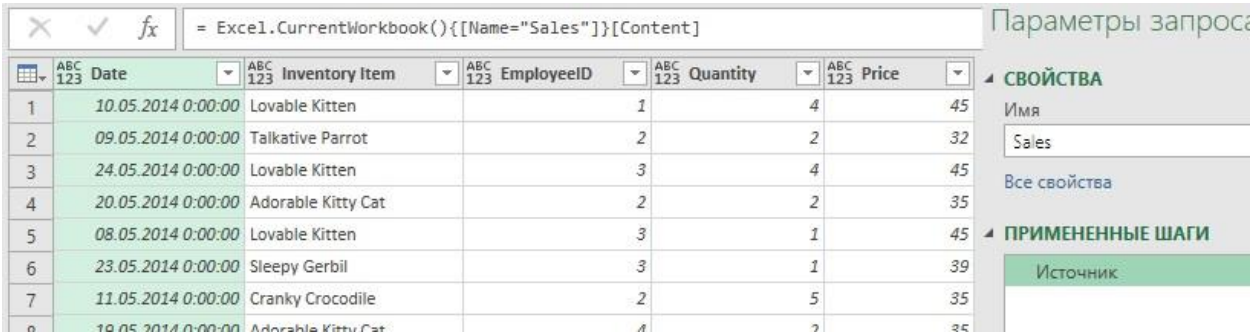
Глава 20. Понимание языка M

Это продолжение перевода книги Кен Пульси Мигель Эскобар. Язык M для Power Query. Главы не являются независимыми, поэтому рекомендую читать последовательно.

Предыдущая глава Содержание Следующая глава

Откройте *Understanding M.xlsx*. Выберите любую ячейку в таблице Sales. *Данные –> Из таблицы/диапазона*. В редакторе Power Query удалите шаг *Измененный тип*.



	Date	Inventory Item	EmployeeID	Quantity	Price
1	10.05.2014 0:00:00	Lovable Kitten	1	4	45
2	09.05.2014 0:00:00	Talkative Parrot	2	2	32
3	24.05.2014 0:00:00	Lovable Kitten	3	4	45
4	20.05.2014 0:00:00	Adorable Kitty Cat	2	2	35
5	08.05.2014 0:00:00	Lovable Kitten	3	1	45
6	23.05.2014 0:00:00	Sleepy Gerbil	3	1	39
7	11.05.2014 0:00:00	Cranky Crocodile	2	5	35

Рис. 20.1. Начальный вид запроса

Ранее вы управляли кодом через пользовательский интерфейс. Вы видели, что Power Query действует как макро-рекордер, и взаимодействовали с ним через окно ПРИМЕНЕННЫЕ ШАНИ. У вас также был небольшой опыт взаимодействие через строку формул. Пора познакомиться с языком программирования. *Главная –> Расширенный редактор*. Откроется окно, содержащее код запроса:

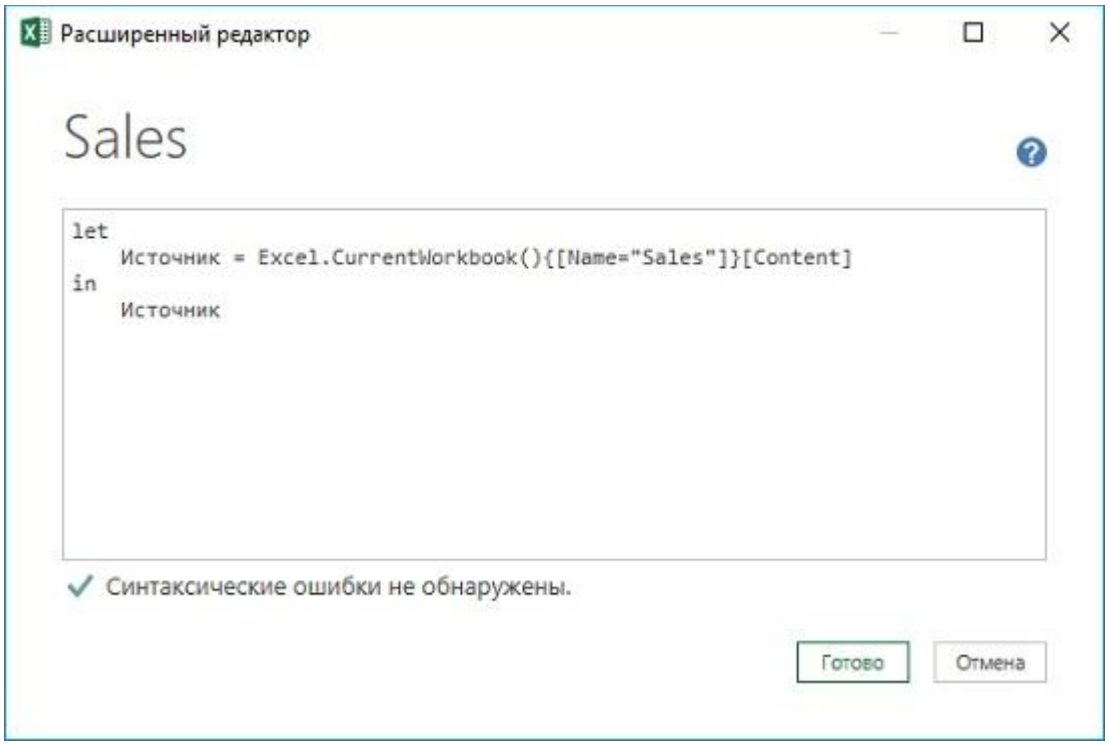


Рис. 20.2. Расширенный редактор

Расширенный редактор – это, по сути, текстовый редактор с проверкой синтаксиса. Взгляните на код внутри окна. Первая строка каждого запроса должна включать единственное слово *let* (для функций может быть иначе, см. главу 21). Вторая строка начинается со слова *Источник*. Это имя шага из поля ПРИМЕНЕННЫЕ ШАГИ. Это имя устанавливает соответствие шагов и строк кода. В нашем примере Power Query вызвал функцию Excel.CurrentWorkbook(). Затем он добавил список импортированных записей (таблиц Excel). Наконец, он извлек содержимое записей для этого объекта. Синтаксис записи можно представить, как Функция(){Объект}[Поле]. Мы уже видели эту конструкцию раньше, см. главу 19, раздел *Создание записи из строки таблицы*.

