Николай Павлов. Скульптор данных в Excel с Power Query

Power Query – мощный инструмент для работы с данными в Microsoft Excel. В версии Excel 2010 года Power Query позиционировалась как надстройка. Сейчас – это элемент стандартного интерфейса вкладки *Данные*. И в самой ленте Excel не найти упоминания имени Power Query. Хотя за функционалом, связанным с импортом данных это имя сохранилось. С помощью Power Query можно легко решать множество задач, для которых раньше требовались сложные формулы или макросы. В книге подробно разбираются вопросы импорта данных в Excel из внешних источников (файлов разных форматов, баз данных, интернета и т.д.) и трансформации полученных таблиц. Книга рассчитана на средних и продвинутых пользователей. Ко всем описанным в книге задачам в комплекте идут файлы-примеры, которые можно использовать в работе.

Поскольку я уже прочитал по теме две книги (Кен Пульс и Мигель Эскобар. Язык M для Power Query и Гил Равив. Power Query в Excel и Power BI: сбор, объединение и преобразование данных), в настоящей заметке я рассказал на том, что было для меня новым (или что я уже успел забыть).

Николай Павлов. Скульптор данных в Excel с Power Query. – М.: Де'Либри, 2019. – 332 с.

Электронную книгу с файлами примерами можно приобрести на сайте автора.

Начало работы с Power Query

Power Query умеет подключаться к большому числу источников данных, но не ко всем. Если у вас какой-то особый источник, отсутствующий в перечне стандартных, поищите в Интернете. Многочисленные энтузиасты и программисты давно разрабатывают для Power Query свои коннекторы для загрузки данных из нестандартных программ или форматов файлов. Рекомендую изучить репозиторий GitHub с готовыми примерами и документацией по пользовательским коннекторам или искать в интернете по фразе Power Query custom connector.

После создания запроса и размещения данных в Excel обновление происходит простым нажатием кнопки. Для профессиональной работы со сложными моделями данных можно использовать отдельную программу 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 (хотя есть и trial-версия).

Загрузка данных в Power Query

Универсальный способ загрузка данных из текущей книги Excel

Наряду с другими способами можно использовать встроенную функцию языка M, которая открывает доступ ко всему содержимому текущей книги. Пройдите *Данные –> Получить данные –> Из других источников –> Пустой запрос* и в строке формул редактора Power Query введите:

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

Нажмите Enter и увидите таблицу с содержимым текущей книги. В ней отображаются все «умные» таблицы, области печати и именованные диапазоны, но не листы книги. Просмотреть содержимое любого из перечисленных объектов можно, если щелкнуть мышью в белый фон соответствующей ячейки в столбце Content:

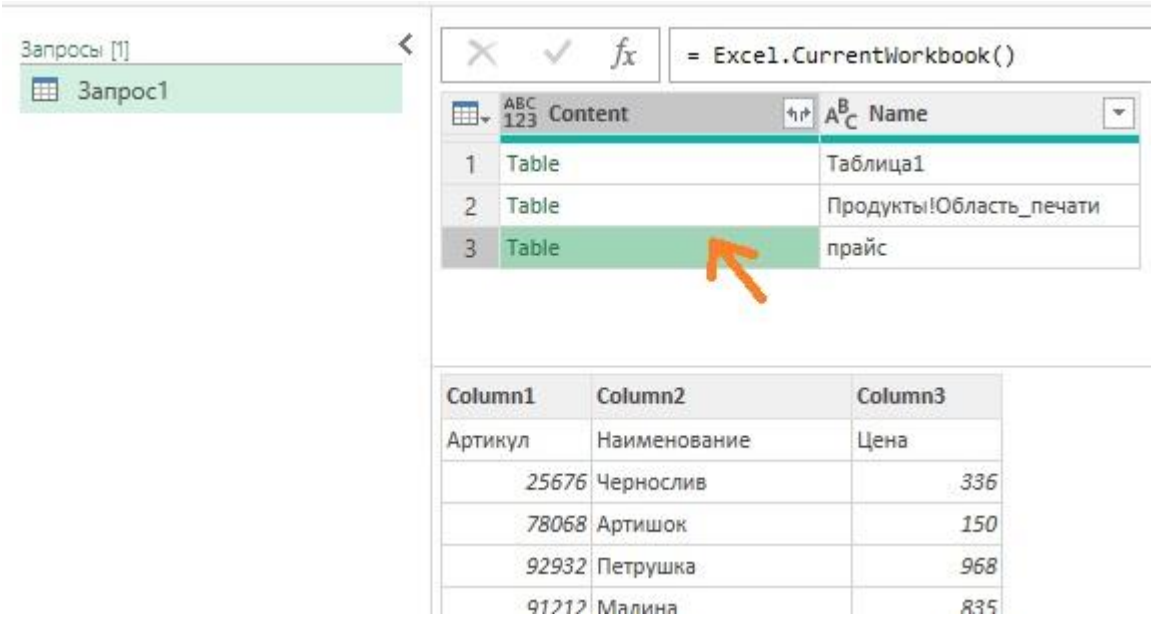


Рис. 1. Загрузка данных из текущей книги Excel с помощью функции Excel.CurrentWorkbook()

Если же щелкнуть мышью не в фон ячейки, а прямо по слову Table, то мы «провалимся» в выбранную таблицу. Это же действие можно выполнить, если щелкнуть правой кнопкой мыши по ячейке и выбрать команду *Детализация*. Эти действия отразятся на правой панели Power Query как шаг *Навигация*.

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). На сегодняшний день этот протокол поддерживает большинство современных корпоративных ERP- и CRM- систем и баз данных: 1С: Предприятие, Microsoft Dynamics, SAP, SQL Server, SharePoint и др.

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*. Далее можно обратиться к базе данных их Excel, пройдя *Данные –> Получить данные –> Из других источников –> Из веб-канала OData*. Задайте в качестве адресной строки имя веб-сервера и имя базы 1С, добавив к ним путь /odata/standard.odata/. Введите пароль, и вы получите доступ к таблицам 1С, которые разрешено импортировать.

Загрузка данных из PDF через Word

Задача переноса данных из таблицы в PDF-файле на лист Microsoft Excel – это всегда «весело».[1] Прямое копирование обычно ни к чему хорошему не приводит, т.к. после вставки скопированных данных на лист они, скорее всего, слипнутся в один столбец. При чем 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.

С 2013 года Microsoft Word научился открывать и распознавать PDF- файлы (даже отсканированные, т.е. без текстового слоя!). Открываем Word, жмем *Файл –> Открыть* и выбираем PDF-формат в выпадающем списке в правом нижнем углу окна. Затем выбираем нужный нам PDF-файл и жмем Открыть. Word сообщает нам, что собирается запустить

[1] На момент публикации заметки (апр.2022) Power Query обзавелся коннектором для импорта из pdf.

