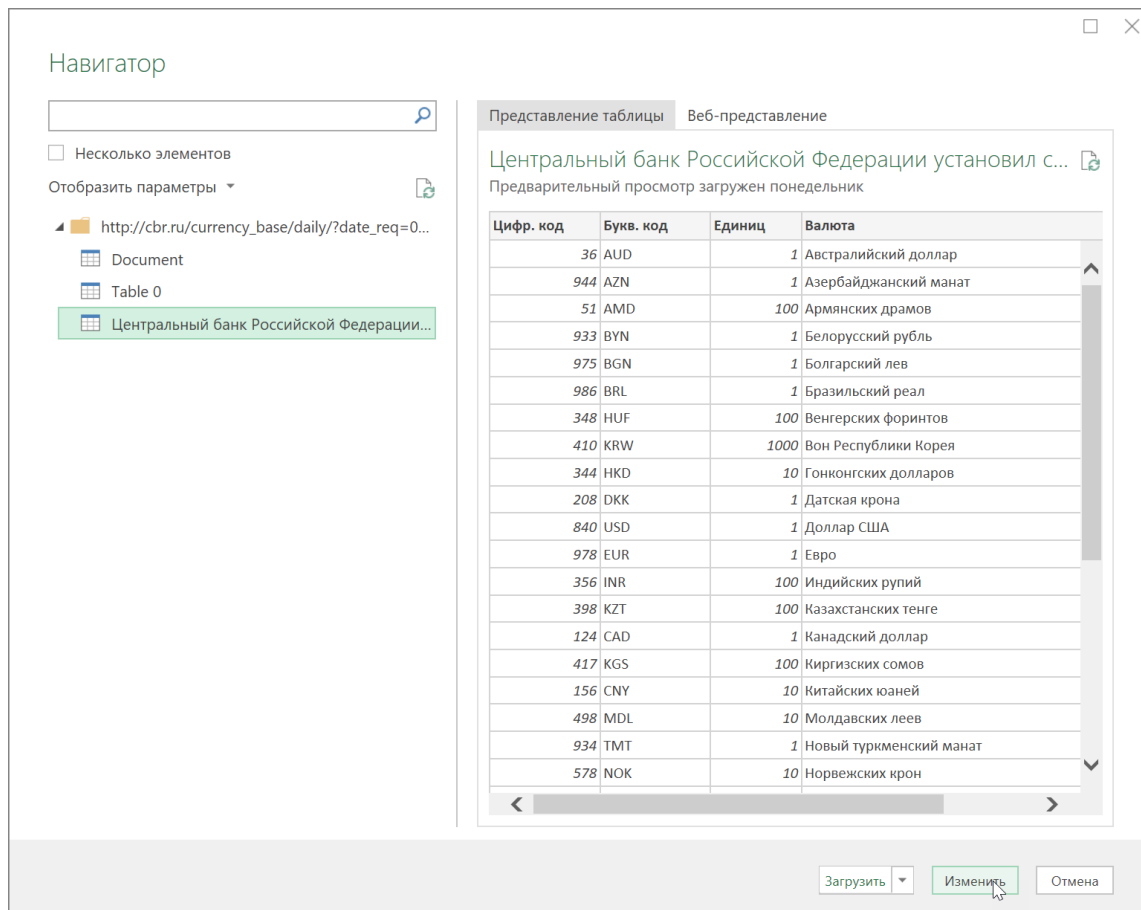
Этап 1. Создаём простой веб-запрос

Тут ничего революционного. Можете при желании перечитать главу про импорт данных из интернета, чтобы обновить знания.

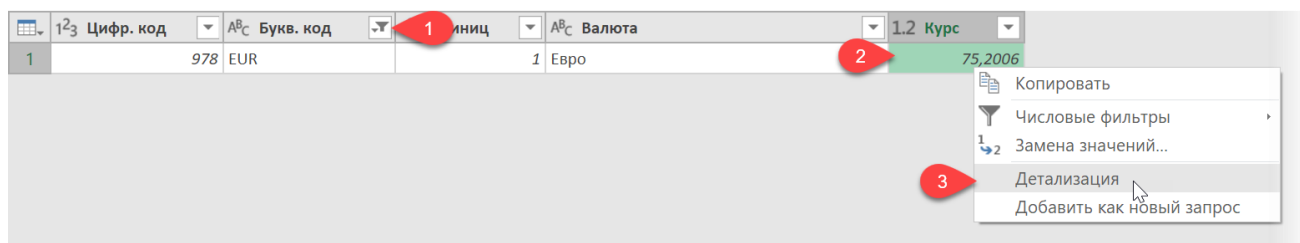
На вкладке **Данные** жмём кнопку **Получить данные** → **Из других источников** → **Из интернета** (Data → Get Data → From Other Sources → From Internet) и вводим адрес страницы, откуда будем грузить данные:



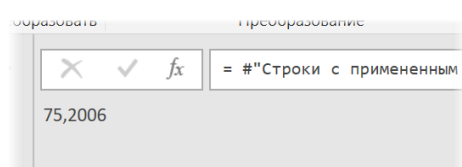
В следующем окне **Навигатора** находим нужную нам таблицу с курсами в списке слева и жмём на кнопку **Изменить (Edit)** внизу:



После загрузки таблицы со всеми курсами в Power Query отфильтруем только нужную нам строку (EUR) по столбцу **Букв. код** и затем щёлкнем правой кнопкой мыши по значению курса в последней ячейке и выберем команду **Детализация (Drill down)**:



На выходе мы получим то, к чему стремились, – числовое значение курса евро на 1 февраля 2019 г.:

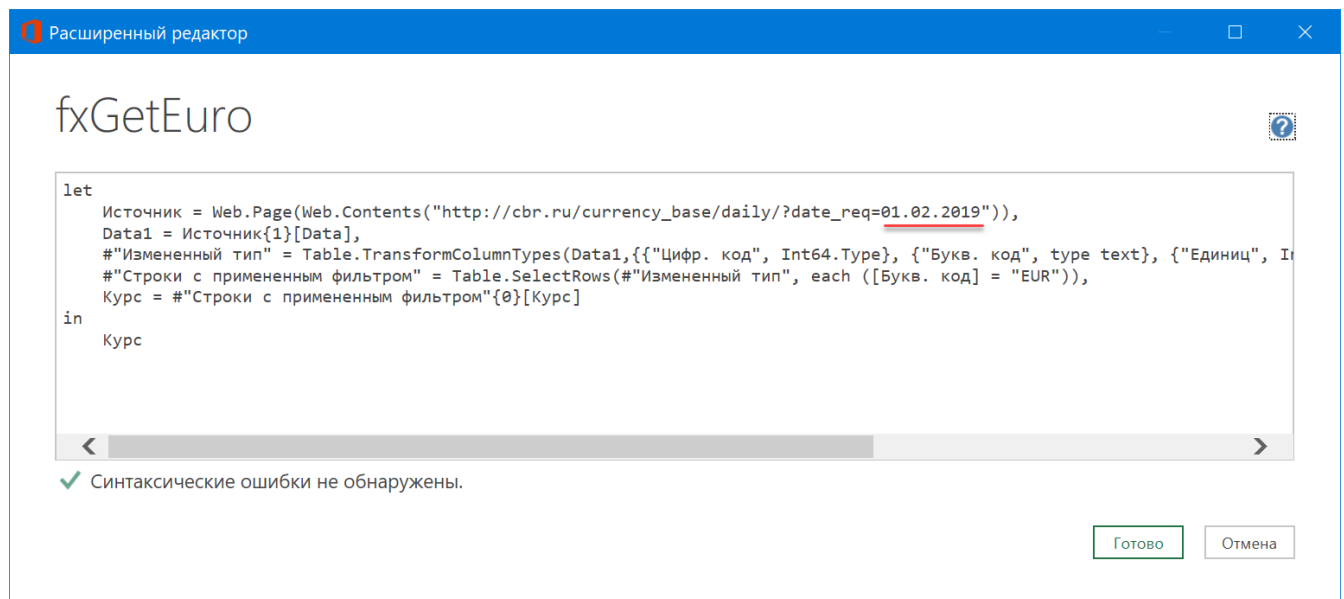


Теперь дадим нашему запросу любое подходящее имя (например, **Курс**) и сохраним его как подключение через **Главная → Закрывать и загрузить → Закрывать и загрузить в... → Только создать подключение** (Home → Close&Load → Close&Load to... → Create only connection).

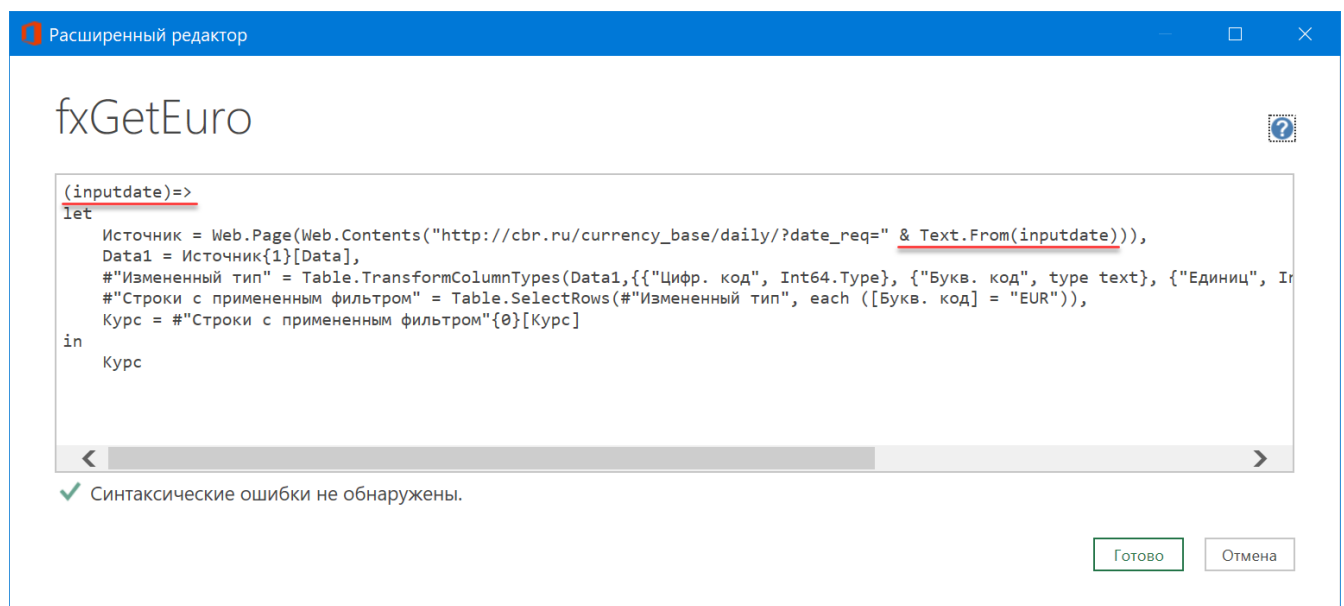
Этап 2. Преобразуем запрос в функцию

Сделаем копию нашего запроса, щёлкнув по нему правой кнопкой мыши и выбрав команду **Дублировать** (**Duplicate**). Используя команду там же в контекстном меню, переименуем созданный дубликат в **fxGetEuro** и откроем его в редакторе Power Query.

Перейдем на вкладку **Просмотр** (**View**) и нажмем на кнопку **Расширенный редактор** (**Advanced Editor**), чтобы просмотреть исходный код запроса на языке M:

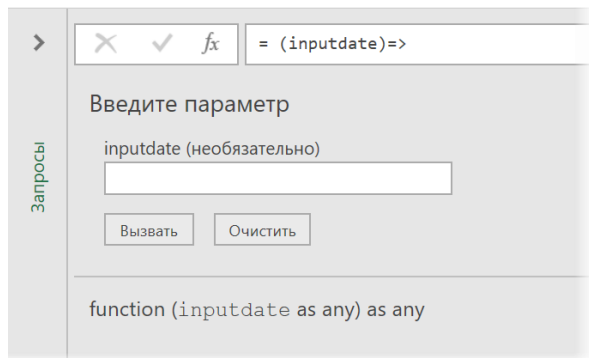


Теперь аккуратно вносим правки, чтобы превратить наш запрос в функцию:

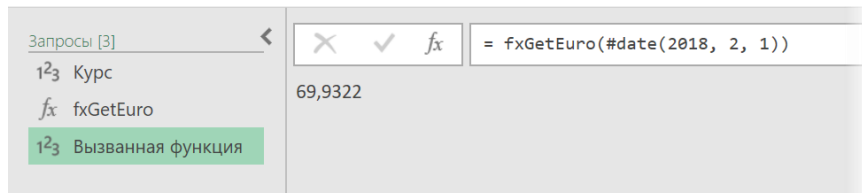


- В самом начале, перед оператором **let** добавляем строку с именем переменной **inputdate**, которая будет являться входным аргументом для нашей функции.
- Заменяем жёстко прописанную в запросе дату **01.02.2019** на нашу переменную **inputdate**, приклеивая её к адресу веб-страницы с помощью символа амперсанда **&**.
- Поскольку в Power Query нельзя склеивать данные разного типа (текст и дату), то необходимо преобразовать нашу входную дату в текст, что мы и делаем с помощью функции **Text.From()**.