Предпоследняя строка каждого запроса состоит из единственного слова *in*. Последняя строка (одно слово) – ссылка на имя шага, содержащего данные, возвращаемые запросом. Как правило, это имя последнего шага, но не обязательно.

Добавим еще один шаг к нашему запросу. Закройте *Расширенный редактор*, нажав кнопку *Готово*. Щелкните правой кнопкой мыши столбец *Price –> Тип изменения –> Десятичное число*. Вернитесь в *Расширенный редактор*.

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Sales"]}[Content],
    #"Измененный тип" = Table.TransformColumnTypes(Источник,{{"Price", type number}})
in
    #"Измененный тип"
```

Рис. 20.3. В запрос добавлен новый шаг

Обратите внимание. Во-первых, в конец второй строки была добавлена запятая. Это важно: каждая строка между строками *let* и *in* должна заканчиваться запятой. Исключение – последняя строка перед *in.* Во-вторых, написание имени второго шага отличается по синтаксису от первого. Отличие вызвано наличием пробела в имени *Измененный тип*. Power Query рассматривает эти два слова как отдельные термины.

Если хотите, переименуйте шаг. Это можно сделать двумя способами. Щелкните правой кнопкой мыши имя шага в области ПРИМЕНЕННЫЕ ШАГИ (перед входом в *Расширенный редактор*) и переименуйте шаг. В самом *Расширенном редакторе* удалите каждое вхождение #"Измененный тип" и замените на NewType. Независимо от того, какой метод вы выберете, ваш код приобретет вид:

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Sales"]}[Content],
    NewType = Table.TransformColumnTypes(Источник,{{"Price", type number}})
in
    NewType
```

Рис. 20.4. Упрощение кода удалением пробела из названия шага

Связывая шаги вместе

Обратите внимание, что отдельные строки связаны друг с другом. Так, например, в строке NewType есть ссылка на *Источник*:

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Sales"]}[Content],
    NewType = Table.TransformColumnTypes(Источник,{{"Price", type number}})
in
    NewType
```

Рис. 20.4. Элемент в строке NewType ссылается на Источник

Именно такие ссылки позволяют Power Query связывать все команды вместе. Вы можете интерпретировать строку NewType следующим образом: получить выход предыдущего шага и загрузить его в функцию преобразования типов данных для столбцов.

Разрыв строк

Строки можно переносить без особых ограничений. Когда Power Query читает код, он ищет ключевые слова и запятые. При чтении строки он игнорирует запятые, заключенные в круглые, фигурные или квадратные скобки и кавычки. Но обнаружив одинокую запятую, он распознает ее как конец строки и дальнейший код относит к следующему шагу. А найдя ключевое слово *in*, он понимает, что запрос завершен, и смотрит, какой шаг нужно вернуть. Это означает, что код…

```
NewType = Table.TransformColumnTypes(Source,{{"Price", type number}})
```

… эквивалентен коду…

```
NewType = Table.TransformColumnTypes
        (Source,{{"Price", type number}})
```

… или…

```
NewType = Table.TransformColumnTypes(
        Source,
        {
                {"Price", type number}
        }
)
```

Вы не можете вставить перенес в середине имени функции или слова, но перенос любого знака препинания допустим.

Такое представление кода может быть очень полезно при отладке строк с большим количеством различных скобок. Разделяя открывающие и закрывающие скобки и пары элементов списка на отдельные строки, вероятность ошибиться меньше. Еще один бонус этого подхода заключается в том, что очень легко добавлять новые столбцы в шаг изменения типов данных TransformColumnTypes. Давайте потренируемся и добавим установку типов данных для всех столбцов:

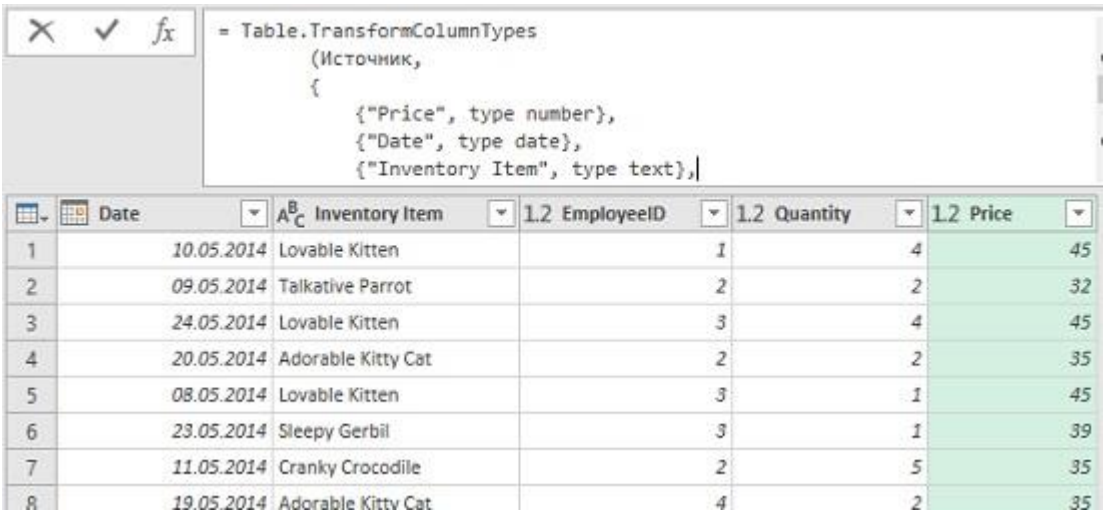
Column(s)	Data Type
Date	Date
Inventory Item	Text
EmployeeID, Quantity	Number

Рис. 20.5. Типы данных для столбцов таблицы *Sales*

Вы уже знаете из главы 19, что все элементы списка должны быть разделены запятыми:

```
let
        Source = Excel.CurrentWorkbook(){[Name="Sales"]}[Content],
        NewType = Table.TransformColumnTypes
        (Source,
                {
                        {"Price", type number},
                        {"Date", type date},
                        {"Inventory Item", type text},
                        {"EmployeeID", type number},
                        {"Quantity", type number}
                }
        )
in
        NewType
```

Если вы введете этот код в *Расширенный редактор* и нажмете *Готово*, то увидите, что запрос возвращает желаемые результаты:



	Date	Inventory Item	EmployeeID	Quantity	Price
1	10.05.2014	Lovable Kitten	1	4	45
2	09.05.2014	Talkative Parrot	2	2	32
3	24.05.2014	Lovable Kitten	3	4	45
4	20.05.2014	Adorable Kitty Cat	2	2	35
5	08.05.2014	Lovable Kitten	3	1	45
6	23.05.2014	Sleepy Gerbil	3	1	39
7	11.05.2014	Cranky Crocodile	2	5	35
8	19.05.2014	Adorable Kitty Cat	4	2	35

Рис. 20.6. Столбцы в запросе изменили тип данных в соответствии с кодом

Комментарии к коду

Помимо основного предназначения комментарии полезны для временного отключения строки кода. Чтобы отметить одну строку кода в качестве комментария, поместите две косые черты в начале:

```
let
        // Retrieve the Sales table from Excel
        Source = Excel.CurrentWorkbook(){[Name="Sales"]}[Content],
```

Иногда вам нужны более длинные комментарии, которые не помещаются в одной строке. В этом случае вы помещаете символы /* перед текстом, который вы не хотите выполнять, и символы */ в конце раздела, например:

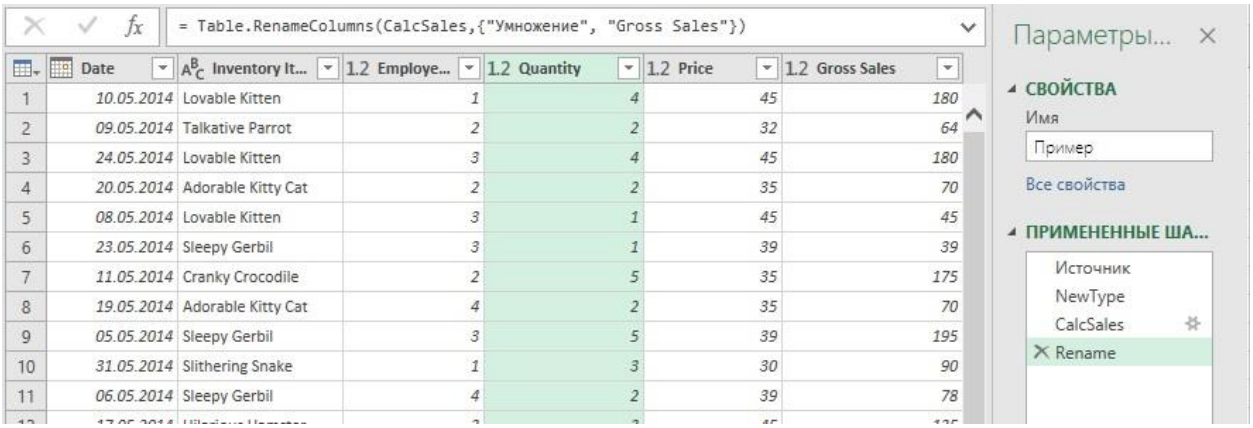
```
        /* I broke this code comment across multiple lines
        (by reading M is for Data Monkey) */
```

Символы
(Параметры функции)
{Список}
[Запись]
"Текст"
#"Имя шага"
//комментарий
/* комментарий в несколько строк */

Рис. 20.7. Сводка специальных символов Power Query

Работа с каждой строкой в столбце

Существует еще одна очень важная конструкция для понимания M: как читать и изменять код, который работает на каждой строке в столбце. Чтобы получить этот код, можно взять существующий запрос и добавить столбец для определения общего объема продаж по строкам. Убедитесь, что вы вышли из *Расширенного редактора*. Выберите столбцы *Quantity* и *Price* –> *Добавление столбца* –> *Стандартны* -> *Умножить*. Щелкните правой кнопкой мыши столбец *Умножение* –> *Переименовать* –> *Gross Sales*. Щелкните правой кнопкой мыши шаг *Вставлено умножение* –> *Переименовать* –> *CalcSales*. Щелкните правой кнопкой мыши на шаге *Переименованные столбцы* –> *Переименовать* –> *Rename*:



	Date	Inventory It...	Employe...	Quantity	Price	Gross Sales
1	10.05.2014	Lovable Kitten	1	4	45	180
2	09.05.2014	Talkative Parrot	2	2	32	64
3	24.05.2014	Lovable Kitten	3	4	45	180
4	20.05.2014	Adorable Kitty Cat	2	2	35	70
5	08.05.2014	Lovable Kitten	3	1	45	45
6	23.05.2014	Sleepy Gerbil	3	1	39	39
7	11.05.2014	Cranky Crocodile	2	5	35	175
8	19.05.2014	Adorable Kitty Cat	4	2	35	70
9	05.05.2014	Sleepy Gerbil	3	5	39	195
10	31.05.2014	Slithering Snake	1	3	30	90
11	06.05.2014	Sleepy Gerbil	4	2	39	78

Рис. 20.8. Запрос после обработки

Просмотр кода в *Расширенном редакторе* показывает две новые строки кода (отступы добавлены для удобства восприятия):