распознавание этого документа в текст. Соглашаемся. Сохраняем документ как веб-страницу: этот формат является общим знаменателем между Word и Excel. *Файл –> Сохранить как* и выбираем тип файла *Веб-страница в одном файле (*.htm, *mhtml)*.

Идем в Excel *Данные –> Получить данные –> Из файла –> Из XML*. Чтобы были видны не только XML-файлы меняем в выпадающем списке в правом нижнем углу окна фильтры на *Все файлы* и указываем наш MHTML-файл. Power Query ждет от нас XML поэтому импорт «споткнется». В появившемся окне нужно будет щелкнуть правой кнопкой мыши по непонятному для Power Query файлу и уточнить его формат:

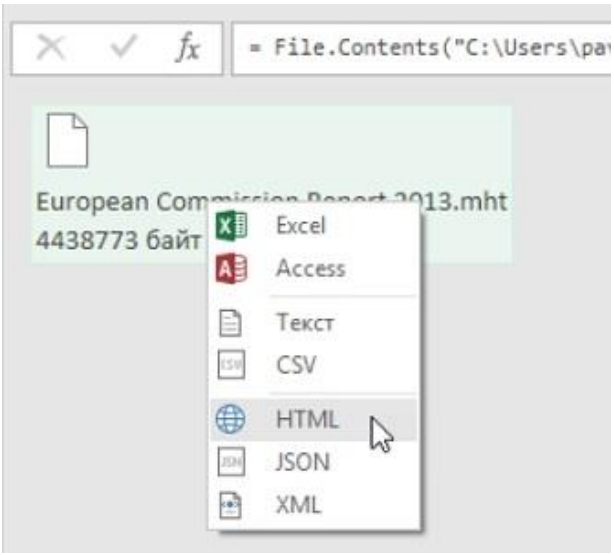


Рис. 2. Уточнение формата импортируемого файла

Power Query позволяет загрузить данных почты и календаря из Microsoft Exchange.

Типы слияния в Power Query

Объединение по нескольким столбцам

Power Query позволяет объединить две таблицы, когда поиск и подстановка данных из одной таблицы в другую должны происходить по совпадению не одного, а сразу нескольких параметров в нескольких столбцах.

Загрузите два запроса в режиме *Только подключение*. И в Excel пройдите *Данные –> Получить данные –> Объединение запросов –> Объединить*. В открывшемся окне выберете исходную таблицу (Заказы) и таблицу, откуда хотим подставить данные (Прайс) из выпадающих списков. Выделите в первой таблице, удерживая клавишу Ctrl, те столбцы, которые нужны при подстановке в любой последовательности. Например, Модель-Цвет-Память. Обратите внимание, что рядом с именами столбцов появятся их порядковые номера при выделении. Далее выделите те же столбцы во второй таблице, соблюдая исходную последовательность. Жмите Ok.

Рис. 3. Слияние по нескольким столбцам

Разверните кнопкой с двойными стрелками содержимое вложенных таблиц в столбце *Прайс* и выберите в раскрывающемся списке те колонки, которые хотите вывести в объединенной таблице (Цена):

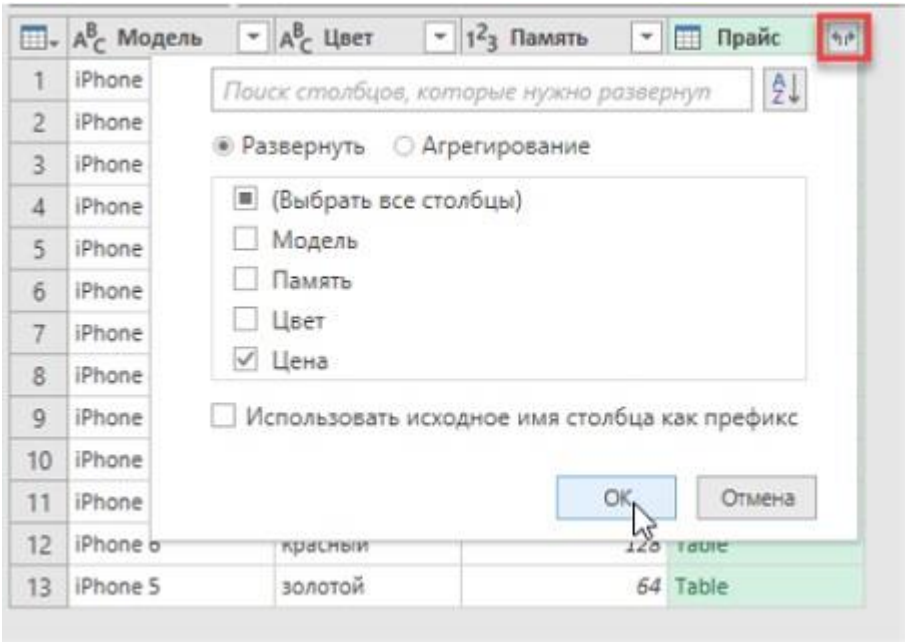


Рис. 4. Добавление столбца Цена

Сравнение таблиц объединением разных типов

Внешнее слева. Этот тип слияния имитирует поведение классической функции ВПР. На выходе получим все элементы из первой таблицы и в дополнительном столбце найденное совпадение из второй таблицы.

Внешнее справа. Вариант обратный предыдущему. На выходе все товары из второго списка и рядом с ними совпадения из первого.

Анти-соединение слева. Выводит позиции, которые есть в первом списке, но отсутствуют во втором.

Анти-соединение справа. Выводит позиции, которые есть во втором списке, но отсутствуют в первом.

Внутреннее. Этот тип слияния реализует пересечение двух множеств. На выходе мы получим список только тех товаров, которые присутствуют в обоих списках одновременно.

Полное внешнее. Этот тип слияния выводит общую таблицу, где присутствуют все товары из обоих списков. Те, что совпадают, встанут друг напротив друга в одной строке, остальные будут иметь в паре null.

Настройка уровней конфиденциальности источников данных

Power Query выделяет 4 уровня конфиденциальности:

- *Частный*. Личная информация, доступная 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...
- *Организационный*. 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...
- *Общий*. Данные, доступные всем. Публичные веб-страницы, открытые базы данных…
- *Нет*. В этом случае 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...