Всё. Жмём на **ОК**, и если всё сделали правильно, то должны увидеть следующую картину:

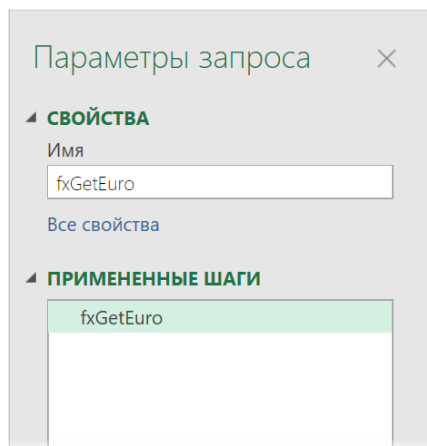


Давайте протестируем нашу функцию. Введём в поле **inputdate** какую-либо дату из прошлого, например 1 февраля 2018 года, т. е. год назад, и нажмем кнопку **Вызвать (Invoke)**:

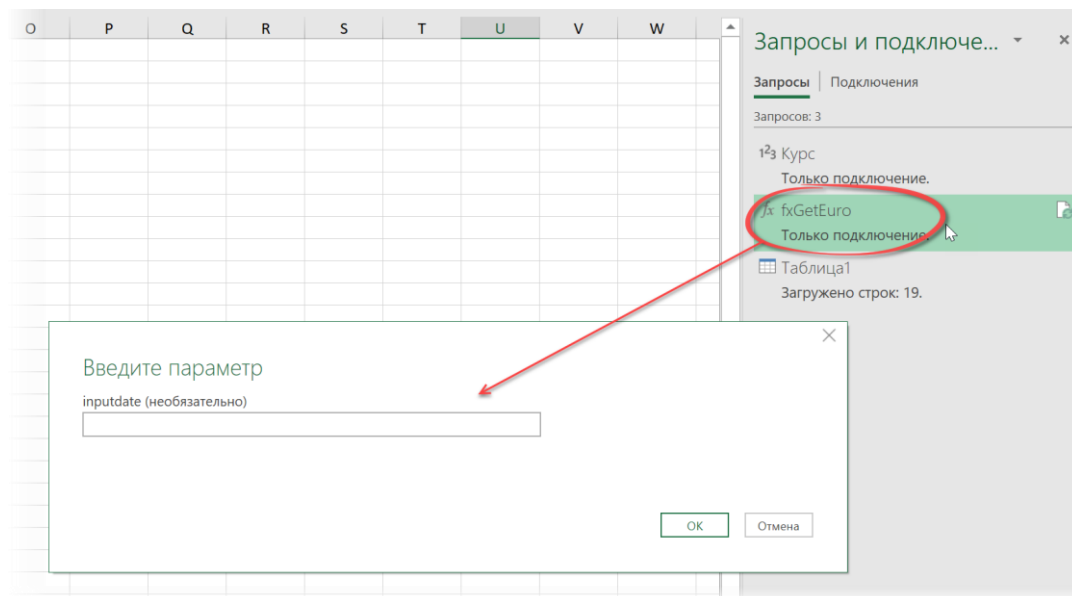


Сайт ЦБ выдаёт то же значение для этой даты, всё ОК.

Также обратите внимание, что после преобразования в функцию мы, к сожалению, уже не видим шаги запроса в правой панели редактора Power Query и уже не можем их здесь редактировать:



Двойной щелчок мышью по названию нашей функции в панели **Запросы и подключения** в самом Excel тоже не даст привычного эффекта: вместо входа в редактор запросов мы увидим окно ввода аргументов:



Чтобы отредактировать запрос, нужно щёлкнуть по нему правой кнопкой мыши и использовать команду **Изменить (Edit)**, а любые правки теперь возможны только через **Расширенный редактор** непосредственно в М-коде.

Поэтому имеет смысл до превращения запроса в функцию создать его копию, что мы и сделали, если помните, командой **Дублировать (Duplicate)** в самом начале второго этапа.

Этап 3. Применяем созданную функцию

Остался последний штрих. Загрузим в Power Query исходную таблицу с данными по продажам и воспользуемся кнопкой **Вызвать настраиваемую функцию** с вкладки **Добавление столбца (Add Column → Invoke Custom Function)**. В открывшемся окне выберем созданную функцию **fxGetEuro** и зададим её аргумент – дату из столбца **Дата продажи**:

The screenshot shows the Power Query editor interface. At the top, the formula bar contains the M-code: `= Table.AddColumn("#Измененный тип", "Курс евро", each fxGetEuro([Дата продажи]))`. Below the formula bar is a table with the following data:

	Товар	Дата продажи	Сумма, руб.	Курс евро
1	Помидор	13.07.2017	25200	69,4494
2	Абрикос	14.12.2015	86800	75,7472
3	Дыня	25.12.2015	2600	76,0441
4	Капуста	03.07.2018	47600	73,469
5	Банан	05.03.2017	54500	61,985
6	Лук-порей	17.07.2017	52000	68,3597
7	Устрицы	01.11.2018	73600	74.4189
8	Лук			
9	Фасоль			
10	Яблоки			
11	Редис			
12	Инжир			
13	Огурец			
14	Рис			
15	Тунец			
16	Киви			
17	Лимон			
18	Крапива			
19	Рожь			

Overlaid on the table is a dialog box titled "Вызвать настраиваемую функцию". It contains the following fields:

- Имя нового столбца: Курс евро
- Запрос функции: fxGetEuro
- inputdate (необязательно): Дата продажи

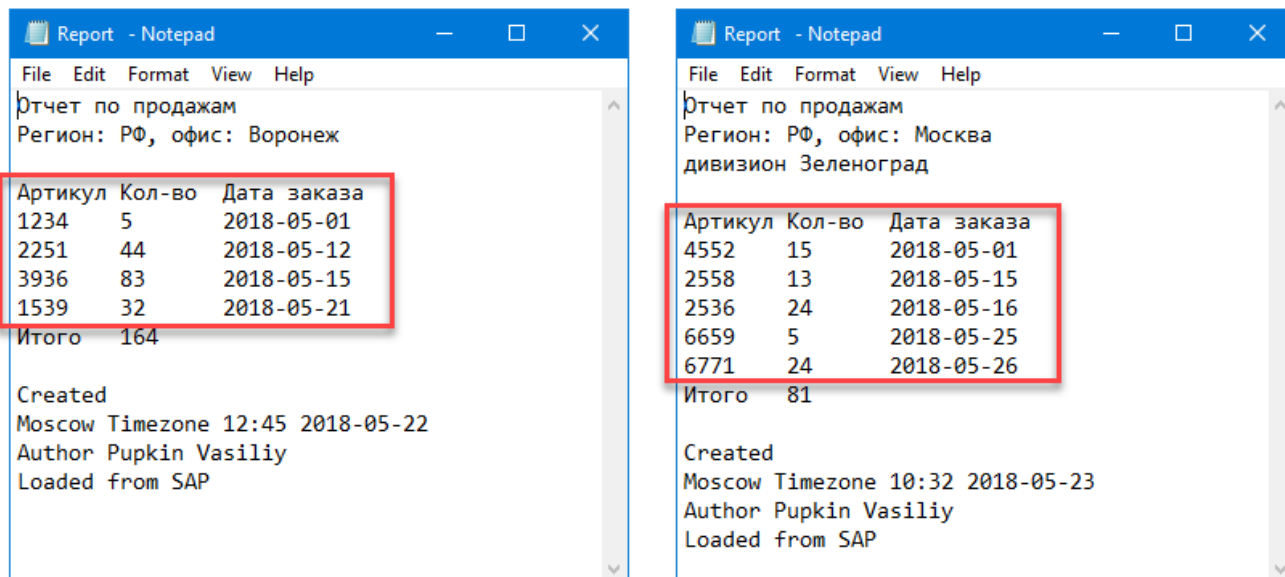
Buttons for "OK" and "Отмена" are visible at the bottom right of the dialog.

Также можно воспользоваться кнопкой **Настраиваемый столбец (Custom Column)** и ввести название нашей функции прямо в формулу вручную:

```
=fxGetEuro([Дата продажи])
```

Загрузка «плавающего» фрагмента данных

Иногда бывают ситуации, когда заранее неизвестно, сколько именно и каких строк нужно загрузить из исходных данных. Допустим, мы должны загрузить в Power Query данные из текстового файла, что, на первый взгляд, не представляет большой проблемы. Сложность в том, что файл регулярно обновляется, и завтра в нем может быть другое количество строк с данными, шапка из трех, а не двух строк, и т. д.:



То есть мы заранее не можем с определенностью сказать, начиная с какой строки и сколько именно строк нужно импортировать. А это проблема, т. к. эти параметры жестко прописываются в M-коде запроса. И если сделать запрос по первому файлу (импорт 5 строк начиная с 4-ой), то он уже не будет правильно работать со вторым.

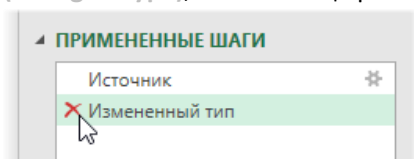
Было бы здорово, если бы наш запрос мог сам определять начало и конец «плавающего» текстового блока, который нам нужен. Реализовать это можно несколькими способами, в зависимости от внешнего вида исходных данных.

Способ 1. Маркеры начала и конца

В основе этого способа лежит идея, что в наших данных есть некие ключевые слова или значения, которые можно использовать как маркеры (признаки) начала и конца нужного нам блока данных. В нашем примере началом станет строка, начинающаяся со слова *Артикул*, а концом – строка со словом *Итого*. Такую проверку легко реализовать в Power Query с помощью условного столбца – аналога функции **ЕСЛИ** (IF) в Microsoft Excel.

Давайте посмотрим, как это сделать.

1. Сначала загрузим содержимое нашего текстового файла в Power Query стандартным способом – через команду **Данные → Получить данные → Из файла → Из текстового / CSV-файла** (Data → Get Data → From file → From text/CSV file).
2. После импорта можно (как всегда) убрать автоматически добавленный шаг **Измененный тип** (Changed Type), т. к. нам еще рано назначать типы данных для столбцов:



3. Теперь с помощью команды **Добавление столбца → Условный столбец** (Add Column → Conditional Column) добавим столбец с проверкой двух условий (на начало и конец блока) и выводом любых различных значений (например, чисел **1** и **2**) и **null**, если ни одно из условий не выполняется:

Добавление условного столбца

Добавьте условный столбец, который вычисляется из других столбцов или значений.