```
  CalcSales = Table.AddColumn(NewType, "Умножение",
        each [Quantity] * [Price], type number),
  Rename = Table.RenameColumns(CalcSales,
        {"Умножение", "Gross Sales"})
```

Функция RenameColumns ссылается на предыдущий шаг (CalcSales), а затем предоставляет список предыдущего имени столбца и новое имя. В этой строке вы можете переименовать несколько столбцов, если используете список списков. Например:

```
Rename = Table.RenameColumns(CalcSales,
        {
                {"Умножение", "Gross Sales"},
                {"Price", "Price1"}
        })
```

Функции *Table.AddColumn* добавляет столбец и имеет четыре аргумента, причем первые три – обязательные (как получить справку по функции см. последний раздел главы 19):

1. Имя предыдущего шага
2. Имя нового столбца. Поскольку на следующем шаге вы переименовали это имя "Умножение" на "Gross Sales", можно сразу ввести здесь нужное имя.
3. Действие, которым получаются значения в новом столбце. Обратите внимание на новое ключевое слово – *each*. Оно указывает Power Query, что действие должно выполняться для каждой строки запроса. После этого вы какие столбцы следует умножать друг на друга
4. Тип данных, возвращаемый функцией (необязательный).

Вы можете внести изменения, чтобы сделать код короче:

```
let
    Источник = Excel.CurrentWorkbook(){[Name="Sales"]}[Content],
    NewType = Table.TransformColumnTypes
        (Источник,
        {
            {"Price", type number},
            {"Date", type date},
            {"Inventory Item", type text},
            {"EmployeeID", type number},
            {"Quantity", type number}
        }
        ),
    CalcSales = Table.AddColumn(NewType, "Gross Sales", each [Quantity] * [Price], type number)
in
    CalcSales
```

Рис. 20.9. Оптимизированный код

Завершите запрос. Переименуйте его в *GrossSales*. *Главная –> Закрыть и загрузить*.

Ссылка на шаги или строки

Рассмотрим сценарий, в котором у вас есть текстовый файл:

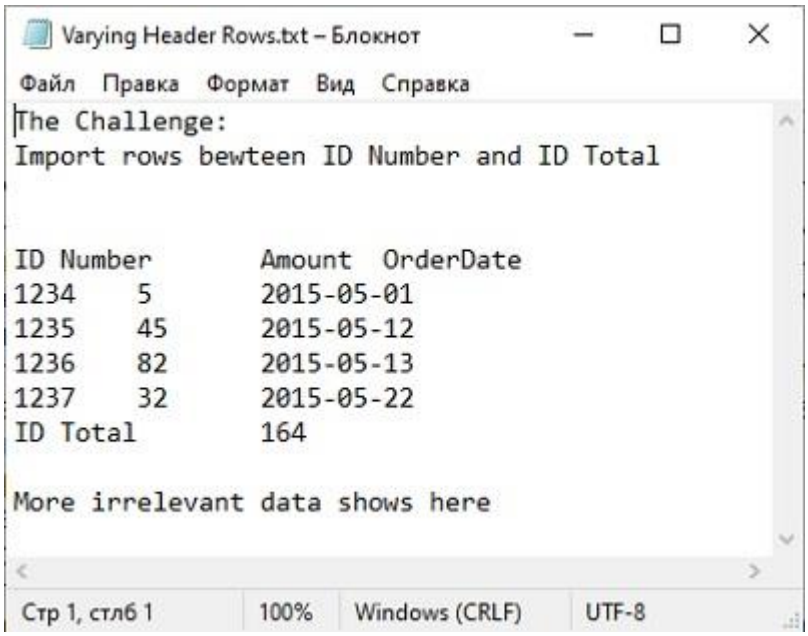


Рис. 20.10. Текстовый файл с разделителем табуляция

Обратите внимание:

- Данные разделены табуляцией. Однако, в первой строке табуляций нет. Это означает, что разделять столбцы нужно будет вручную.
- Число строк, предшествующих строке заголовка может варьироваться от 5 до 5000.
- Количество строк, между заголовком и итогами также является переменным.

Вам нужно извлечь строки из середины набора данных, причем точное положение извлекаемых строк не известно. Допустим, вам также нужно подсчитать количество дней между датами заказов. Это нельзя сделать в пользовательском интерфейсе Power Query.

Создайте пустую книгу Excel. *Данные –> Из текстового/CSV-файла –>* загрузите *Varying Header Rows.txt*. В окне предварительного просмотра кликните *Изменить*. В редакторе Power Query перейдите на вкладку *Добавление столбца –> Столбец индекса*. Фильтр *Column1 –> Текстовые фильтры –> Начинается с…* Установите для фильтра значения начинается с *ID Number* или начинается с *ID Total*.

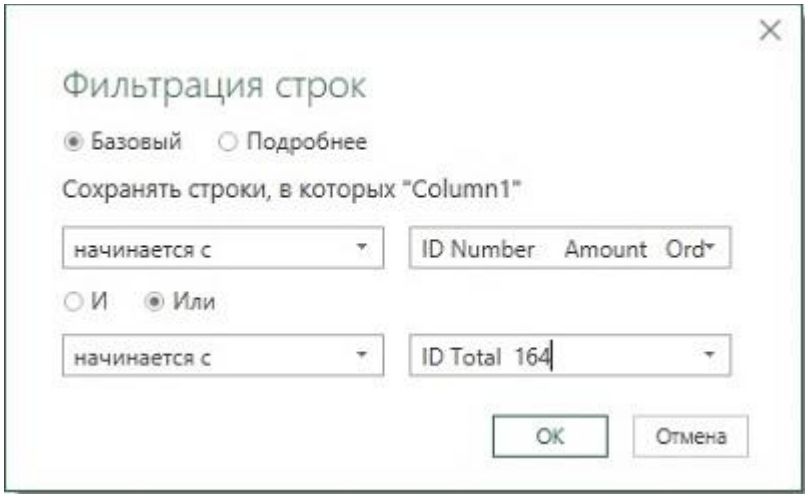


Рис. 20.11. Настройки фильтра строк

Вы можете достичь той же цели, фильтруя строки, начинающиеся с ID, но вы не знаете, есть ли в наборе данных другие строки, начинающиеся с ID. Делая условия как можно более жесткими, вы можете уменьшить вероятность ошибок в будущем. (Конечно, если еще какая-то строка начинается с ID, вам придется решать, как с ней поступать.) Переименуйте шаг *Строки с примененным фильтром* в *RowNumbers*.

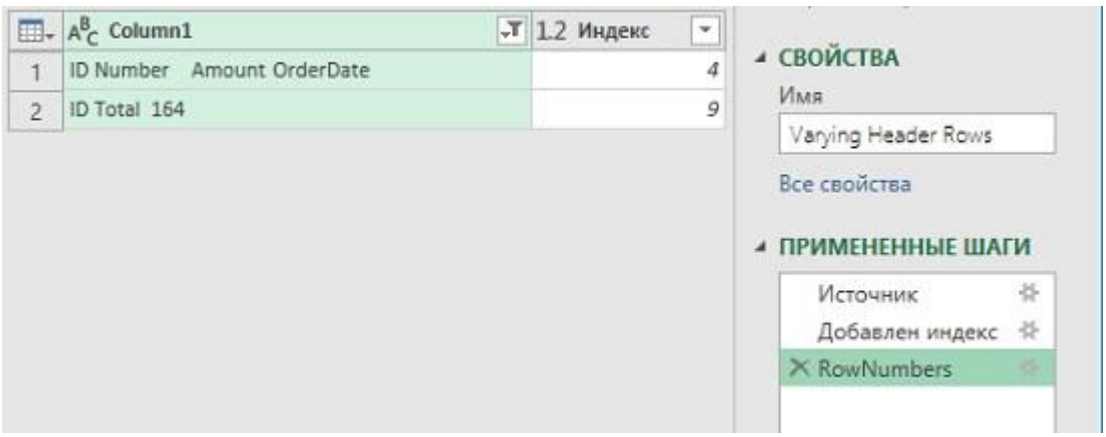


Рис. 20.12. Запрос показывает только две нужные строки с их индексами

Обратите внимание, что ваш фильтр динамичен: не важно, под какими номерами появляются строки, начинающиеся с *ID Number* или *ID Total*, в любом случае отфильтруются нужные строки.

Объединение шагов

Вы можете посмотреть код, пройдя по меню *Главная –> Расширенный редактор*.