Если смешать в запросе данные из разных источников, а уровни для них не были заранее заданы, то получим сообщение об ошибке с предложением настроить недостающие уровни. Мы можем управлять тем, как PQ реагирует на смешивание данных из источников с разными уровнями, пройдя в PQ по меню *Файл –> Параметры и настройки –> Параметры запроса*:

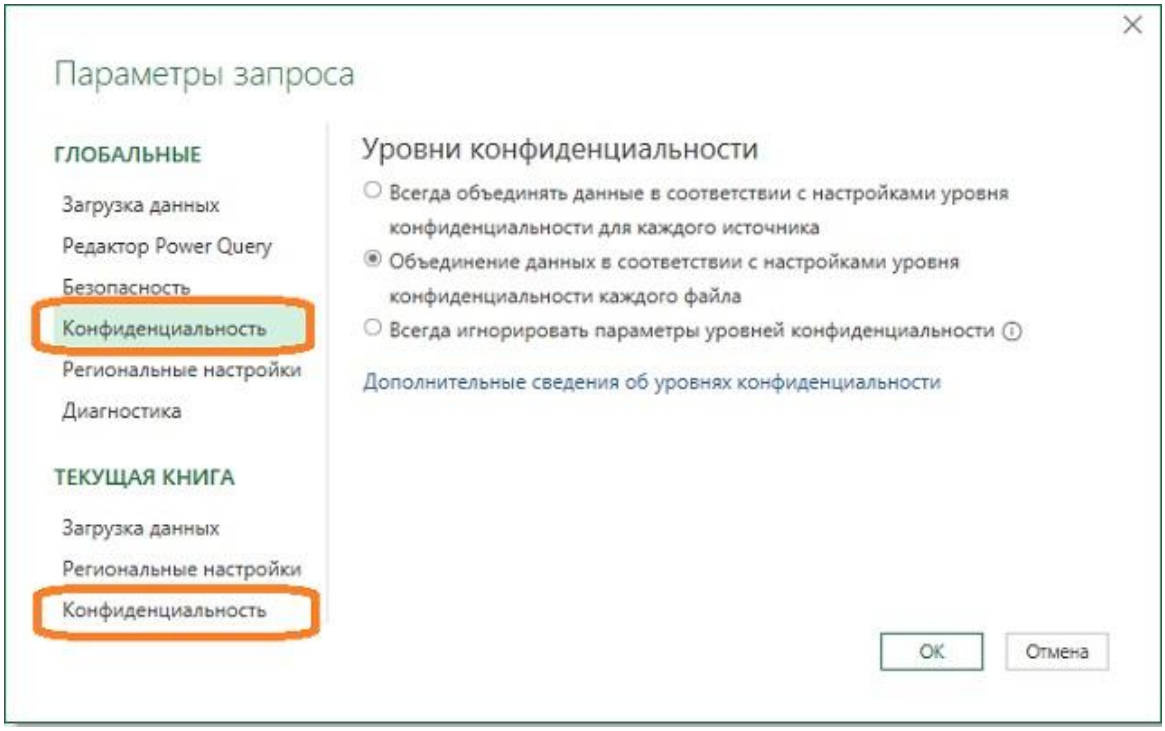


Рис. 5. Параметры запроса

Обратите внимание на два раздела *Конфиденциальность*: один отвечает за глобальные настройки для всего PQ, другой – за настройки для текущего файла. Если в глобальных настройках был выбран пункт 1 или 3, то настройки для текущей книги будут недоступны.

Преобразования таблиц

Первый/последний элемент в каждой группе

Рассмотрим сценарий поиска крайних элементов (строк) в каждой группе – последней сделки по клиенту/товару, последнего платежа, первой продажи заданного товара и т.п.

Загрузим таблицу в Power Query, отсортируем данные по возрастанию даты от старых к новым, используя кнопку фильтра в шапке столбца *Дата*. Выполним группировку *Преобразование –> Группировать по* и настроив диалоговое окно:

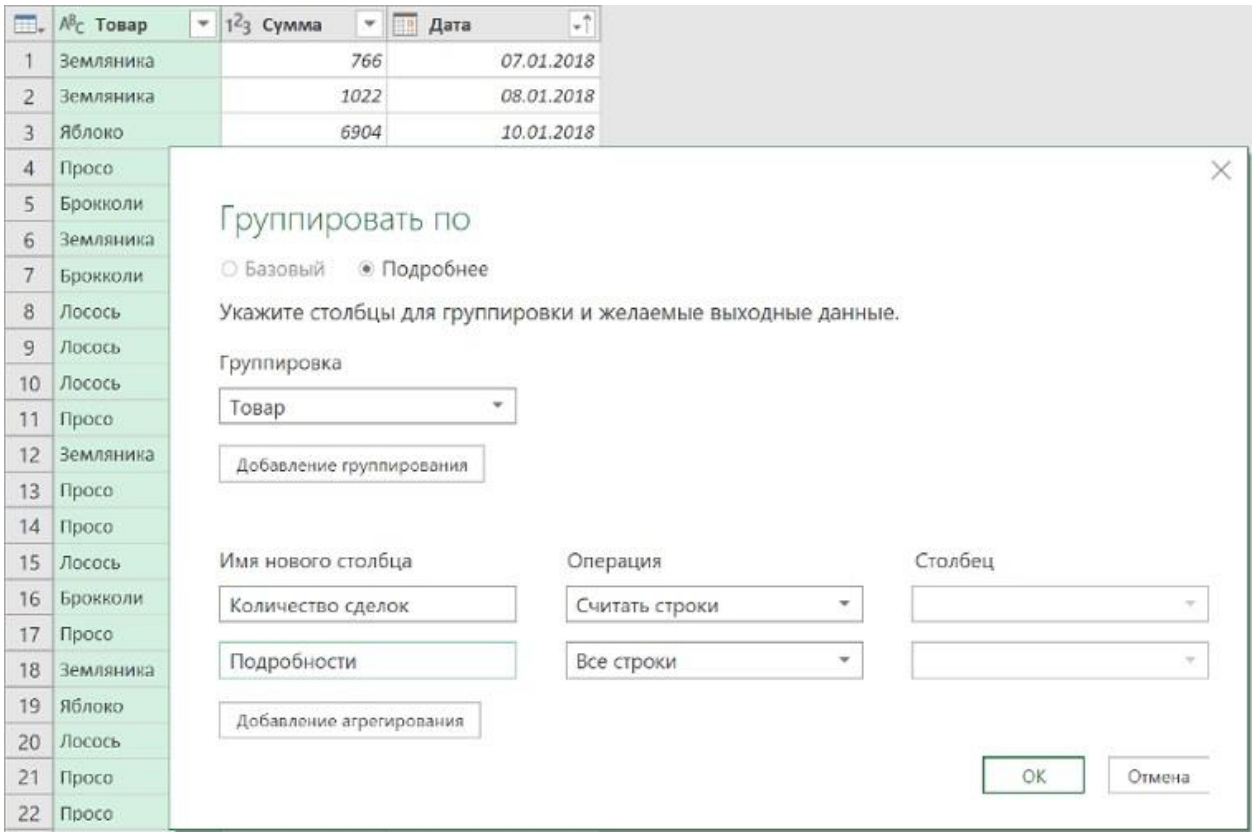
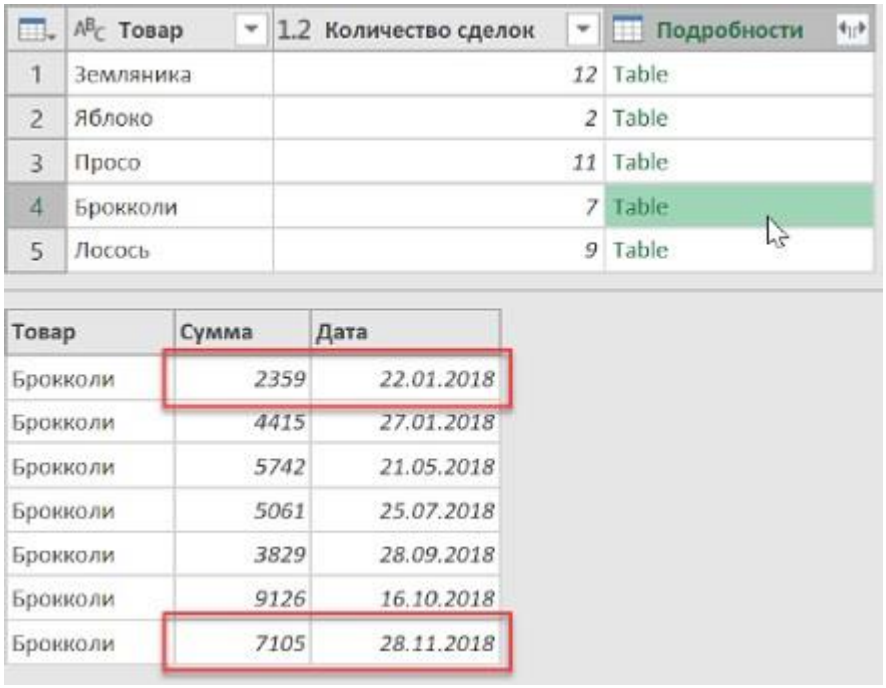