Имя нового столбца

Если	Имя столбца	Оператор	Значение	То	Вывод
	Column1	начинается с	ABC 123 Артикул	To	ABC 123 1
Иначе...	Column1	начинается с	ABC 123 Итого	To	ABC 123 2

В противном случае

После нажатия на **OK** получим такую картину:

	Column1	Column2	Column3	Пользовательская
1	Отчет по продажам			null
2	Регион: РФ, офис: Воронеж			null
3				null
4	Артикул	Кол-во	Дата заказа	1
5	1234	5	2018-05-01	null
6	2251	44	2018-05-12	null
7	3936	83	2018-05-15	null
8	1539	32	2018-05-21	null
9	Итого	164		2
10				null
11	Created			null
12	Moscow Timezone 12:45 2018-05-22			null
13	Author Pupkin Vasiliiy			null
14	Loaded from SAP			null

4. Теперь идем на вкладку **Преобразование** и выбираем команду **Заполнить → Вниз** (Transform → Fill → Down) – наши единички и двойки протянутся вниз по столбцу:

	Column1	Column2	Column3	Пользовательская
1	Отчет по продажам			null
2	Регион: РФ, офис: Воронеж			null
3				null
4	Артикул	Кол-во	Дата заказа	1
5	1234	5	2018-05-01	1
6	2251	44	2018-05-12	1
7	3936	83	2018-05-15	1
8	1539	32	2018-05-21	1
9	Итого	164		2
10				2
11	Created			2
12	Moscow Timezone 12:45 2018-05-22			2
13	Author Pupkin Vasiliiy			2
14	Loaded from SAP			2

5. Ну, а дальше, как легко догадаться, можно просто отфильтровать в условном столбце единицы – и вот наш желанный кусок данных:

	ABC Column1	ABC Column2	ABC Column3	ABC 123 Пользовательская
1	Артикул	Кол-во	Дата заказа	
2	1234	5	2018-05-01	1
3	2251	44	2018-05-12	1
4	3936	83	2018-05-15	1
5	1539	32	2018-05-21	1

6. Останется только поднять первую строку в шапку **командой Использовать первую строку в качестве заголовков** на вкладке **Главная** (Home → Use First Row as Headers) и удалить ненужный более условный столбец, щелкнув по его заголовку правой кнопкой мыши и выбрав команду **Удалить столбец** (Remove Column):

	123 Артикул	123 Кол-во	Дата заказа
1	1234	5	01.05.2018
2	2251	44	12.05.2018
3	3936	83	15.05.2018
4	1539	32	21.05.2018

Задача решена. При изменении данных в исходном текстовом файле запрос теперь будет самостоятельно определять начало и конец данных и импортировать каждый раз правильное количество строк. Конечно же, подобный подход работает и в случае импорта не одиночного, а сразу всех файлов из папки.

Способ 2. Вводим переменные

Но что делать, если признаки начала и конца в данных одинаковые? Например, в нашем файле нужный блок выделяется строками из нескольких дефисов с обеих сторон:

```

Report3 - Notepad
File Edit Format View Help
Россия, Москва

-----
4552    15    2018-05-01
2558    13    2018-05-15
2536    24    2018-05-16
6659     5    2018-05-25
6771    24    2018-05-26
-----

Created
Moscow Timezone 10:32 2018-05-23
Author Pupkin Vasiliy
Loaded from SAP|

```

Предыдущий простой подход здесь не сработает, т. к. начало и конец неотличимы друг от друга, и придется действовать по-другому.

Для начала, как и в предыдущем случае, загрузим содержимое всего файла в Power Query через команду **Данные → Получить данные → Из файла → Из текстового / CSV-файла** (Data → Get Data → From file → From text/CSV file) и удалим лишний шаг **Измененный тип** (Changed Type).

Затем добавим к нашим данным столбец с нумерацией строк на вкладке **Добавить столбец → Столбец индекса → От 1** (Add column → Index Column → From 1):

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3	1.2 Индекс
1	Россия, Москва			1
2				2
3	-----			3
4	4552	15	2018-05-01	4
5	2558	13	2018-05-15	5
6	2536	24	2018-05-16	6
7	6659	5	2018-05-25	7
8	6771	24	2018-05-26	8
9	-----			9
10				10
11	Created			11
12	Moscow Timezone 10:32 2018-05-23			12
13	Author Pupkin Vasily			13
14	Loaded from SAP			14

Теперь, чтобы проще было понять главную идею, отфильтруем строки, которые начинаются с дефисов:

Фильтрация строк ✕

Базовый Подробнее

Сохранять строки, в которых "Column1"

начинается с

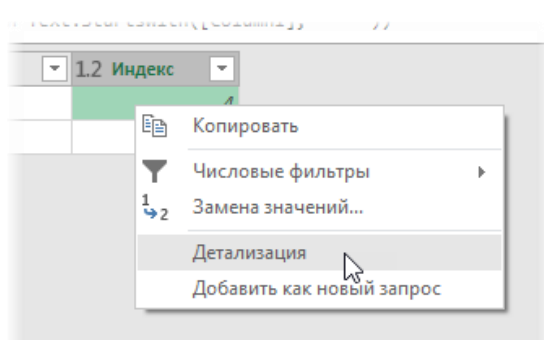
И Или

После применения фильтра на экране должны остаться наши строки-ограничители:

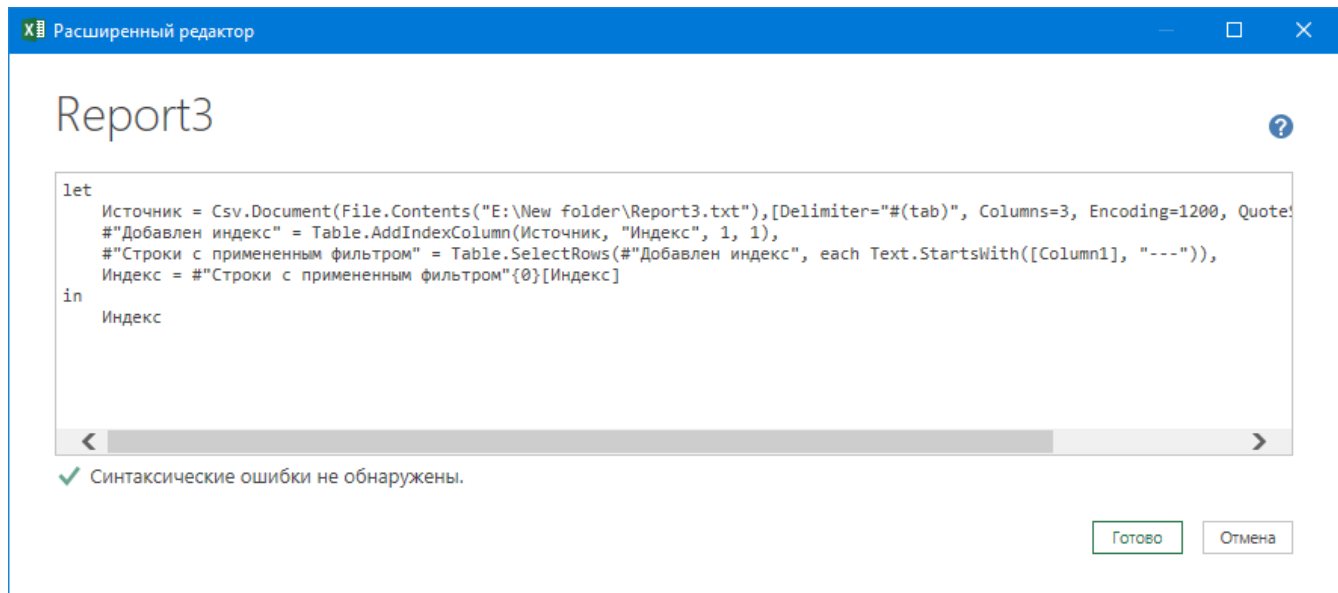
✕ ✓ fx = Table.SelectRows("#Добавлен индекс", each Text.StartsWith([Column1], "---"))

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3	1.2 Индекс
1	-----			3
2	-----			9

Теперь щелкнем правой кнопкой мыши по ячейке с числом 3 и выберем команду **Детализация** (Drill Down), чтобы «провалиться» в нее и дойти до её содержимого – номера стартовой строки:



Дальше – самое интересное: посмотрим исходный код запроса на языке M, открыв его на вкладке **Просмотр** кнопкой **Расширенный редактор** (View – Advanced Editor):



Давайте внимательно посмотрим на последних две строки перед оператором **in**.

Функция **Table.SelectRows** осуществляет фильтрацию таблицы, полученной с предыдущего шага с именем **"#Добавлен индекс"**:

```
#"Строки с примененным фильтром" = Table.SelectRows(#"Добавлен индекс", each
Text.StartsWith([Column1], "---")),
```

А следующая строка выводит содержимое первой ячейки из столбца [Индекс] из результатов фильтрации:

```
Индекс = #"Строки с примененным фильтром"{0}[Индекс]
```

Нумерация строк в Power Query идет с нуля, поэтому первая строка превращается в нулевую, что и видно в фигурных скобках.

Соединив эти две строки в одну, мы можем создать переменную (назвав её, допустим, **StartRow**), в которую будет записываться номер стартовой строки блока:

```
StartRow = Table.SelectRows(#"Добавлен индекс", each Text.StartsWith([Column1], "---
")){0}[Индекс]
```

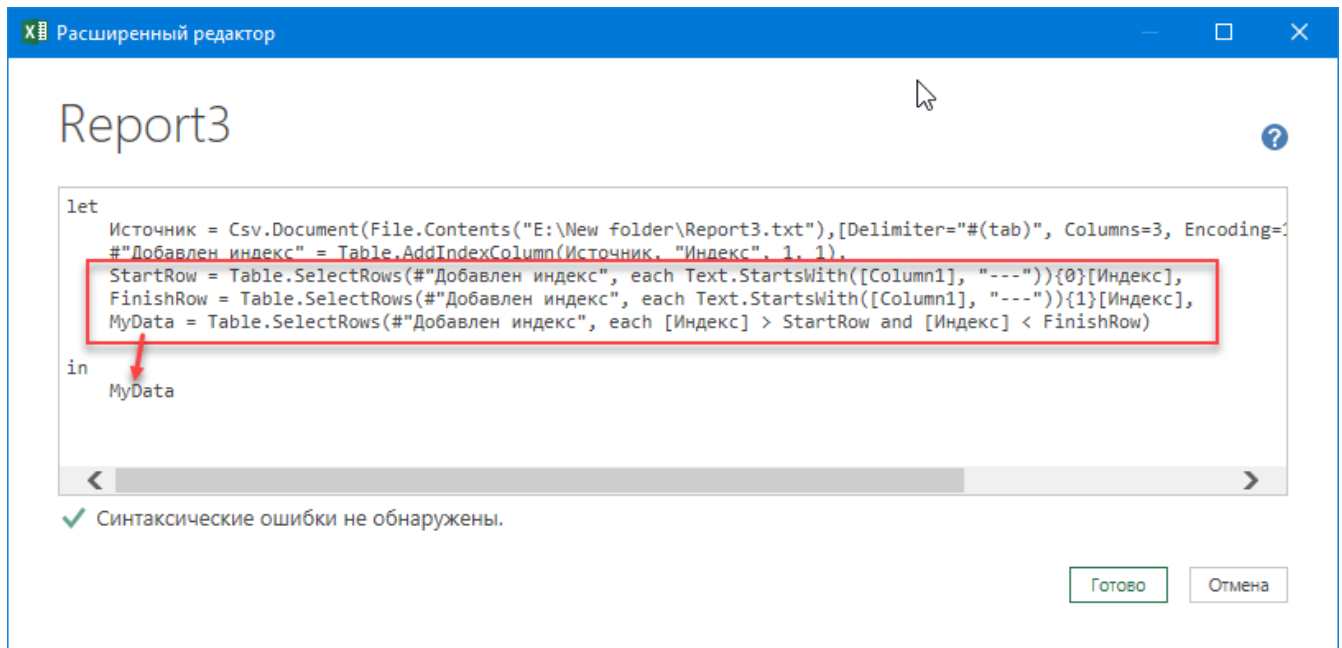
Подобным же образом можно создать и переменную для хранения номера второй строки с дефисами – у неё ноль в фигурных скобках меняется на 1:

```
FinishRow = Table.SelectRows(#"Добавлен индекс", each Text.StartsWith([Column1], "---
")){1}[Индекс]
```

Ну а теперь можно добавить и третью переменную (я назвал её **MyData**), которая и будет фильтровать строки по столбцу [Индекс] в интервале от **StartRow** до **FinishRow**:

```
MyData = Table.SelectRows(#"Добавлен индекс", each [Индекс] > StartRow and [Индекс] <
FinishRow)
```

Новые команды можно вписать в код вместо последних двух строк перед **in** (только без переносов строк, как в книге, здесь они просто не уместились на одной строке):



Не забудьте поставить запятые в конце всех строк, кроме последней перед **in**, и изменить название переменной в строке после **in**, чтобы содержимое **MyData** выводилось как результат работы запроса.