```
let
    Источник = Table.FromColumns({Lines.FromBinary(File.Contents(
        "...\\Varying Header Rows.txt"), null, null, 1251)}),
    #"Добавлен индекс" = Table.AddIndexColumn(Источник, "Индекс", 0, 1),
    RowNumbers = Table.SelectRows
        (
        #"Добавлен индекс", each Text.StartsWith
            (
            [Column1], "ID Number#(tab)Amount#(tab)OrderDate")
                or Text.StartsWith([Column1], "ID Total#(tab)164"
            )
        )
in
    RowNumbers
```

Рис. 20.13. Исходный код

Объединение шагов похоже на создание мегаформул в Excel. Вы можете подставлять предыдущий шаг, в последующую ссылку. Например, в предыдущем коде строка *#"Добавлен индекс…* может войти в строку *RowNumbers*:

```
let
    Источник = Table.FromColumns({Lines.FromBinary(
        File.Contents("D:\\Varying Header Rows.txt"), null, null, 1251)}),
    RowNumbers = Table.SelectRows
        (
            Table.AddIndexColumn(Источник, "Индекс", 0, 1),
            each Text.StartsWith
            (
            [Column1], "ID Number#(tab)Amount#(tab)OrderDate")
            or Text.StartsWith([Column1], "ID Total#(tab)164"
            )
        )
in
    RowNumbers
```

Рис. 20.14. Один шаг сокращен

Создание новых шагов

Для создания нового шага кликните на кнопку *fx* слева от строки формул:

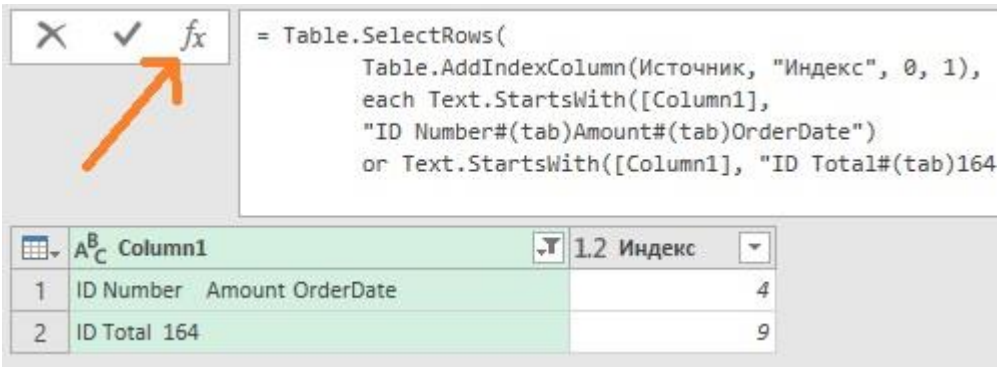


Рис. 20.15. Создание нового шага запроса вручную

В области ПРИМЕНЕННЫЕ ШАНИ вас появится новый шаг, а в строке формул, заготовка для написания формулы шага:

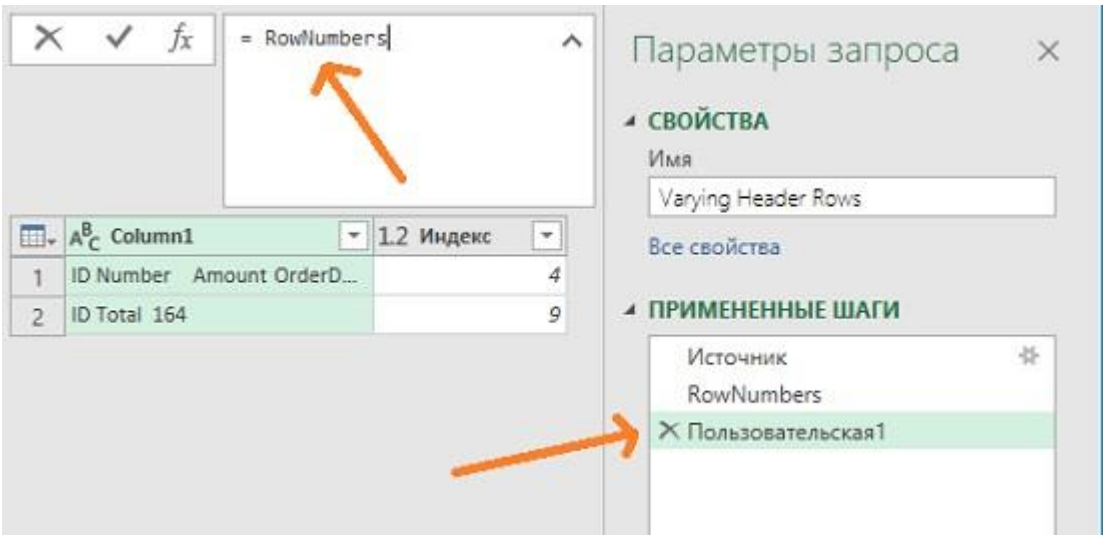


Рис. 20.16. Новый шаг и заготовка для формулы

Нажатие *fx* всегда создает новый шаг, который ссылается на предыдущий. Применительно к нашей задаче, измените ссылку на =Источник. Войдите в *Расширенный редактор.* Вы увидите, что появилась запятая в конце шага *RowNumbers* и финальная ссылка изменилась с *RowNumbers* на *Пользовательская1*.