Рис. 6. Настройки группировки

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.

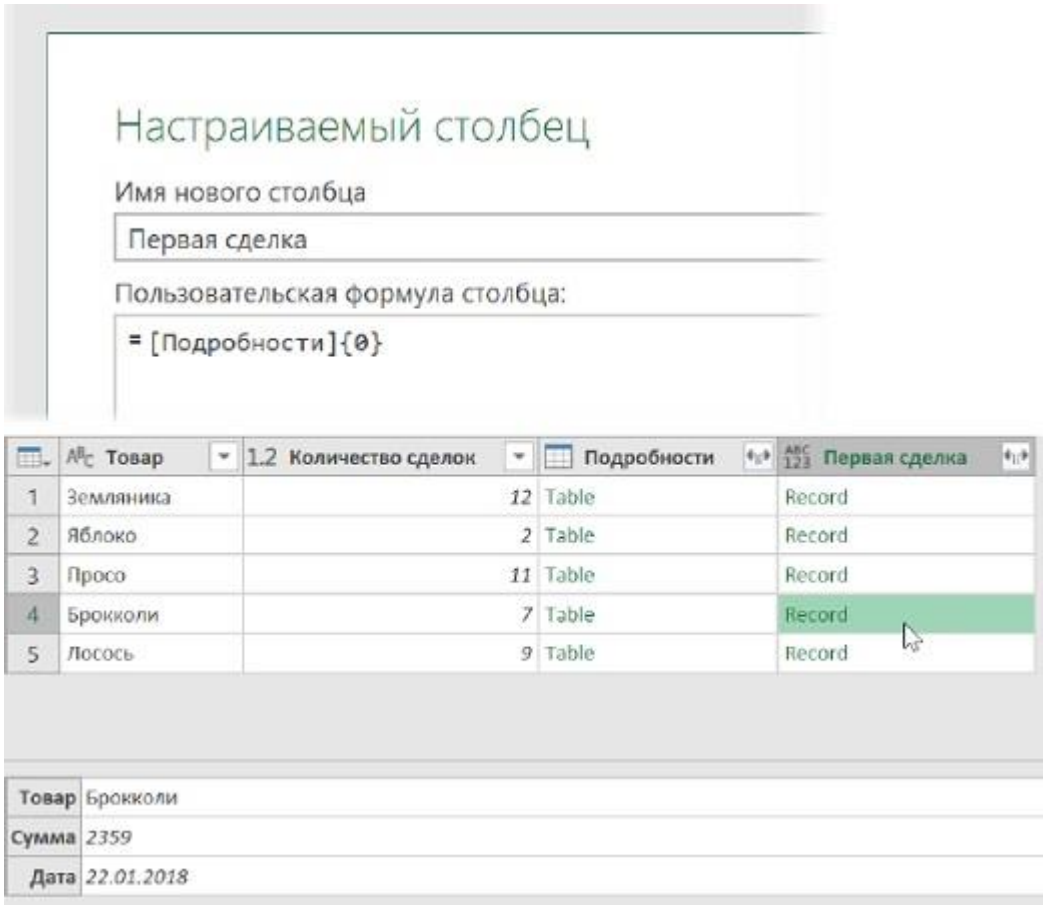


	Товар	Количество сделок	Подробности
1	Земляника	12	Table
2	Яблоко	2	Table
3	Просо	11	Table
4	Брокколи	7	Table
5	Лосось	9	Table

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

Рис. 7. Результат группировки

Для извлечения из каждой вложенной таблицы содержимое первой/последней строки добавим вычисляемый столбец:



	Товар	Количество сделок	Подробности	Первая сделка
1	Земляника	12	Table	Record
2	Яблоко	2	Table	Record
3	Просо	11	Table	Record
4	Брокколи	7	Table	Record
5	Лосось	9	Table	Record