После нажатия на **Готово** мы увидим то, что требовалось: все строки в интервале от **StartRow** до **FinishRow**:

	A ^B _C Column1	A ^B _C Column2	A ^B _C Column3	1.2 Индекс
1	4552	15	2018-05-01	4
2	2558	13	2018-05-15	5
3	2536	24	2018-05-16	6
4	6659	5	2018-05-25	7
5	6771	24	2018-05-26	8

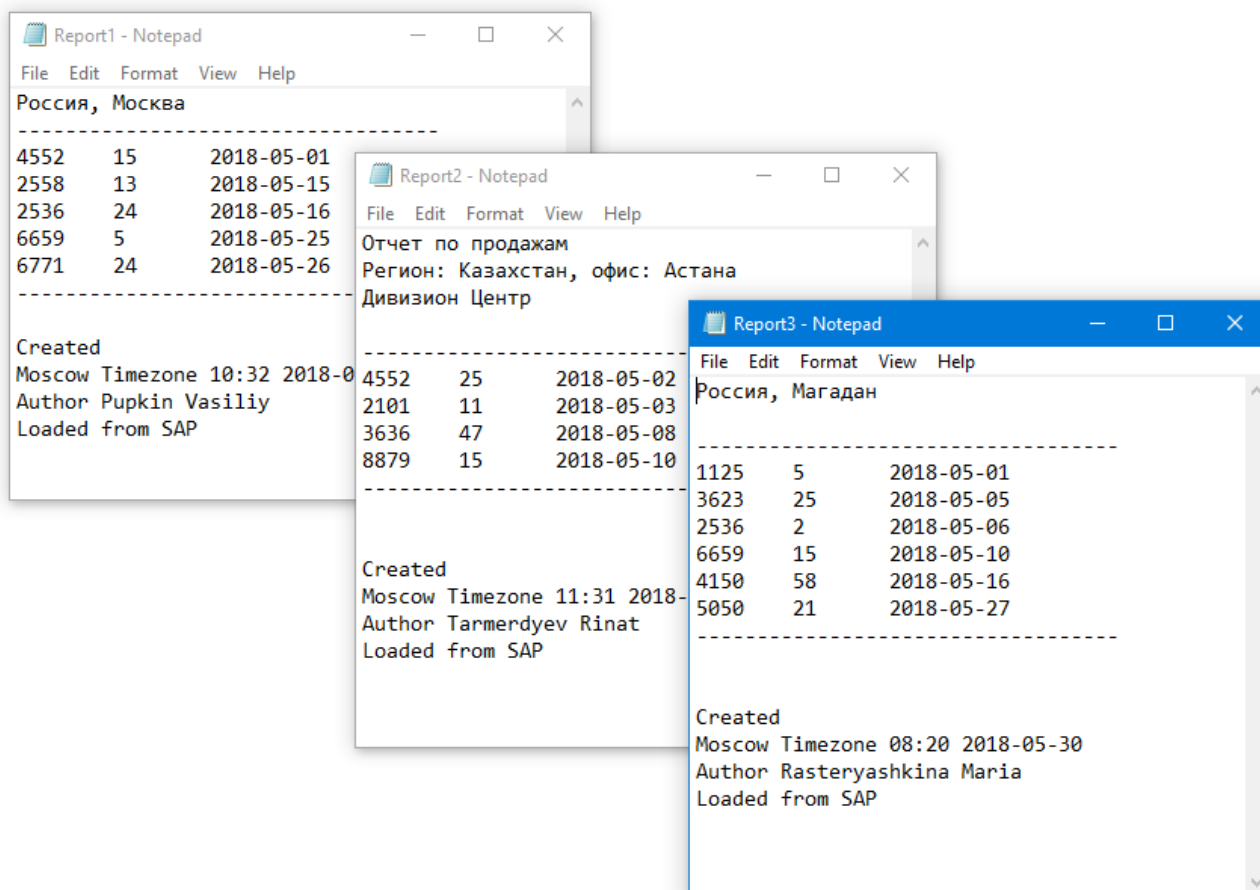
Останется только переименовать столбцы в шапке, назначить форматы данных и удалить ненужный теперь столбец **[Индекс]**:

	1 ² ₃ Артикул	1 ² ₃ Количество	Дата
1	4552	15	01.05.2018
2	2558	13	15.05.2018
3	2536	24	16.05.2018
4	6659	5	25.05.2018
5	6771	24	26.05.2018

Выборка фрагмента при массовой загрузке файлов

*Я всегда буду искать ленивого человека для работы, ведь он найдет много легких путей для решения поставленной задачи.
(Билл Гейтс)*

Чтобы довести разбор ситуации из предыдущей главы до логического финала, представим себе случай, когда текстовых файлов с дефисами-разделителями у нас несколько, и мы хотим собрать их все в одну таблицу. Причем в каждом файле наблюдается та же картина: «плавающий» блок данных может начинаться с разной строки, быть разного размера и не иметь четких различающихся признаков начала и конца данных:



В такой ситуации последовательность действий будет похожа на ту, что мы разбирали в главе [Преобразование запроса в функцию на примере веб-запроса курса валют](#). То есть мы:

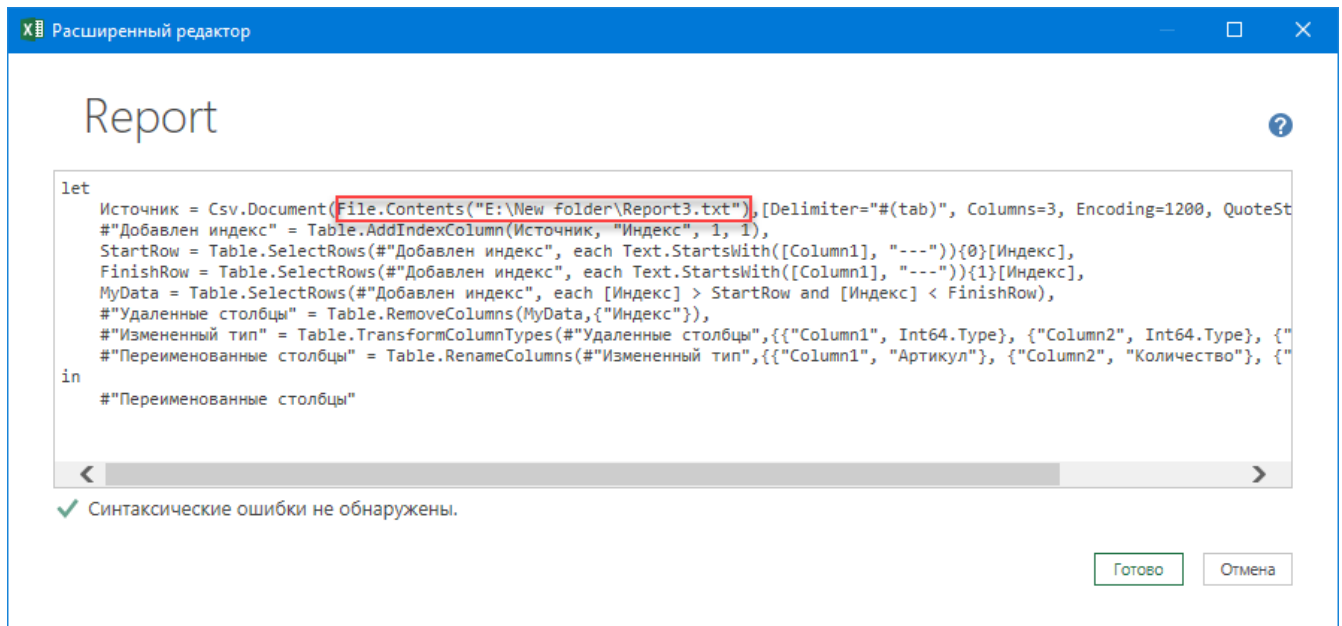
1. Сначала сделаем запрос для импорта нужного фрагмента из одного файла (как в предыдущей главе).
2. Преобразуем запрос в функцию.
3. Загрузим в Power Query все исходные текстовые файлы и применим созданную функцию к каждому из них.

Шаг 1. Одиночный запрос

Тут делаем все аналогично предыдущей главе. На выходе должны получить работающий запрос, загружающий «плавающий» фрагмент из одного файла.

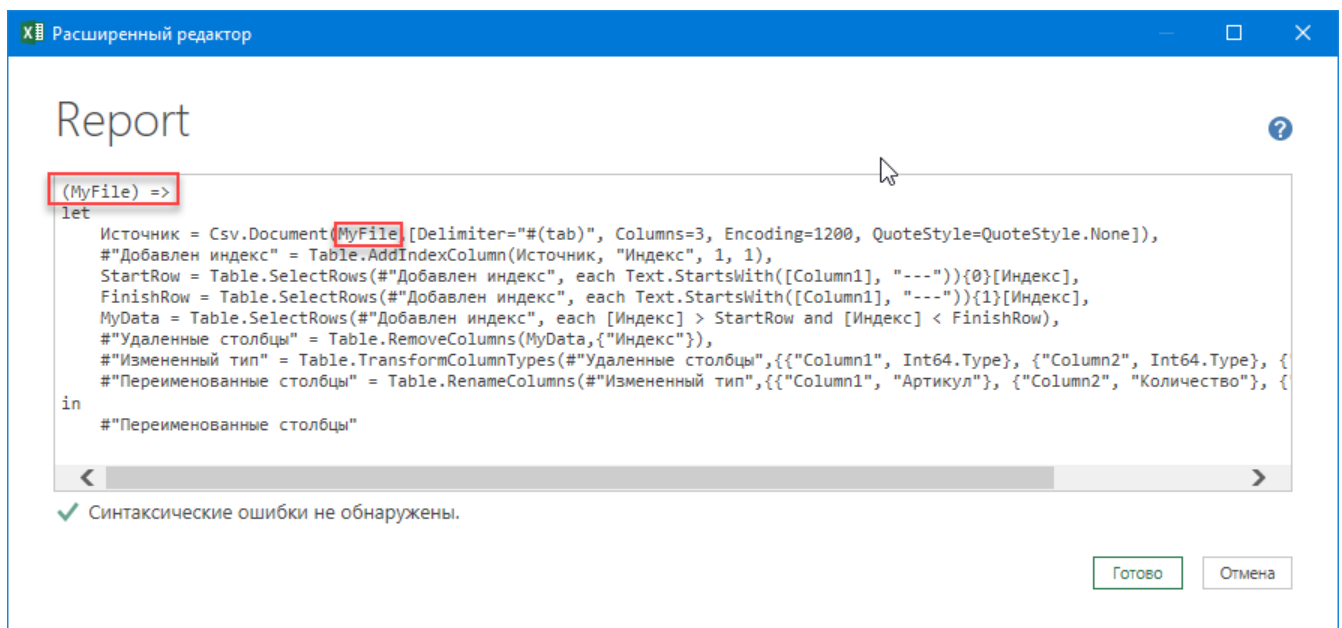
Шаг 2. Преобразуем запрос в функцию

Открываем **Расширенный редактор** на вкладке **Просмотр** (View → Advanced Editor) и видим исходный код нашего запроса:



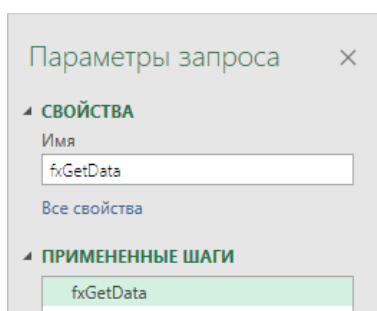
Вносим в код две небольшие правки, преобразующие его в функцию:

- Добавляем в самом начале кода пустую строку и вписываем туда имя переменной, которая будет аргументом (входными данными) для нашей функции. Назовём ее, допустим, **MyFile**.
- Заменяем в строке **Источник** (Source) функцию **File.Contents(...)** на имя нашей переменной.



После нажатия на кнопку **Готово** наш запрос будет преобразован в функцию и сможет выдергивать «плавающий» фрагмент из любого текстового файла, содержимое которого задается в качестве аргумента нашей функции.