Товар	Брокколи
Сумма	2359
Дата	22.01.2018

Рис. 8. Настраиваемый столбец для извлечения первой записи

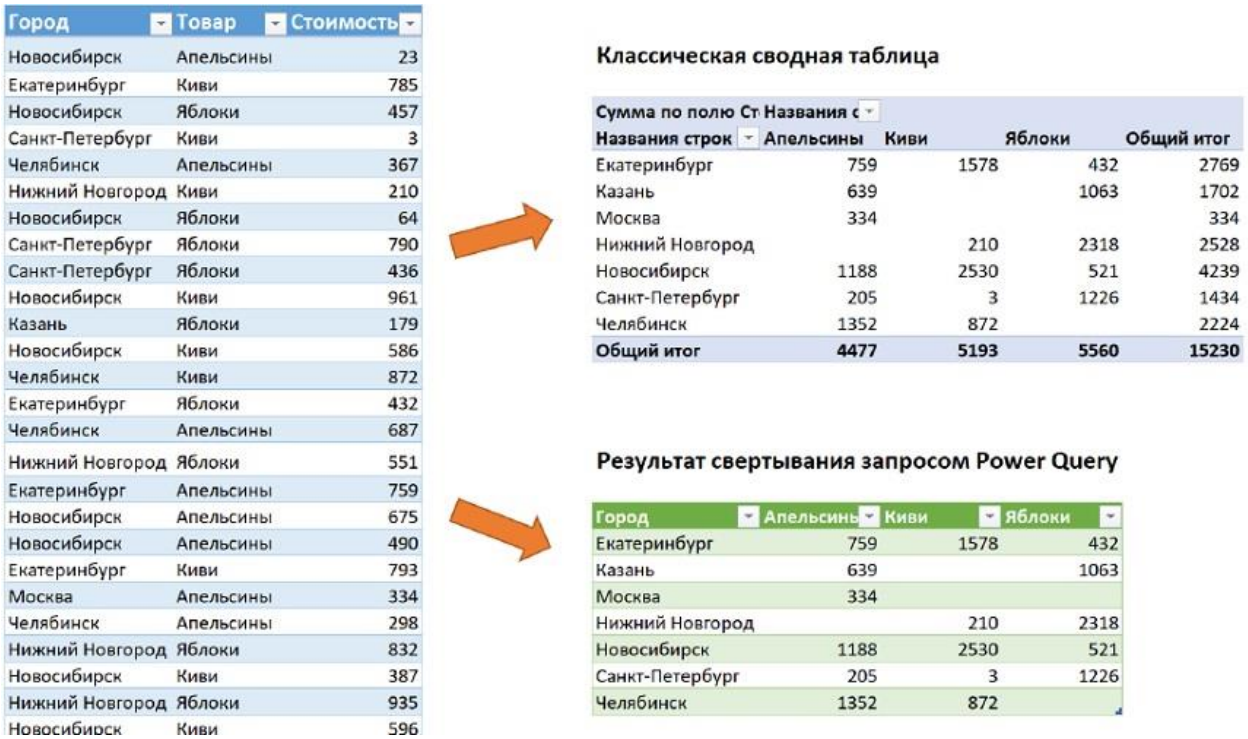
Формула [Подробности]{0} означает, что мы хотим взять первую строку из каждой вложенной таблицы столбца Подробности (нумерация строк в Power Query начинается с нуля). После нажатия Ok мы получим столбец с записями (Records) – первыми строками из вложенных таблиц по каждому товару. Как и списки (Lists), записи (Records) можно развернуть в новые столбцы.

Для получения дат и сумм по последним сделкам в настраиваемом столбце следует использовтаь формулу:

=[Подробности]{[Количество сделок]-1}

Свёртывание таблиц

Под свертыванием в Power Query понимается тип трансформации таблиц, при котором уникальные значения из заданного столбца превращаются в заголовки новых столбцов (напоминает сводную таблицу, но 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

Рис. 9. Свёртывание таблиц

Одним из важных преимуществ свёртывания в Power Query по сравнению с классической сводной таблицей является возможность помещать в область значений не числа, а текст.

Операции с текстом

В стандартном наборе команд для изменения регистра не хватает варианта, обычно называемого «Как в предложениях», когда заглавной становится только первая буква в ячейке, а не начальная буква в каждом слове. Реализовать это вариант можно, добавив настраиваемый столбец

=Text.Upper(Text.Start([Текст],1)) & Text.Lower(Text.End([Текст], Text.Length([Текст])-1))

Здесь: Text.Upper – преобразует текст, указанный в качестве аргумента, в верхний регистр; Text.Lower – преобразует весь текст в нижний регистр; Text.Length – определяет длину исходной строки текста; Text.Start – выдает заданное количество символов от начала строки текста (аналог ЛЕВСИМВ в Excel).

Функцией Text.Start мы отщипываем от строки начальный символ и с помощью Text.Upper преобразуем его в верхний регистр. Затем с помощью "&" приклеиваем к полученному заглавному символу остальную строку, преобразованную функцией Text.Lowe в нижний регистр.

Вместо настраиваемого столбца можно ещё воспользоваться кнопкой *Вызвать настраиваемую функцию*.

Обработка дат и времени

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 принят международный стандарт ISO 8601, в котором первой неделей года считается та, куда попадает первый четверг года или 4 января (в России используется ГОСТ Р 7.0.64-2018, созданный на базе

ISO 8601). В Excel есть функция НОМНЕДЕЛИ.ISO(), а в Power Query аналогичной встроенной функции пока нет.

Предположим, что у нас есть столбец с датами, для каждой из которых нужно определить номер недели по 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

Рис. 10. Исходные даты

Добавим пользовательский столбец *ISO Year*, чтобы понять, к какому году по ISO относится каждая дата. *Добавление столбца –> Пользовательский столбец*:

=Date.Year(Date.AddDays([Дата], 3 - Date.DayOfWeek([Дата],1)))

Добавим еще один столбец *Start Date* с датой 3 января каждого ISO-года для каждой даты:

=#date([ISO Year],1,3)

Наконец, добавим 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

Рис. 11. Последовательность из трех столбцов для вычисления номера недели по ISO

Можно заменить три столбца одним, используя формулу:

```
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 вы часто используете номер недели по ISO, то удобнее сделать пользовательскую функцию, которую можно использовать в разных запросах и разных файлах.

Работа с запросами

Описан экспорт запроса в файл подключения в формате ODC (Office Database Connection file), и открытие файла подключения в другой книге Excel Дана пошаговая инструкция обновления запросов по расписанию. Для этого используется *Планировщик Windows* – специально встроенной в любую версию Windows программой, которая умеет по расписанию выполнять заданные действия. Планировщик открывает файл Excel, а код VBA обновляет запросы по событию Workbook_Open.

Power Query и VBA

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.

Для помещения запроса в код VBA можно использовать следующую заготовку:

```
Sub Macro_Template()
  qname = "Мой запрос"          'имя создаваемого запроса
  ActiveWorkbook.Queries.Add Name:=qname, Formula:="Тут должен быть М-код запроса"
End Sub
```

Однако код запроса нельзя просто скопировать из окна *Расширенного редактора* Power Query и вставить между кавычками после параметра *Formula*. Необходимо соблюсти следующие правила синтаксиса VBA.

- 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 удвоены.

Язык М

Справка по встроенным функциям

Подробная справка по всем функциям языка М предоставлена Microsoft. Но можно получить справку и внутри редактора Power Query. Для этого создайте новый пустой запрос *Данные –> Получить данные –> Из других источников –> Пустой запрос*. Дайте имя запросу, например, Справка. В строке формул введите =#shared, нажмите Enter. На экране появится список всех встроенных функций языка М. Щелчок мышью в белый фон ячейки рядом со словом Function отобразит в нижней части окна подробную справку по интересующей вас функции:

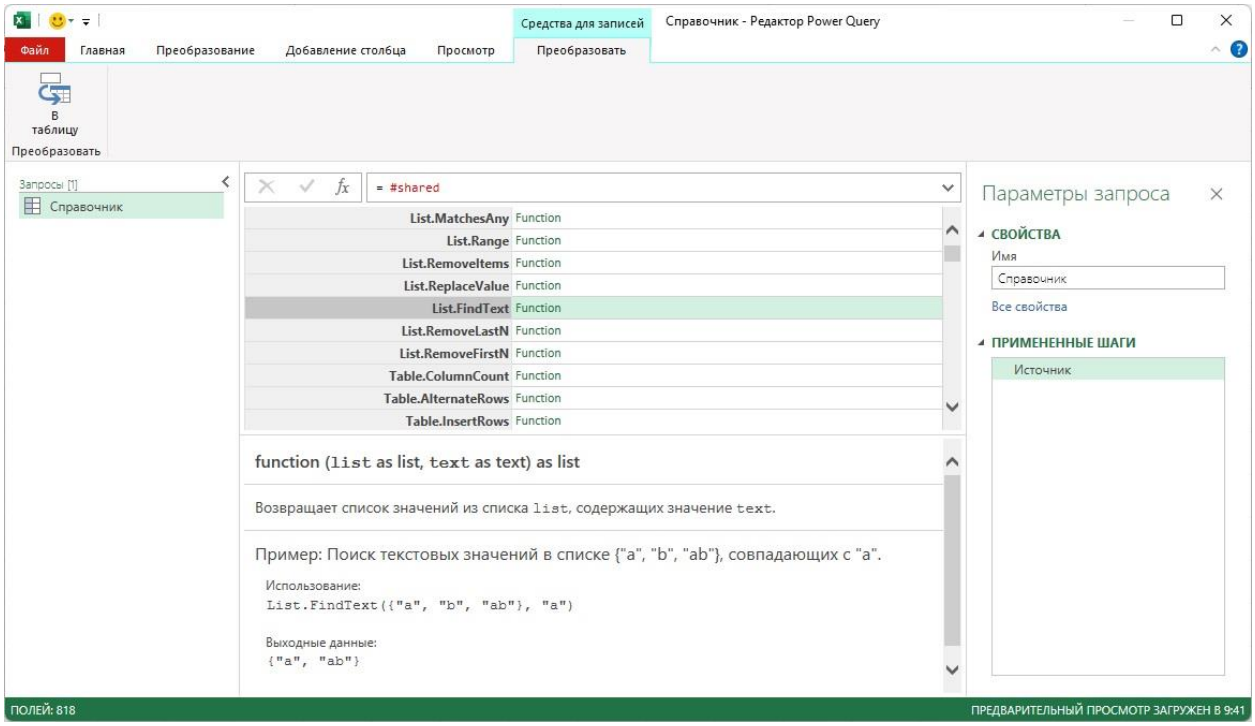


Рис. 12. Справочник функций языка M

Если щёлкнуть мышью в само слово *Function*, появится дополнительный шаг *Навигация* 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* на вкладке *Преобразовать*. После этого в шапке появятся привычные фильтры, которыми можно будет воспользоваться для быстрого поиска требуемых функций.

Запрос Справка можно сохранить как подключение и 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

Ещё один удобный инструмент для ввода и редактирования M-кода – это бесплатный текстовый редактор Notepad++. 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.

Скачайте и установите последнюю версию Notepad++ с сайта. Скопируйте файл M.xml, содержащий подсказки для функций языка M, из папки с примерами к этой книге в папку C:\Program Files\Notepad++\autoCompletion. Запустите Notepad++, выберите *Синтаксисы –> Задать свой синтаксис –> Импорт*. Укажите файл, содержащий информацию о цветовой подсветке синтаксиса – файл M Language Notepad Plus Markup.xml из папки с примерами к этой книге. Перезапустить Notepad++.

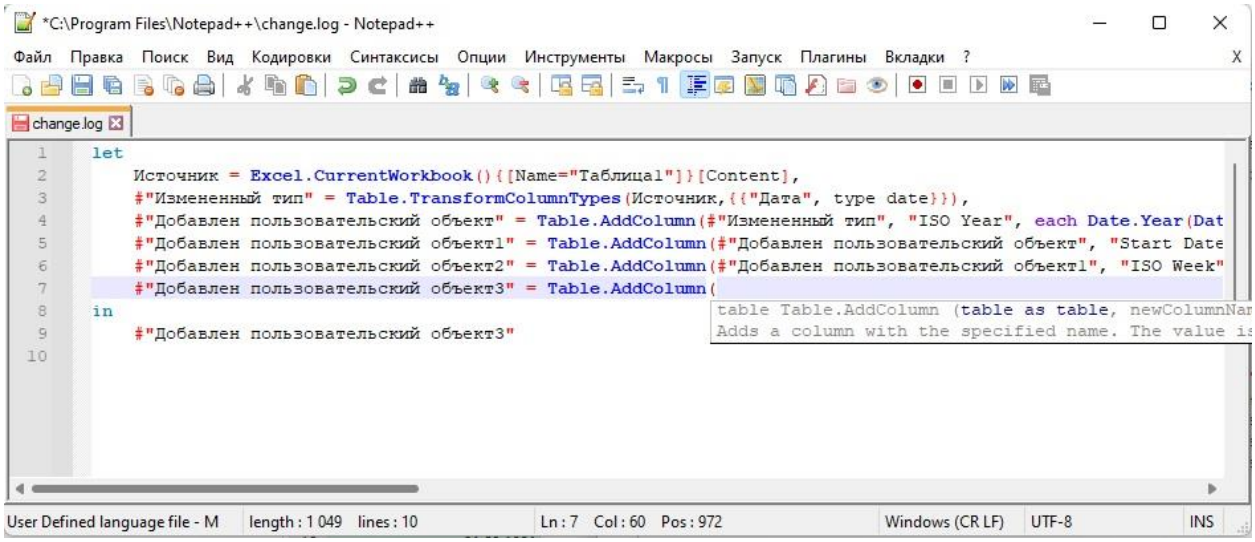


Рис. 13. Редактор M-кода на основе приложения Notepad++

Можно внести правки в цветовую схему через меню *Синтаксисы –> Задать свой синтаксис*, выбрав затем в верхней части окна из выпадающего списка наш язык M. А на вкладках *Ключевые слова*, *Комментарии и числа* и *Операторы и разделители* можно задать свои параметры форматирования (шрифт, цвет, начертание и т.д.) для каждой группы с помощью соответствующих кнопок *Стиль*.

Ключевое слово each

Ключевое слово each предназначено для создания и вызова небольших функций на лету. Например, нам нужно накинуть на исходную цену 20% НДС:

(Price) => Price*1.2

Поскольку имя функции может быть любым, выражение можно переписать, заменив имя аргумента Price на нижнее подчеркивание:

(_) => _*1.2

Each заменяет левую часть этого выражения, делая его ещё компактнее:

each _*1.2

Код на языке M:

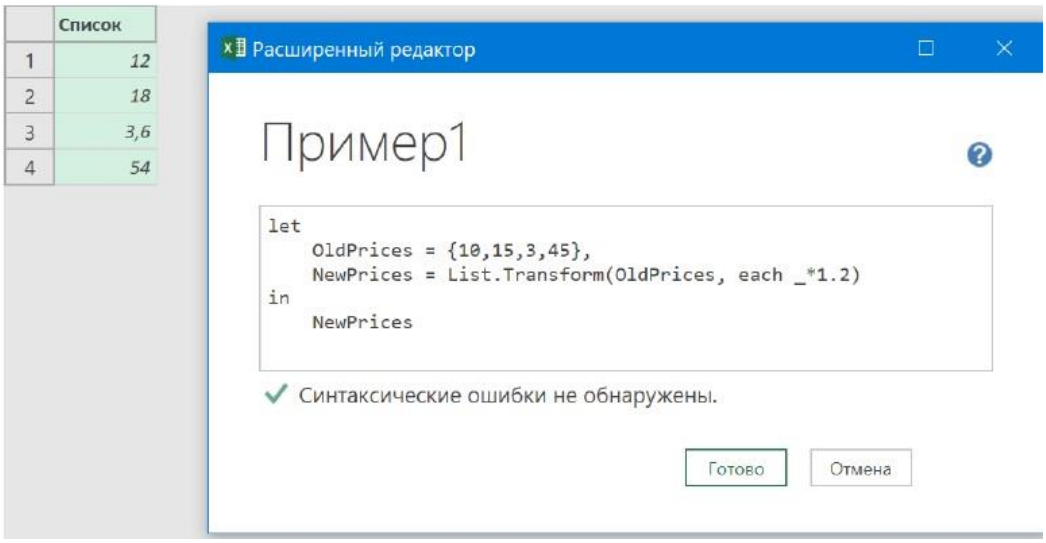


Рис. 14. Увеличение цен на НДС

Используется функция List.Transform, первым аргументом которой является исходный список, а вторым – функция, которая применяется к каждому его элементу. Еще несколько примеров для прояснения концепции использования ключевого слова *each*.

Обработка текстового списка. Допустим, нам нужно применить к каждому элементу текстового списка функцию преобразования регистра Text.Proper, чтобы сделать первую букву каждого слова прописной:

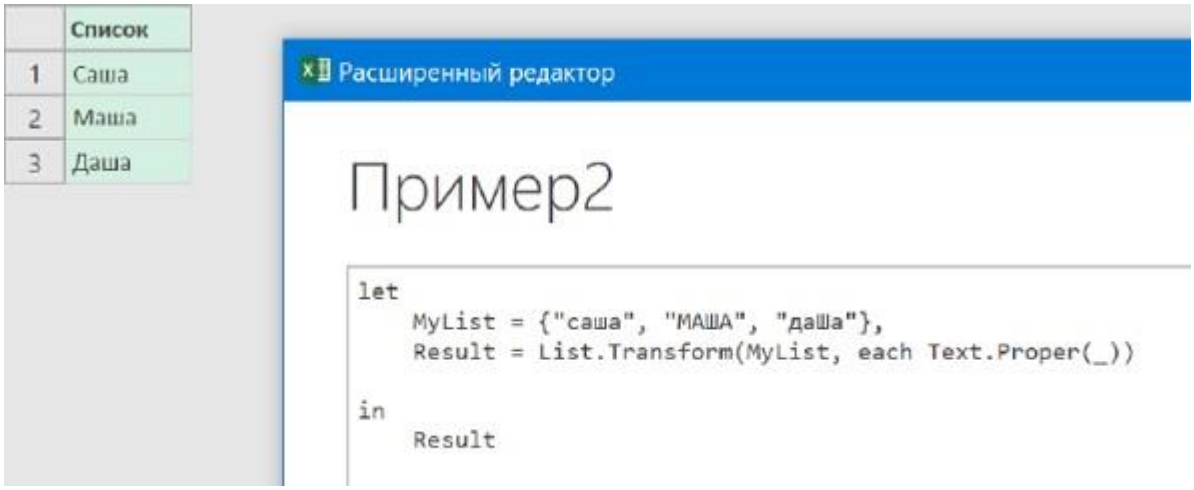
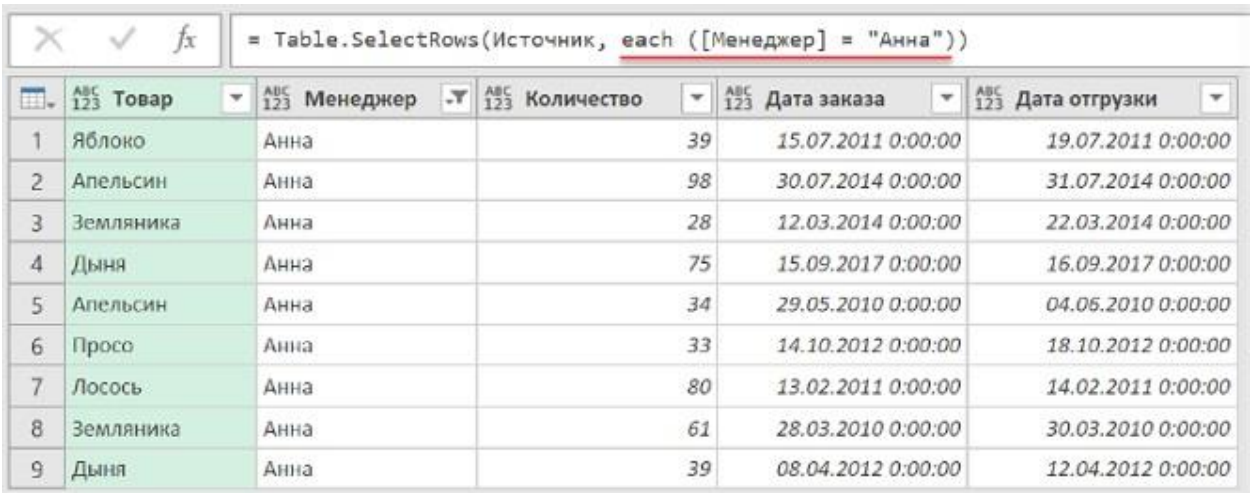


Рис. 15. Обработка текста

Фильтрация строк в таблице. Предположим, на шаге Источник мы загрузили в Power Query таблицу с данными по продажам. Если отфильтровать все сделки менеджера *Анны*, то в строке формул мы увидим, как 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

Рис. 16. Фильтрация строк

В этом случае символ нижнего подчёркивания не используется, т.к. идёт обращение к полю (столбцу) в таблице или записи. Формула…

= Table.SelectRows(Источник, each ([Менеджер] = "Анна"))

… эквивалентна созданию 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

Обработка ошибок в запросах

В Excel для обработки ошибокиспользуется функция ЕСЛИОШИБКА, а в M есть её аналог – конструкция…

try (проверяемое выражение) *otherwise* (что выводим вместо ошибки)

Это применимо для ошибок, возникающих в отдельных ячейках. 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, например…

Expression.Error: Столбец "Количество" таблицы не найден. Сведения: Количество

Здесь требуется вмешательство пользователя или другие, более сложные подходы (параметризация, проверка структуры таблиц и т.д.).

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.

Параметризация путей к файлам исходных данных

Если вы обмениваетесь файлами с коллегами, или открываете файл на разных ПК, то столкнётесь с одной надоедливой проблемой, связанной с постоянно ломающимися ссылками на исходные данные. Поскольку в 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 запрос работать не будет.