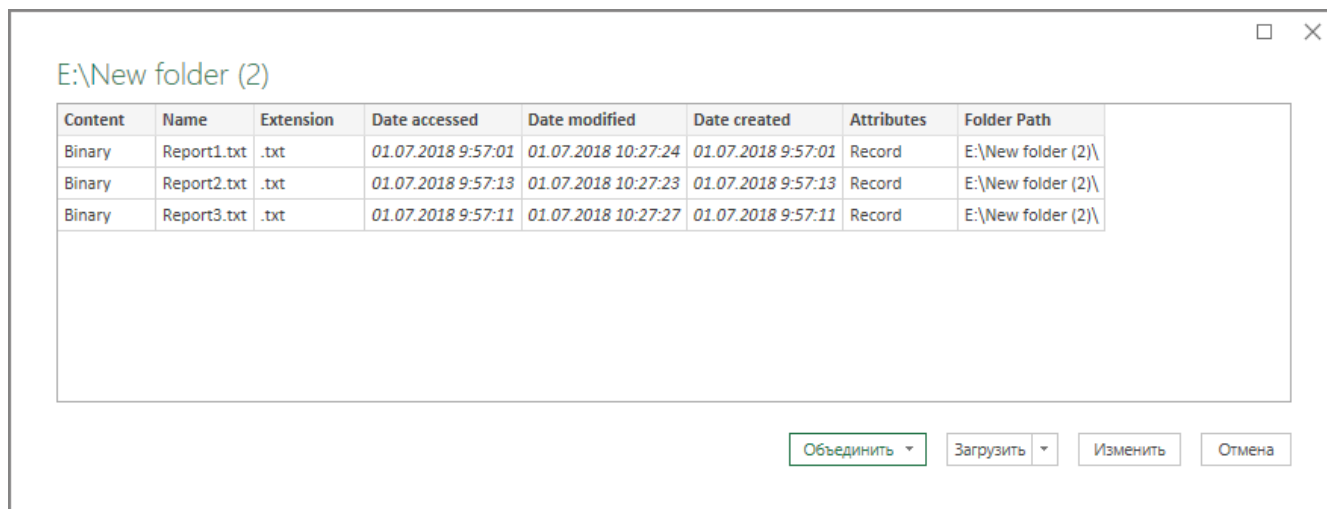
Для наглядности можно переименовать запрос (например, в *fxGetData*) в правой панели:



Шаг 3. Собираем файлы и применяем нашу функцию

Вернемся назад в Excel и создадим новый запрос, собирающий данные из всех файлов папки, как мы уже делали ранее. Используем на вкладке **Данные** команду **Получить данные → Из файла → Из папки** (Data → Get Data → From file → From folder).

После выбора папки в следующем окне увидим список все файлов и ждем **Изменить** (Edit):



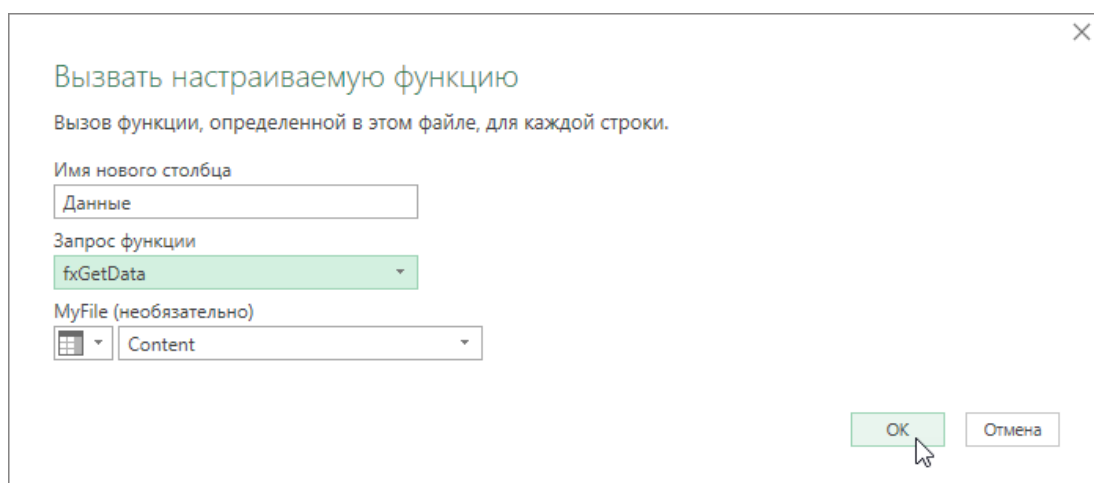
Далее тот же список файлов мы должны увидеть уже в Power Query:

	Content	Name	Extension	Date accessed	Date modified	Date created
1	Binary	Report1.txt	.txt	01.07.2018 9:57:01	01.07.2018 10:27:24	01.07.2018 9:57:01
2	Binary	Report2.txt	.txt	01.07.2018 9:57:13	01.07.2018 10:27:23	01.07.2018 9:57:13
3	Binary	Report3.txt	.txt	01.07.2018 9:57:11	01.07.2018 10:27:27	01.07.2018 9:57:11

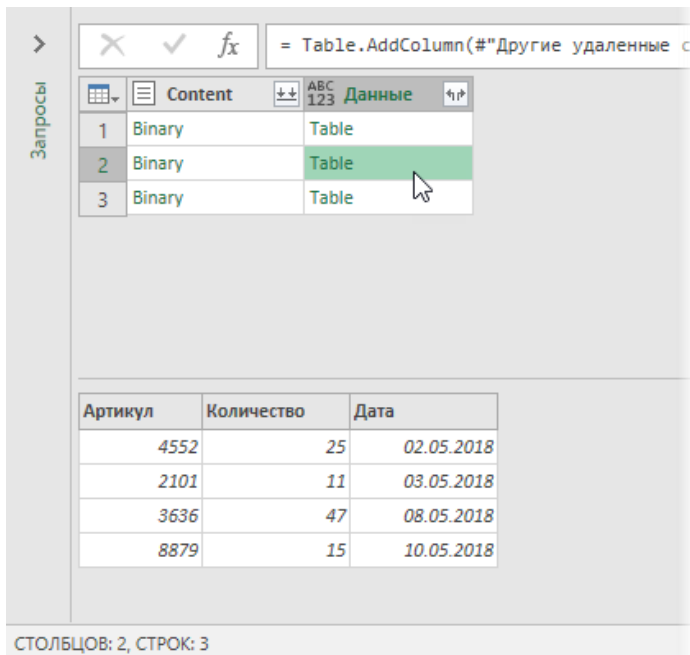
И тут начинается самое интересное.

Удалим все столбцы, кроме первого, где, собственно, и хранится содержимое каждого файла.

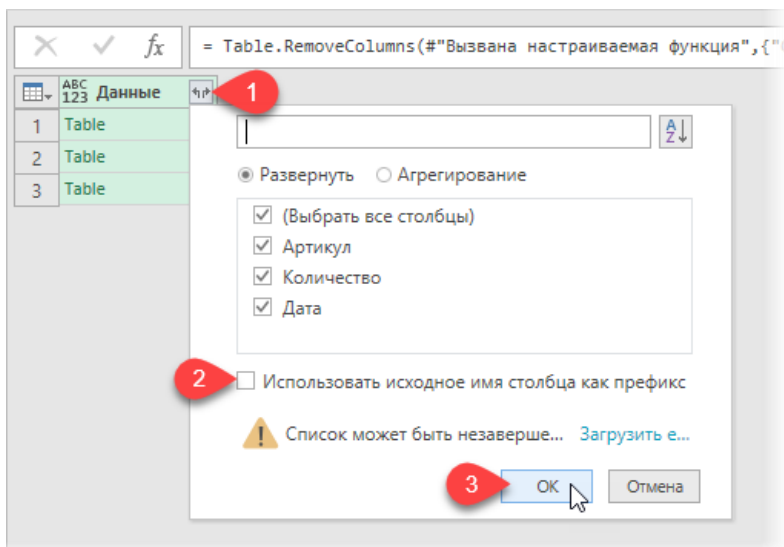
Затем применим к каждому файлу созданную нами функцию на вкладке **Добавление столбца → Вызвать настраиваемую функцию** (Add Column → Invoke Custom Function). В открывшемся окне выберем нашу функцию и столбец **[Content]** в качестве её аргумента:



После нажатия на **ОК** получим еще один столбец, в каждой ячейке которого лежат извлеченные функцией данные. Щелкнув левой кнопкой мыши в белый фон ячейки (не в слово **Table!**), мы как раз и увидим нужный нам «плавающий» фрагмент из каждого файла!



Осталось удалить ненужный более столбец **[Content]**, а колонку **Данные** развернуть кнопкой с двойными стрелками в правом верхнем углу:



Галочку **Использовать имя столбца как префикс** (Use original column name as prefix) можно снять, и после нажатия на **ОК** мы увидим желанную сборку всех «плавающих» текстовых фрагментов из всех файлов в одной таблице.

Танцы на граблях

Эта глава посвящена разбору типичных проблем и ошибок, возникающих при работе с Power Query в реальной жизни. В идеале наша задача не только создать работающий запрос, а продумать наперёд и подстраховаться в ситуациях, когда небольшие изменения в исходных данных или креативность других пользователей могут нарушить его работу.

В этой главе мы разберем:

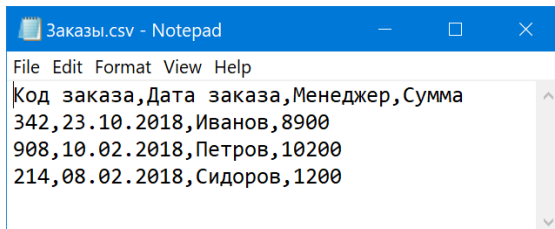
- как сделать **неубиваемый запрос**, если в исходных данных могут быть удалены или добавлены новые столбцы;
- как правильно **«причёсывать» шапку** загруженных данных;
- как **правильно выполнять переименование и удаление** столбцов, чтобы запрос потом не «умер»;
- в чем **опасность изменения порядка** столбцов в таблице;
- как подстраховаться от подводных камней при безобидной на первый взгляд **фильтрации** данных;
- какими техниками и трюками можно воспользоваться, чтобы **ускорить ваши запросы**, если они долго обновляются.



Добавленные или удалённые столбцы в данных

Внезапно добавленные или удалённые столбцы в исходных данных весьма частая проблема при работе в Power Query. Конечно, эта трудность не будет преследовать вас при импорте «умных» таблиц, т. к. они автоматически растягиваются и сжимаются при добавлении или удалении данных, но в случае загрузки информации из других источников, например из CSV или текстовых файлов, эти «грабли» явятся вам во всей красе.

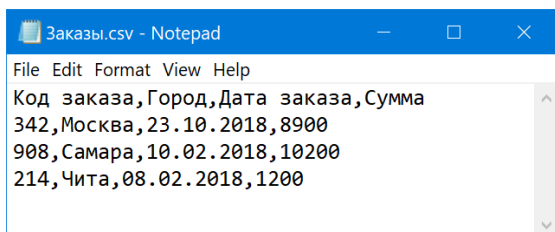
Представим, что нам нужно импортировать данные из файла **Заказы.csv** вот такого вида:



Это несложная задача, которую мы подробно разбирали в главе [Основные принципы работы в Power Query на примере загрузки TXT-файла](#). В результате мы должны получить вот такую таблицу:

	1 ² Код заказа	Дата заказа	A ^B C Менеджер	1 ² 3 Сумма
1	342	23.10.2018	Иванов	8900
2	908	10.02.2018	Петров	10200
3	214	08.02.2018	Сидоров	1200

Теперь представим, что в исходном файле спустя какое-то время меняется структура столбцов. Например, добавляется лишний столбец *Город* и исчезает столбец *Менеджер*. Плюс ко всему ещё и нарушается последовательность колонок:



Теперь при попытке обновить данные мы, конечно же, получим ошибку **[Expression.Error] Столбец «Менеджер» таблицы не найден**, и весь процесс обновления закончится, едва начавшись. Что же делать?

Чтобы наш запрос не сбоил каждый раз из-за лишней или недостающей колонки, имеет смысл слегка усложнить его, но обеспечить наличие всех необходимых столбцов в результате. Лучше всего будет сделать это по следующему алгоритму.

1. Загрузим в Power Query пустую «идеальную» таблицу без данных, но со всеми необходимыми столбцами и настроенными типами данных.
2. Добавим к «идеальной» таблице данные из файла заказов, как делали в главе [Добавление \(Append\)](#).
3. Удалим лишние столбцы, если они появились.

Думаю, вы ухватили суть. Давайте проделаем всё это по шагам.

Шаг 1. Идеальная таблица

Создадим на любом листе нашего текущего файла Excel прообраз «идеальной» таблицы именно с теми столбцами, которые нам нужны на выходе:

	A	B	C	D
1	Код заказа	Дата заказа	Менеджер	Сумма
2				
3				

Загрузим эту таблицу в Power Query как отдельный запрос с именем, например *ИдеальнаяТаблица*.