Например, вы создали запрос, обрабатывающий ТОП 100 продаж:

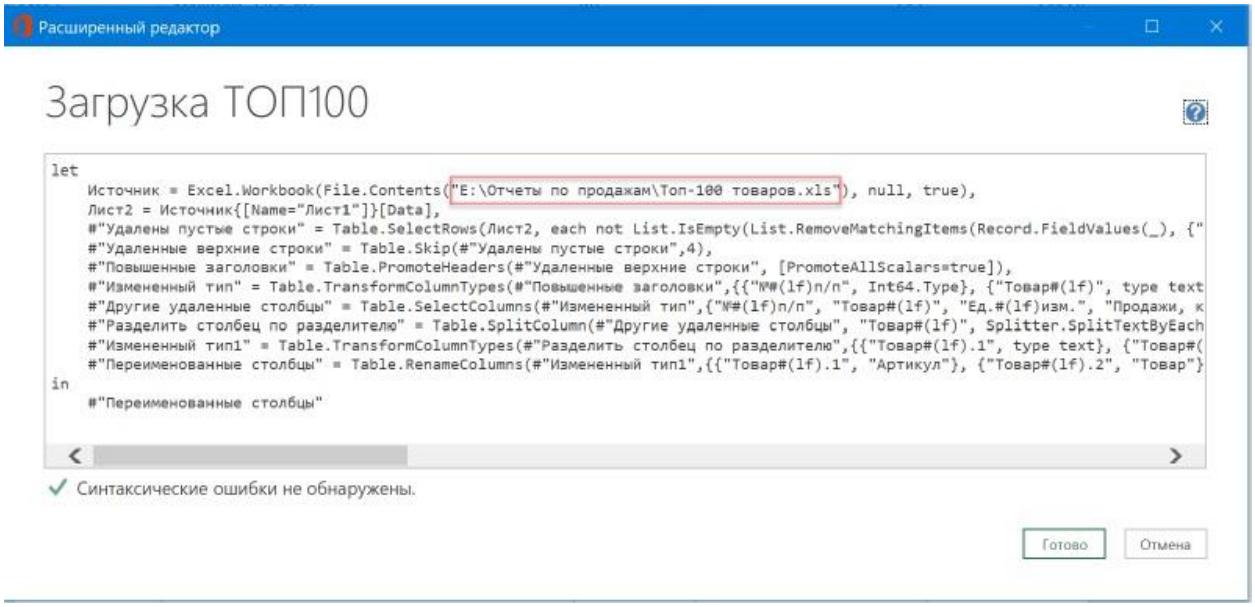


Рис. 17. Код запроса

Во второй строке виден жёстко прописанный путь к исходному файлу.

Добавьте новый пустой лист Excel в файл с запросом и создайте маленькую «умную» таблицу, в единственной ячейке которой будет записан полный путь к файлу исходных данных:

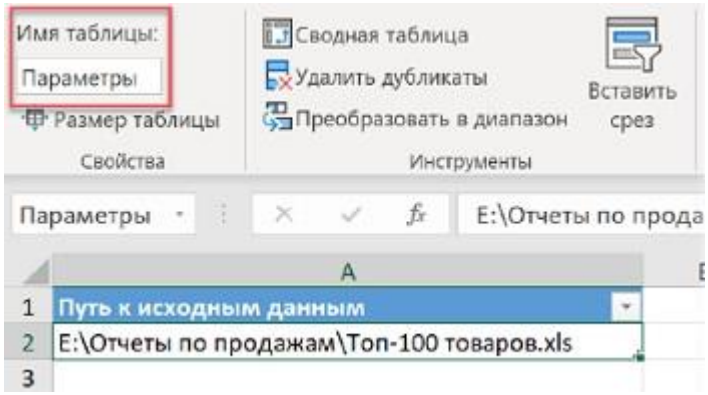


Рис. 18. Путь к исходному файлу

Имя заголовка в ячейке A1 и имя таблицы важно сохранить, так как на них будет ссылка в коде M. В ячейку A2 введите формулу:

=ЛЕВСИМВ(ЯЧЕЙКА("имяфайла");НАЙТИ("[";ЯЧЕЙКА("имяфайла"))-1)&"Топ-100 товаров.xls"

Мы считаем, что исходный файл расположен в той же папке, что и файл Excel с запросом. Функция ЯЧЕЙКА() с аргументом "имяфайла" отражает путь к файлу, имя файла и имя листа. В формуле мы урезаем это содержимое до знака [, с которого начинается имя файла, и к пути (оставшейся части) добавляем имя исходного файла.

Осталось отредактировать ссылку на путь к файлу в запросе:

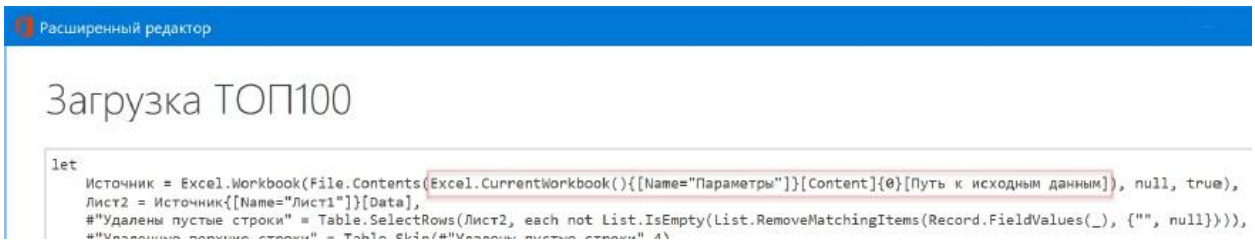


Рис. 19. Имя исходного файла берется из ячейки Excel

Excel.CurrentWorkbook(){[Name="Параметры"]}[Content]{0}[Путь к исходным данным]

Здесь Excel.CurrentWorkbook() – это функция языка M для обращения к содержимому текущего файла. {[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.

Танцы на граблях

Переименование столбцов

Дать столбцам в таблице данных после импорта удобные и наглядные имена – естественное и правильное желание. Однако, в будущем это может привести к сбою запроса. Если предположить, что могут меняться названия столбцов, но не их порядок, то лучше переименовать столбцы, привязываясь не к их старым названиям, а к их положению в таблице.

	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

Рис. 20. Переименование столбцов по их положению

Чтобы наш запрос стал универсальным, нужно в исходном коде шага переименования заменить жёстко прописанные имена столбцов…

```
= Table.RenameColumns(
   #"Повышенные заголовки",
   {
      {"Region", "Город"},
      {"Date", "Дата"},
      {"Units", "Количество"}
   }
)
```

…на соответствующие им номера…

```
= Table.RenameColumns(
   #"Повышенные заголовки",
   {
      {Table.ColumnNames(#"Повышенные заголовки"){0}, "Город"},
      {Table.ColumnNames(#"Повышенные заголовки"){1}, "Дата"},
      {Table.ColumnNames(#"Повышенные заголовки"){2}, "Количество"}
   }
)
```