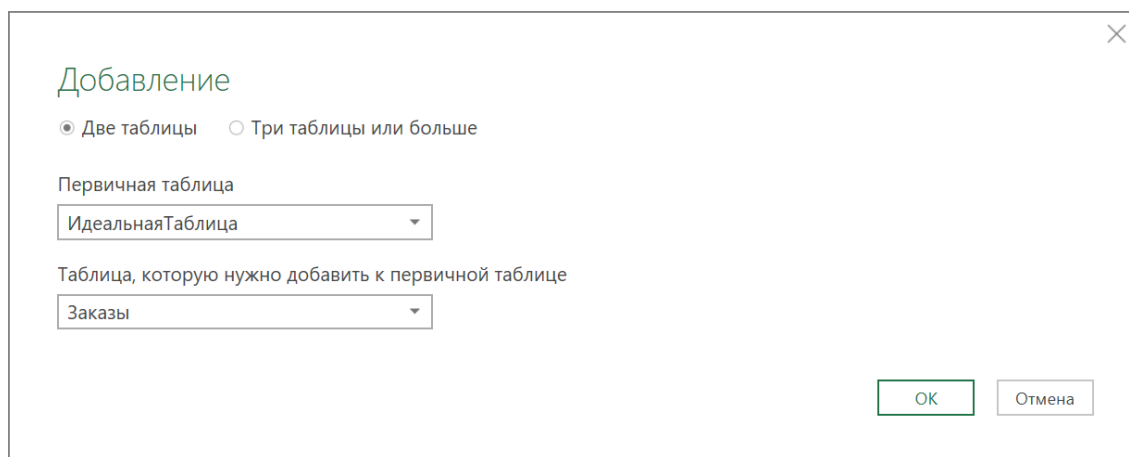
Microsoft Excel не позволяет сделать на листе «умную» таблицу, состоящую только из одной шапки, поэтому в наших данных при таком подходе всегда будет присутствовать пустая строка, состоящая из *null*. Можно избавиться от неё с помощью фильтра или команды **Удалить строки** → **Удаление верхних строк** (Remove rows → Remove top rows) на вкладке **Главная** (Home). В итоге должно получиться примерно так:



Наша «идеальная таблица» готова.

Шаг 2. Добавляем запросы

Удалим из первоначального запроса **Заказы** к CSV-файлу все шаги, кроме **Источника** (Source) и **Повышенные заголовки** (Promoted Headers), и добавим обе таблицы друг к другу, выбрав в Excel на вкладке **Данные** → **Получить данные** → **Объединение запросов** → **Добавить** (Data → Get Data → Combine queries → Append):



После нажатия на **OK** на экране должна появиться объединенная таблица:

	Код заказа	Дата заказа	Менеджер	Сумма	Город
1	342	23.10.2018	null	8900	Москва
2	908	10.02.2018	null	10200	Самара
3	214	08.02.2018	null	1200	Чита

Обратите внимание, что в ней присутствуют все столбцы из обеих таблиц, но столбец **Менеджер** пустой, т. к. он теперь отсутствует в исходных данных.

Шаг 3. Убираем лишнее

Теперь давайте избавимся от ненужных столбцов, оставив только те, что требуются. Для этого выделим (удерживая **Ctrl**) те четыре колонки, которые нам нужны (*Код заказа*, *Дата заказа*, *Менеджер* и *Сумма*), и выберем **Главная** → **Удалить столбцы** → **Удалить другие столбцы** (Home → Remove Columns → Remove Other Columns).

Останется настроить форматы данных для каждой колонки и насладиться результатом:

	Код заказа	Дата заказа	Менеджер	Сумма
1	342	23.10.2018	null	8900
2	908	10.02.2018	null	10200
3	214	08.02.2018	null	1200

Такой запрос будет гораздо надёжнее и уже не будет вылетать с ошибкой при любых изменениях структуры столбцов в исходном CSV-файле.

Мусор в названиях столбцов

Предположим, мы загрузили в Power Query данные из какого-либо источника (например, из «умной» таблицы):

	A	B	C
1	Товар	Менеджер	Сумма
2	Пончо	Лев	6900
3	Реглан	Марина	100
4	Поло	Яна	3700
5	Пуловер	Ульяна	7300
6			

Следующим шагом, скорее всего, будет сделанный вручную или автоматически добавленный шаг **Измененный тип (Changed Type)**, назначающий каждому столбцу соответствующий тип данных:

The screenshot shows the Power Query Advanced Editor. The formula bar contains the following M code:

```
= Table.TransformColumnTypes(Источник,{{"Товар", type text}, {"Менеджер", type text}, {"Сумма", Int64.Type}})
```

The 'Parameters' pane on the right shows the source as 'Таблица1' and the applied step as 'Измененный тип'.

Далее могут следовать ещё какие-то шаги по трансформации и «причёсыванию» таблицы, фильтрации, сортировке и т. д. Общий код нашего запроса в **Расширенном редакторе** на вкладке **Просмотр (View → Advanced Editor)** должен выглядеть примерно так:

```
let
    Источник = Excel.CurrentWorkbook()[Name="Таблица1"][Content],
    #"Измененный тип" = Table.TransformColumnTypes(Источник,{{"Товар", type text}, {"Менеджер", type text}, {"Сумма", Int64.Type}})
in
    #"Измененный тип"
```

Теперь представим себе следующую картину: другой пользователь, работающий с нашей исходной «умной» таблицей, допускает небрежность при создании отчёта и добавляет лишний пробел к названию какого-нибудь столбца или пишет его заголовок с маленькой буквы:

	A	B	C
1	Товар	менеджер	Сумма
2	Пончо	Лев	6900
3	Реглан	Марина	100
4	Поло	Яна	3700
5	Пуловер	Ульяна	7300
6			

С маленькой буквы

Невидимый лишний пробел в конце

Разумеется, наш запрос остановится с ошибкой **[Expression.Error] Столбец "Менеджер" таблицы не найден!** на том самом шаге **Изменённый тип (Changed Type)**, т. к. имена столбцов в нашей таблице уже не те, что прежде. Конечно, можно открыть исходный файл и вручную подправить «шапку», но где гарантии, что эта проблема не возникнет в будущем снова?

Исправить ситуацию в корне раз и навсегда лучше с помощью дополнительной строчки M-кода с очень полезной для таких случаев функцией **Table.Transform.ColumnNames** со следующим синтаксисом:

```
=Table.TransformColumnNames(таблица, функция)
```

где первый аргумент – это таблица (или имя шага, откуда мы её берём), а второй – это функция, которую мы хотим применить к названию каждого столбца.

Так, например, если мы хотим избавиться от лишних пробелов, то можно применить функцию **Text.Trim** (аналог экселевской **СЖПРОБЕЛЫ**), о которой мы уже упоминали в главе [Удаление лишних пробелов и SuperTrim](#):

```
=Table.TransformColumnNames(Источник, each Text.Trim(_))
```

Знак подчёркивания здесь играет роль «аргумента по умолчанию», т. е. того самого очередного заголовка столбца, который мы обрабатываем нашей функцией.

Если нам нужно исправить регистр всех заголовков, то помогут функции **Text.Upper** или **Text.Lower**, преобразующие весь текст в верхний или нижний регистр соответственно. Чтобы каждое слово в заголовке было с прописной буквы, а потом шли все строчные можно использовать функцию **Text.Proper**:

```
=Table.TransformColumnNames(Источник, each Text.Proper(_))
```

Если в заголовках появились какие-то лишние символы, то их в общем случае можно удалить функцией замены:

```
=Table.TransformColumnNames(Источник, each Text.Replace(_, "руб", ""))
```

Естественно, можно вкладывать эти функции одна в другую для компактности.

Строчку с этими функциями нужно будет вручную добавить в наш код запроса до шага **Измененный тип (Changed Type)**, не забыв подправить в нём имя предыдущего шага, чтобы данные передавались от одной переменной (строки) к другой по цепочке:

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Таблица1"]}[Content],
    ИсправитьЗаголовки = Table.TransformColumnNames(Источник, each Text.Proper(Text.Trim(_))),
    #"Измененный тип" = Table.TransformColumnTypes(ИсправитьЗаголовки,{{"Товар", type text},
        {"Менеджер", type text}, {"Сумма", Int64.Type}})
in
    #"Измененный тип"
```

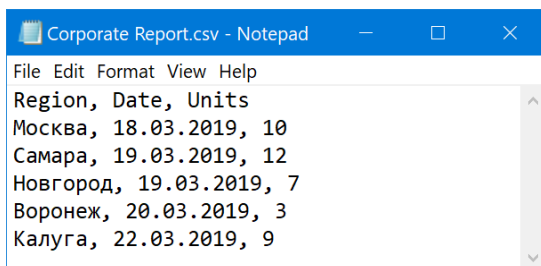
Переименование столбцов

— У этого оружия есть имя?
 — Гром-Секира.
 — Как-то топорно.
 («Мстители: Война бесконечности»)

Дать столбцам в таблице данных после импорта удобные и наглядные имена – естественное и правильное желание. Однако, делая это, мы очень повышаем вероятность того, что в будущем наш запрос может дать сбой на этом шаге.

Рассмотрим простой пример.

Допустим, мы каждый день выгружаем из корпоративной ERP-системы данные в виде CSV-файла вот такого вида (файл **Corporate Report.csv** в папке с примерами к этой книге):



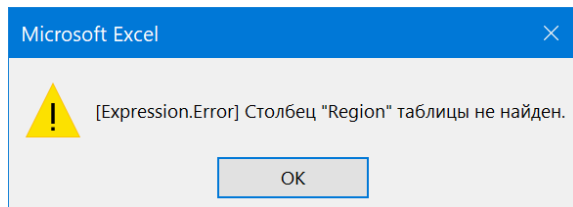
Этот файл мы импортируем в Power Query обычным образом через **Данные → Получить данные → Из файла → Из TXT/CSV-файла** (Data → Get Data → From file → From TXT/CSV-file), чтобы потом использовать далее:

	City	Date	Units
1	Москва	18.03.2019	10
2	Самара	19.03.2019	12
3	Новгород	19.03.2019	7
4	Воронеж	20.03.2019	3
5	Калуга	22.03.2019	9

Допустим, что для удобства сразу после импорта и повышения заголовков мы добавляем 3-й шаг и переименовываем столбцы из английских названий в более понятные русские:

Вот тут и кроются «грабли».

Если внимательно посмотреть в строку формул на аргументы функции **Table.RenameColumns**, которая выполняет переименование, то можно увидеть, что в ней жёстко прописаны старые названия столбцов: *Region*, *Date*, *Units*. Если завтра формат выгрузки из ERP-системы слегка поменяется и вместо слова *Region* первый столбец будет называться, допустим, *City*, то на шаге переименования наш запрос вылетит с ошибкой:



Как же обойти проблему?

Если предположить, что в будущем могут меняться названия столбцов, но не их порядок, т. е. город всегда будет первой колонкой, дата – второй и т. д., то лучше переименовать столбцы, привязываясь не к их старым названиям, а к их положению в таблице.

Получить список всех текущих заголовков таблицы можно с помощью M-функции:

```
=Table.ColumnNames("#Повышенные заголовки")
```

где **#"Повышенные заголовки"** – это имя предыдущего шага, с которого мы берём таблицу.

Если нам нужны не все, а какой-то отдельный заголовок, то мы добавляем к этой функции порядковый номер требуемого элемента (начиная с нуля). Так, например, чтобы получить заголовок третьего столбца (*Units*), нам потребуется выражение:

```
= Table.ColumnNames("#Повышенные заголовки"){2}
```

Чтобы наш запрос стал универсальным и работал при любых входных данных, нужно в исходном коде шага переименования заменить жёстко прописанные имена столбцов:

```
= Table.RenameColumns("#Повышенные заголовки",
  {"Region", "Город"},
  {"Date", "Дата"},
  {"Units", "Количество"})
```

на соответствующие им функции:

```
= Table.RenameColumns("#Повышенные заголовки",
  {{Table.ColumnNames("#Повышенные заголовки"){0}, "Город"},
  {Table.ColumnNames("#Повышенные заголовки"){1}, "Дата"},
  {Table.ColumnNames("#Повышенные заголовки"){2}, "Количество"}})
```

Сделать это можно прямо в строке формул либо в исходном коде запроса, открыв его в **Расширенном редакторе** на вкладке **Просмотр** (View → Advanced Editor).

Удаление несуществующих столбцов

*Я просто беру глыбу мрамора и отсекаю от неё всё лишнее.
(Огюст Роден)*

Как и в случае с переименованием в предыдущем пункте, при удалении ненужных столбцов мы всегда повышаем вероятность сбоя при последующем обновлении запроса. Представьте, что вы загрузили из внешнего источника таблицу и удалили в ней пару ненужных столбцов:

	Страна	Город	Дата	Время	Стоимость
1	Россия	Москва	01.06.2010 10:44:14	10:44:14	30100
2	Казахстан	Астана	28.10.2017 16:06:50	16:06:50	36600
3	Россия	Москва	17.05.2013 22:49:25	22:49:25	9800
4	Казахстан	Астана	02.01.2017 10:04:34	10:04:34	6500
5	Россия	Москва	27.10.2014 19:08:08	19:08:08	86100
6	Россия	Москва	15.05.2013 8:45:13	8:45:13	81800
7	Казахстан	Астана	17.03.2016 18:04:21	18:04:21	60200

Если спустя какое-то время один из этих же столбцов будет убран из самого источника данных, то мы получим ошибку на шаге удаления.

Способов обхода этой проблемы можно придумать несколько.

Способ 1. Сохранять вместо удаления

Эта идея хоть и выглядит, наверное, очевидной для многих, но повторить её в любом случае стоит, т. к. при всей внешней простоте этот нюанс очень важен.

Всегда по возможности старайтесь удалять ненужные столбцы «от противного», т. е. вместо того, чтобы:

1. Выделить ненужные столбцы (*Страна* и *Время*).
2. Удалить их командой **Удалить столбцы** (Home → Remove Columns) на вкладке **Главная** (Home) или в контекстном меню.

...делать «наоборот»:

1. Выделить столбцы, которые нужно сохранить (*Город*, *Дата* и *Стоимость*).
2. Удалить остальные командой **Удалить другие столбцы** (Remove Other Columns).

В этом случае в M-коде вместо функции:

```
= Table.RemoveColumns(Источник, {"Страна", "Время"})
```

где жёстко прописываются имена удаляемых столбцов, будет записана функция:

```
= Table.SelectColumns(Источник, {"Город", "Дата", "Стоимость"})
```

где имена удаляемых столбцов никак не участвуют, и наш запрос не будет внезапно сбиться из-за отсутствия одного из них завтра.

Способ 2. Удалять по номеру, а не по имени столбца

Эта методика один в один повторяет идею из предыдущей главы. Если положение столбцов в таблице статично, то удалять их по их положению (порядковому номеру) без привязки к конкретному имени столбца во многих случаях будет более надёжным подходом.

Как и ранее, для ссылки на нужный столбец можно использовать функцию

```
= Table.ColumnNames(Таблица){Номер_столбца}
```

где **Таблица** – это исходная таблица (или имя шага, с которого мы её берём), а в фигурных скобках указывается порядковый номер столбца (считая с нуля!).

Тогда команда для удаления, например, пятой колонки выглядела бы как:

```
= Table.RemoveColumns(Источник, Table.ColumnNames(Источник){4})
```

Для удаления нескольких столбцов подряд можно использовать функцию List.Range, которая извлекает из списка заголовков сразу несколько элементов. Например, для удаления столбцов с третьего по седьмой можно использовать команду:

```
= Table.RemoveColumns(Источник, List.Range(Table.ColumnNames(Источник),2,5))
```

А для удаления несмежных столбцов можно просто перечислить их в фигурных скобках, как список, через запятую:

```
= Table.RemoveColumns(Источник, {Table.ColumnNames(Источник){0},  
Table.ColumnNames(Источник){3},  
Table.ColumnNames(Источник){5}})
```

Способ 3. Удалять по признаку

Для некоторых сценариев этот способ может оказаться предпочтительнее, чем предыдущие. Если в заголовках у всех удаляемых столбцов есть некий общий признак, то им можно воспользоваться для определения того, удалять или оставлять соответствующий столбец в данных.

Допустим, у нас есть вот такая загруженная в Power Query таблица:

	ABC 123	Товар	ABC 123	Январь План	ABC 123	Январь Факт	ABC 123	Февраль План	ABC 123	Февраль Факт	ABC 123	Март План	ABC 123	Март Факт
1		Орехи кешью		83693		95950		77802		45130		98242		92360
2		Авокадо		93356		9039		99801		32833		61537		60770
3		Свекла		778		86369		56519		34051		31260		68970
4		Лук-порей		11563		85351		53067		1429		45569		14210

Предположим, мы хотим удалить те столбцы, где в заголовке упоминается слово *План*. Нам поможет M-функция **List.Select**, которая может извлечь из списка, возвращаемого **Table.ColumnNames**, только те имена столбцов, которые удовлетворяют заданному условию. Условием будет наличие в тексте слова *План*, что мы и проверим функцией **Text.Contains**:

```
=Table.RemoveColumns(Источник,  
List.Select(Table.ColumnNames(Источник), each Text.Contains(_, "План")))
```

Конечно же, все вышеперечисленные трюки в M-коде можно проделывать, заменяя в выражениях функцию **Table.RemoveColumns**, на **Table.SelectColumns**, т. е. не удаляя ненужные, а сохраняя только требуемые столбцы.

Изменение порядка столбцов

Представим, что вы загрузили в Power Query таблицу и в процессе её обработки решили изменить последовательность столбцов. Переупорядочить столбцы очень легко, достаточно просто перетащить их заголовки мышью:

	ABC 123 Код заказа	ABC 123 Дата	ABC 123 Товар	ABC 123 Менеджер	ABC 123 Адрес доставки	ABC 123 Цена
1	BTI	30.04.2017 0:00:00	Туника	Надежда	Балаково, Солнечный пер., д.22	49290
2	NB7	08.05.2017 0:00:00	Комбинезон	Андрей	Северодвинск, Социалистическая ул., д.30	42880
3	ZL6	25.01.2017 0:00:00	Лосины	Алиса	Калининград, Фрунзе ул., д.85	99530
4	H4J	05.01.2017 0:00:00	Жилет	Антон	Люберцы, Чапаева ул., д.18	11160
5	DPA	28.07.2017 0:00:00	Пуховик	Вадим	Благовещенск, Новая ул., д.77	50330

Однако последствия этой невинной операции для надёжности нашего запроса будут весьма печальными. Если посмотреть в строку формул, то можно увидеть, что в получившемся М-выражении участвуют имена всех столбцов таблицы (хотя мы передвинули только один столбец с ценой):

	ABC 123 Код заказа	ABC 123 Дата	ABC 123 Цена	ABC 123 Товар	ABC 123 Менеджер	ABC 123 Адрес доставки
1	BTI	30.04.2017 0:00:00	49290	Туника	Надежда	Балаково, Солнечный пер., д.22
2	NB7	08.05.2017 0:00:00	42880	Комбинезон	Андрей	Северодвинск, Социалистическая ул., д.30
3	ZL6	25.01.2017 0:00:00	99530	Лосины	Алиса	Калининград, Фрунзе ул., д.85
4	H4J	05.01.2017 0:00:00	11160	Жилет	Антон	Люберцы, Чапаева ул., д.18
5	DPA	28.07.2017 0:00:00	50330	Пуховик	Вадим	Благовещенск, Новая ул., д.77

Строка формул: = Table.ReorderColumns(Источник,{ "Код заказа", "Дата", "Цена", "Товар", "Менеджер", "Адрес доставки" })

А это значит, что если в будущем хоть одна из этих колонок будет отсутствовать в исходных данных, то наш запрос вылетит с ошибкой на шаге переупорядочивания.

Что же делать?

Ну, для начала попытаться в принципе обойтись без перемещения столбцов. На практике далеко не всегда оно необходимо и часто делается «просто для красоты». Красота здесь очень дорого обойдётся, так что лучше без неё.

Если же перестановки не избежать, то надо постараться сделать её так, чтобы в коде упоминались только имена тех столбцов, которые мы перемещаем. Чтобы это реализовать, придётся зайти немного издалека.

Как мы уже упоминали ранее, получить весь набор заголовков столбцов в виде списка (list) можно с помощью функции **Table.ColumnNames**. Переместить в этом списке имя нашего столбца (*Цена*) в новую позицию можно в два действия.

Сначала нам нужно удалить из этого списка имя перемещаемого столбца, что можно сделать с помощью выражения:

```
RemovedPrice = List.RemoveItems(Table.ColumnNames(Источник), "Цена")
```

Затем нужно вставить имя удалённого столбца обратно, но уже в новой позиции. Это делает функция **List.InsertRange**:

```
NewNames = List.InsertRange(RemovedPrice, 2, "Цена")
```

Здесь первый аргумент – это наш список, второй – позиция вставки (начиная с нуля), третий – вставляемый текст.

И, наконец, полученный список с изменённой последовательностью столбцов нужно подставить в стандартную функцию переупорядочивания столбцов:

```
=Table.ReorderColumns(Источник, NewNames)
```

В результате код нашего запроса должен выглядеть примерно так:

```
let
```

```
    Источник = Excel.CurrentWorkbook()[Name="ИсходныеДанные"][Content],  
    RemovedPrice = List.RemoveItems(Table.ColumnNames(Источник), {"Цена"}),  
    NewNames = List.InsertRange(RemovedPrice, 2, {"Цена"}),  
    #"Переупорядоченные столбцы" = Table.ReorderColumns(Источник, NewNames)
```

```
in
```

```
    #"Переупорядоченные столбцы"
```


Опасный фильтр

Как ни странно, но знакомая и безобидная, на первый взгляд, фильтрация таблиц в Power Query тоже скрывает пару неприятных сюрпризов, к которым надо быть готовым. Одного из них мы уже касались в главе [Опасная иллюзия с фильтрацией через поле «Поиск»](#), теперь давайте рассмотрим другой.

Представим, что мы работаем вот с такой таблицей данных по продажам одежды:

	АВс Товар	АВс Размер	123 Цена
1	Жилет	XL	6470
2	Пиджак	L	15140
3	Платье	S	16860
4	Кофта	M	2340
5	Шорты	XL	2370
6	Платье	M	9710
7	Пончо	XS	4780
8	Свитер	S	5210
9	Плащ	L	11390

Допустим, что нам нужно отфильтровать только товары размеров "S" и "XS", что, понятное дело, запросто можно сделать с помощью выпадающего списка и проставления соответствующих галочек:

Если после фильтрации посмотреть в строку формул, то мы увидим там M-функцию Table.SelectRows с вполне ожидаемыми условиями о том, что размер должен быть равен "S" или "XS":

	АВс Товар	АВс Размер	123 Цена
1	Платье	S	16860
2	Пончо	XS	4780
3	Свитер	S	5210
4	Майка	XS	860

Пока всё логично и прозрачно, да?

Теперь предположим, что мы передумали и хотим отфильтровать не только размеры "S" и "XS", но ещё и "M". Удалим последний шаг и выполним фильтрацию заново, отметив нужные размеры тремя соответствующими

галочками. После нажатия на **ОК** мы увидим нужные строки, но выражение в строке формул будет уже принципиально другим:

	Товар	Размер	Цена
1	Платье	S	16860
2	Кофта	M	2340
3	Платье	M	9710
4	Пончо	XS	4780
5	Свитер	S	5210

То есть сейчас **фильтруются не размеры "S", "XS" и "M", а все товары, у которых размер не "L" и "XL"**.

Улавливаете разницу?! Если в будущем к нашей таблице будут добавлены строки с другими размерами (например, XXL), то они тоже появятся в результатах фильтра!

Если вам такое поведение Power Query кажется странным, то хочу добавить, что на самом деле своя логика здесь есть. Если количество включенных галочек в фильтре меньше, чем снятых, то функция Table.SelectRows будет записана с условиями «равно». Если вы снимете меньше галочек, чем оставите, то выражение в M-коде будет записано «от противоположного», как в последнем случае.

Управлять этой логикой мы не можем, поэтому при фильтрации необходимо постоянно следить за теми выражениями, которые оказываются в строке формул, подправляя их по необходимости. В нашем случае, если мы хотим отфильтровать только размеры "S", "XS" и "M", это должно быть, очевидно:

```
= Table.SelectRows("#Измененный тип", each ([Размер]="S" or [Размер]="XS" or [Размер]="M"))
```

Вопросы быстродействия

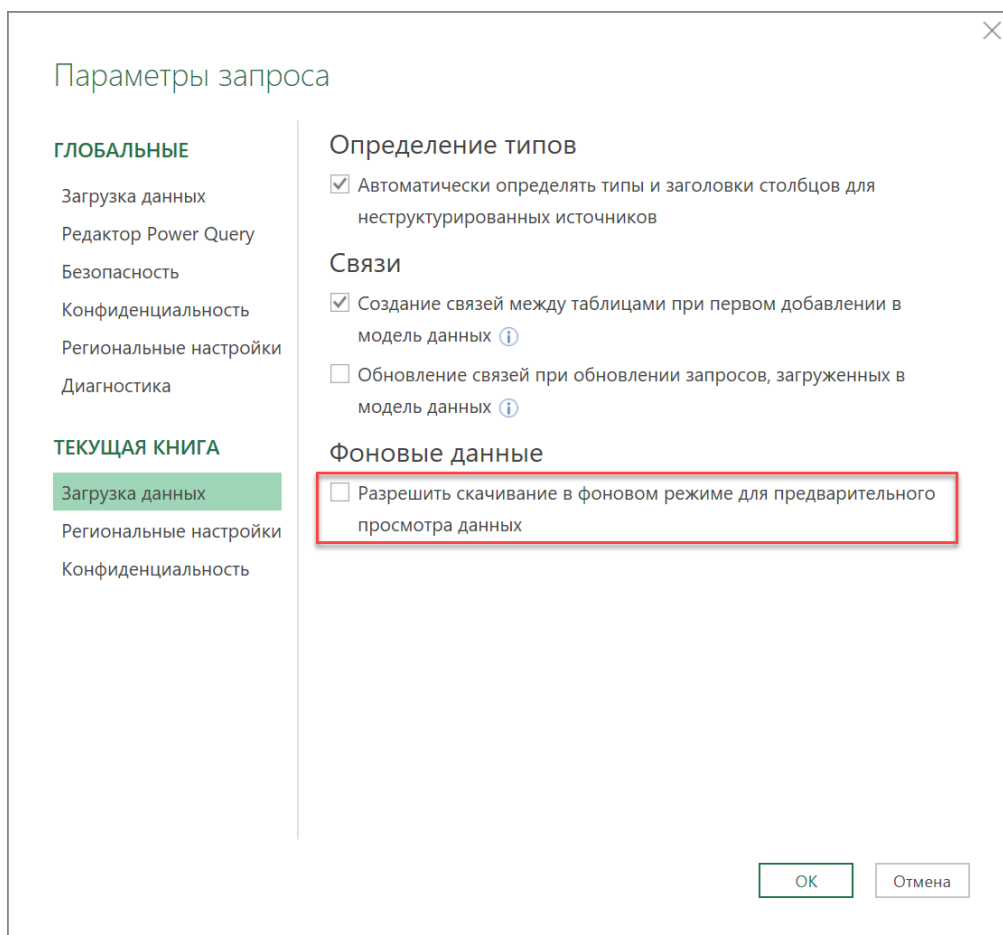
Рано или поздно любой опытный пользователь, применяющий в своей работе Power Query, сталкивается с ситуацией, когда его запросы (а их может быть много, и они, скорее всего, связаны) начинают ощутимо тормозить. Если после нажатия на кнопку **Обновить всё (Refresh All)** вам приходится неприлично долго ждать результатов, то, возможно, вам поможет один или несколько трюков, описанных далее.

Отключите фоновое обновление

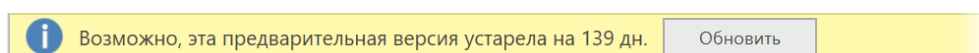
На самом деле, когда вы жмёте кнопку **Обновить всё (Refresh All)** на вкладке **Данные (Data)**, то Power Query не только пересчитывает все ваши запросы в книге, загружая обновлённые данные из соответствующих источников. Помимо этого, обновляются ещё и данные, закэшированные в памяти для создания предварительного просмотра – того самого, что мы видим в окне редактора Power Query. Это весьма значительно тормозит всю цепочку обновления запросов и съедает массу ресурсов вашего компьютера.

Отключить эту опцию можно в окне настроек Power Query, куда можно попасть из Excel через **Данные → Получить данные → Параметры запроса (Data → Get Data → Query Options)** или выбрав в самом Power Query меню **Файл → Параметры запроса (File → Query Options)**.

В открывшемся окне в группе **Текущая книга → Загрузка данных (Current Workbook → Data Load)** нужно снять соответствующий флажок:



Переживать по поводу того, что вы останетесь со старыми данными, не стоит: Power Query обязательно напомнит вам об этом при открытии запроса в редакторе и предложит обновиться вручную:

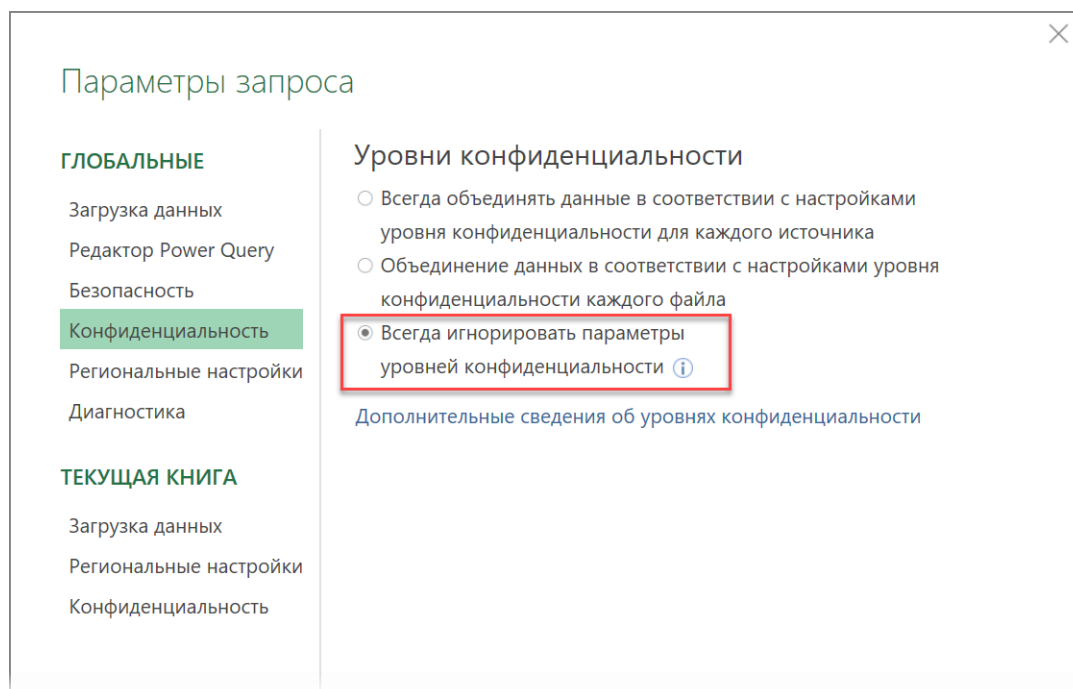


Отключите проверку конфиденциальности

Если помните, мы подробно разбирали механизм проверки конфиденциальности и установки уровней приватности для каждого используемого источника данных в главе [Настройка уровней конфиденциальности источников данных](#).

Если вы не работаете с конфиденциальной информацией, или она в любом случае не уходит у вас за пределы компании, то имеет смысл отключить эту проверку, на которую, естественно, уходит время и ресурсы процессора. Кроме того, при запросах к базам данных (SQL) это позволит перенести часть нагрузки на сервер (механизм *query folding*), что тоже ускорит процесс обновления.

Сделать это можно в предыдущем окне **Параметры запроса (Query Options)** в разделе слева **Глобальные → Конфиденциальность (Global → Privacy)**:



Используйте CSV-файлы вместо книг Excel

Если есть возможность выбора (например, ваша корпоративная ERP-система может выгружать отчёты в разных форматах), то всегда лучше использовать простые форматы типа CSV или TXT, чем обычные книги формата XLSX (и тем более старого XLS). Любой экселевский файл хранит кроме данных массу побочной служебной информации, которая много весит и ощутимо тормозит процесс.

Откажитесь от прямых ссылок на ячейки

В главе [Абсолютные и относительные ссылки в запросах](#) мы разбирали механизм, который позволяет ссылаться в формулах Power Query на отдельные ячейки внутри таблицы. Несмотря на заманчивость использования такого привычного по Excel подхода, не стоит им злоупотреблять: на больших объёмах данных он катастрофически замедляет пересчёт.

По той же причине не стоит использовать в запросах вертикально ориентированные формулы: вычисление суммы или среднего по столбцу, расчёт доли или отличия значения от предыдущих значений в этом же столбце и т. п.

Используйте функции Table.Buffer и List.Buffer

Если в ваших запросах есть таблица или список, к которым идёт многократное обращение, то можно попробовать применить к ним функции **Table.Buffer** или **List.Buffer**, чтобы один раз загрузить их в оперативную память и потом быстро запрашивать уже закэшированные данные.

Добавьте ключ при слиянии запросов

Если вы выполняете слияние (merge) запросов а-ля ВПР, то можно ощутимо ускорить обновление, добавив к таблице, из которой подставляются данные, внутренний невидимый уникальный ключ для каждой строки. Скорость обращения по этому ключу будет существенно выше, чем обычный многократный поиск каждого значения.

Добавить ключ можно либо напрямую в М-коде функцией **Table.AddKey**, либо, что гораздо проще, выполнив операцию удаления дубликатов (**Главная → Удалить строки → Удалить дубликаты**) над тем столбцом таблицы-справочника, по которому идёт поиск и подстановка данных. Даже если у вас не было повторов, выполнение этого действия активирует внутренний механизм Power Query по добавлению невидимого ключа, что весьма позитивно скажется на скорости обновления такого запроса.

Научно-популярное издание

Николай ПАВЛОВ

Скульптор данных в Excel с Power Query

Выпускающий редактор Анастасия Шмулий

Корректор Людмила Куртова

Художественное оформление Алина Кривошеина

Компьютерная вёрстка Алина Кривошеина

0+

*Знак информационной продукции согласно
Федеральному закону №436-ФЗ 29.12.2010 г.*

Подписано в печать 05.05.2019 г.

Формат 60x84/8. Усл. печ. л. 41,5

Ограниченный тираж.

Гарнитура Calibri.

Адрес электронной почты: info@delibri.ru

Сайт в интернете: letmeprint.me

ООО «Де`Либри»

109147, г. Москва, ул. Большая Андроньевская, д. 23, стр. 1

Отпечатано: АО «Т8 Издательские Технологии»

109316, г. Москва, Волгоградский проспект, дом 42, корпус 5

www.t8group.ru; info@t8print.ru

тел.: 8 (499) 332-38-30

ISBN 978-5-4491-0350-5



9 785449 103505

Это первая книга на русском языке, посвящённая надстройке Power Query – мощному инструменту для работы с данными в Microsoft Excel. С её помощью можно легко решать множество задач, для которых раньше требовались сложные формулы или макросы.

Прочитав эту книгу вы научитесь:

- ★ **Загружать** данные в Microsoft Excel из любых источников (файлов, интернета, баз данных, корпоративных программ и т.д.)
- ★ Моментально **собирать** данные из нескольких таблиц, листов или даже файлов.
- ★ **Наводить порядок** в таблицах (исправлять регистр, лишние пробелы, некорректные даты, числа-как-текст).
- ★ **Связывать** между собой таблицы по одному или нескольким столбцам без функций ВПР (VLOOKUP).
- ★ Всячески **трансформировать** и **анализировать** загруженные данные по своему усмотрению.

Об авторе



Николай Павлов – IT-тренер, разработчик и эксперт по программам пакета Microsoft Office и, в частности, по Excel.

За последние 8 лет провел более 500 корпоративных и открытых тренингов для сотрудников таких компаний, как Газпромнефть, Лукойл, ВТБ, Мегафон, ИКЕА, Coca-Cola, Ренова, РЖД, Valeant, Фармстандарт, Mercedes Benz, KIA, Рольф, Мегафон, МТС, RusOutdoor, Casio, Reima и многих других.

10-й год подряд получает номинацию **Microsoft Most Valuable Professional (MVP)** по Excel. Имеет статус **Microsoft Office Master**.

Автор сайта **PlanetaExcel.ru** с посещаемостью свыше 20 тыс. пользователей в день, одноимённого канала с обучающими видео по Excel на YouTube (более 8 млн. просмотров).



DELIBRI