```
let
    Источник = Table.FromColumns({Lines.FromBinary(
        File.Contents("D:\\Varying Header Rows.txt"), null, null, 1251)}),
    RowNumbers = Table.SelectRows
        (
            Table.AddIndexColumn(Источник, "Индекс", 0, 1),
            each Text.StartsWith
            (
            [Column1], "ID Number#(tab)Amount#(tab)OrderDate")
            or Text.StartsWith([Column1], "ID Total#(tab)164"
            )
        ),
    Пользовательская1 = Источник
in
    Пользовательская1
```

Рис. 20.17. Изменения, вызванные добавлением шага

Итак, у вас есть шаг *RowNumbers*, чтобы отфильтровать первую и последнюю строки таблицы, подлежащей обработке, и шаг *Пользовательская1*, чтобы начать писать логику обработки.

Вернитесь в пользовательский интерфейс, чтобы создать синтаксис обработки, а не писать его с нуля. Сначала удалите ненужные нижние строки, начинающиеся с индекса 10 (если вы начнете с удаления верхних строк, то индекс изменится). На самом деле, удалить нижние строки сложно, так как интерфейс запросит число строк для удаления, а вы его не знаете (его можно только вычислить). Поэтому сохраните 9 верхних строк. В редакторе Power Query встаньте на шаг *Пользовательская1.* Пройдите по меню *Главная –> Сохранить строки –> Сохранить верхние строки –> 9.* Теперь вы получаете таблицу фильтруется, чтобы показать только верхние 9 строк:

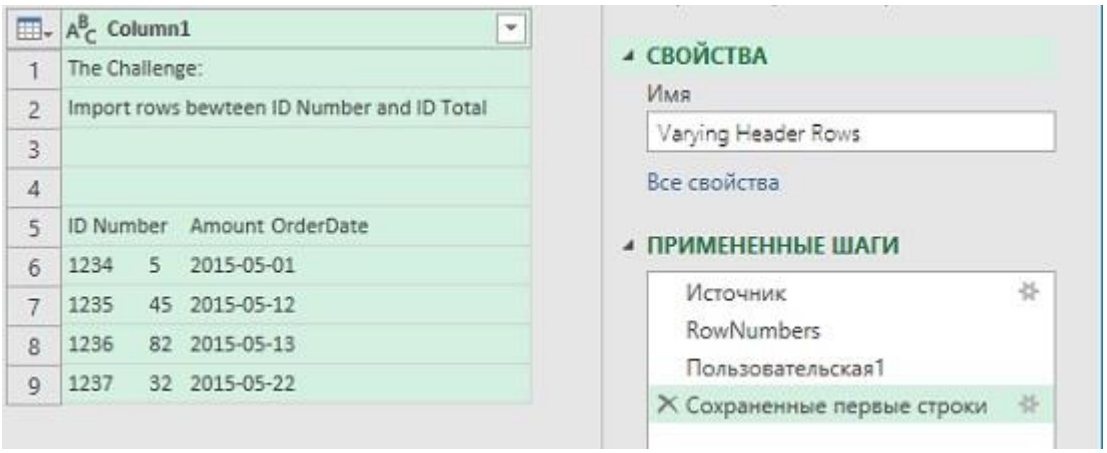


Рис. 20.18. Данные содержат верхние 9 строк

Обратите внимание, что у вас нет строки, которая начинается с *ID Total*. Этого следовало ожидать, потому что строка *ID Total* была десятой строкой в файле. Поскольку Power Query начинает счет с нуля, строке *ID Total* соответствует индекс 9. Однако, отсутствие строки *ID Total* вам на руку, так как всё равно общие итоги вам не нужны.

Зайдите в *Расширенный редактор*, и посмотрите, какой код сгенерировал Power Query:

#"Сохраненные первые строки" = Table.FirstN(Пользовательская1,9)

Чтобы код работал корректно на любых исходных данных, замените цифру 9 «правильным» значением из шага *RowNumbers*. Это можно сделать, обратившись к шагу *RowNumbers* и извлекая значение из столбца *Индекс* для второй строки (помните, что строка 2 имеет индекс 1; см. рис. 20.16):

RowNumbers[Индекс]{1}

Теперь запись шага выглядит так:

#"Сохраненные первые строки" = Table.FirstN(Пользовательская1,RowNumbers[Индекс]{1})

Можно еще упростить код, включив строку…

Пользовательская1 = Источник,

… в следующую строку:

```
let
    Источник = Table.FromColumns({Lines.FromBinary(
        File.Contents("D:\\Varying Header Rows.txt"), null, null, 1251)}),
    RowNumbers = Table.SelectRows
        (
            Table.AddIndexColumn(Источник, "Индекс", 0, 1),
            each Text.StartsWith
            (
            [Column1], "ID Number#(tab)Amount#(tab)OrderDate")
            or Text.StartsWith([Column1], "ID Total#(tab)164"
            )
        ),
    #"Сохраненные первые строки" = Table.FirstN(Источник,RowNumbers[Индекс]{1})
in
    #"Сохраненные первые строки"
```

Рис. 20.19. Оптимизированный код

Закройте *Расширенный редактор*. Удалите первые четыре строки таблицы. *Главная –> Удалить строки –> Удаление верхних строк –> 4.* Переименуйте шаг *Удаленные верхние строки –> ExtractRows*. Перейдите в *Расширенный редактор*. Отредактируйте формулу, заменив…

ExtractRows = Table.Skip(#"Сохраненные первые строки",4)

… на

ExtractRows = Table.Skip(#"Сохраненные первые строки",RowNumbers[Индекс]{0})

В результате у вас набор строк, ограниченный строкой заголовка и необработанными данными, которые вам нужны:

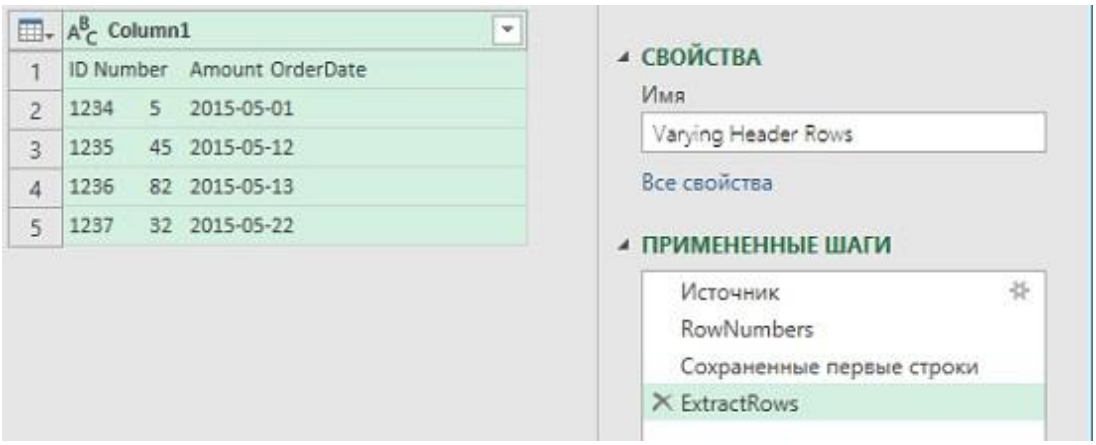


Рис. 20.20. Вы использовали код, чтобы извлечь динамический диапазон

Вы можете объединить последние два шага, что еще больше повысит читаемость кода:

```
let
    Источник = Table.FromColumns({Lines.FromBinary(
        File.Contents("D:\\Varying Header Rows.txt"), null, null, 1251)}),
    RowNumbers = Table.SelectRows
        (
            Table.AddIndexColumn(Источник, "Индекс", 0, 1),
            each Text.StartsWith
            (
            [Column1], "ID Number#(tab)Amount#(tab)OrderDate")
            or Text.StartsWith([Column1], "ID Total#(tab)164"
            )
        ),
    ExtractRows = Table.Skip(
        Table.FirstN(Источник,RowNumbers[Индекс]{1}),
        RowNumbers[Индекс]{0})
in
    ExtractRows
```

Рис. 20.21. Динамический набор данных готов для финальной очистки

Преобразование –> Разделить столбец –> По разделителю –> Табуляция –> По каждому вхождению разделителя. Удалите шаг *Измененный тип*. Он автоматически устанавливает тип данных для всех столбцов – *Текст,* что неверно из-за наличия текстовых заголовков. *Главная –> Использовать первую строку в качестве заголовков*. Вот теперь автоматически добавленный шаг *Измененный тип* должен верно выполнить свою работу, установив для столбцов *ID Number* и *Amount* тип данных – *Целое число*, а для столбца *OrderDate* – *Дата*. Если этого не произошло, установите типы данных вручную:

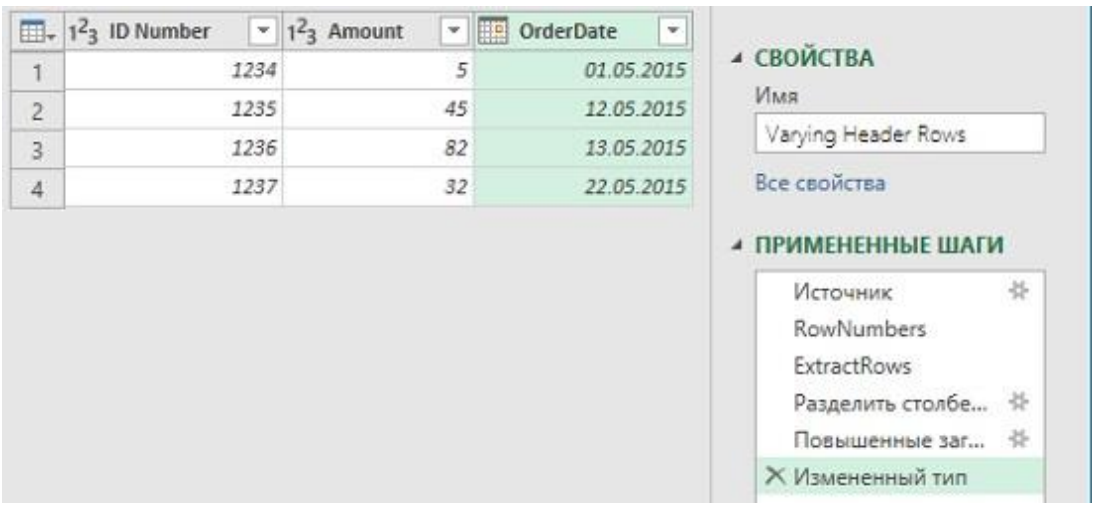


Рис. 20.22. Набор данных очищен и готов к использованию

Динамически ссылка на предыдущую строку

В начале главы мы сформулировали еще одну задачу для нашего примера – определение числа дней с момента предыдущего заказа. К сожалению, Power Query не имеет встроенной функции для выполнения этой задачи. Чтобы выполнить эту работу вручную, создадим новый столбец, который содержит дату предыдущего заказа. Вернитесь в редактор Power Query. Пройдите по меню *Добавление столбца –> Столбец индекса.* Переименуйте шаг *Добавлен индекс –> Transactions*:

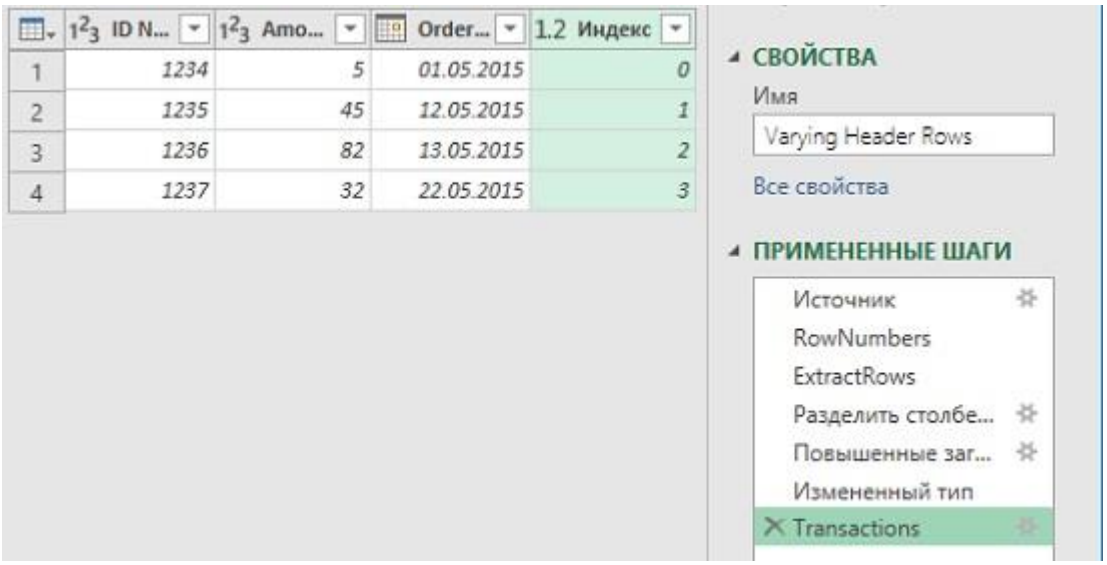


Рис. 20.23. Вы готовы построить формулу, извлекающую дату предыдущего заказа

Добавление столбца –> Настраиваемый столбец. Назовите его *PreviousOrder*. Введите формулу:

(1) =Transactions[OrderDate]{[Индекс]-1}

Вы ссылаетесь на шаг *Transactions* и хотите получить значение из столбца *OrderDate*. Последняя часть формулы – {[Индекс]-1} – заключена в фигурные скобки и не имеет префикса имени шага. Это означает, что ссылка относится к текущему шагу – *Добавлен пользовательский столбец*. Для каждой строки пользовательского столбца формула возьмет значение индекса строки и уменьшит его на 1. Это приведет к извлечению значения из столбца *OrderDate* предыдущей строки.

> Любопытно, что столбец *Индекс* нужен, чтобы указать Power Query, какую строку вы хотите получить из набора данных. Но совершенно необязательно, чтобы ссылка столбец *Индекс* присутствовала в формуле шага. Например, формула =#"Измененный тип"[OrderDate]{[Индекс]-1} также будет работать, несмотря на то, что на шаге *Измененный тип* еще не был создан столбце *Индекс*.

Результат не идеален:

	ID N...	Amo...	Order...	Индекс	PreviousOrder
1	1234	5	01.05.2015	0	Error
2	1235	45	12.05.2015	1	01.05.2015
3	1236	82	13.05.2015	2	12.05.2015
4	1237	32	22.05.2015	3	13.05.2015

Рис. 20.24. В первой строке возвращается ошибка

Это естественно, так как для первой строки не существует предыдущей. Чтобы исправить это, можно использовать оператор *try*, как описано в главе 18. Щелкните значок шестеренки рядом с шагом добавленным пользовательским шагом *Добавлен пользовательский столбец.* Измените формулу:

=try Transactions[OrderDate]{[Индекс]-1} otherwise [OrderDate]

	ID N...	Amo...	Order...	Индекс	PreviousOrder
1	1234	5	01.05.2015	0	01.05.2015
2	1235	45	12.05.2015	1	01.05.2015
3	1236	82	13.05.2015	2	12.05.2015
4	1237	32	22.05.2015	3	13.05.2015

Рис. 20.25. Теперь для первой строки значение *PreviousOrder* берется из текущей строки

Наконец вы можете рассчитать количество дней между двумя датами. Выберите столбец *OrderDate*, удерживая нажатой клавишу Ctrl, выберите столбец *PreviousOrder*. *Добавление столбца –> Дата –> Вычесть дни*.

	ID N...	Amo...	Order...	Индекс	PreviousOrder	Вычитание
1	1234	5	01.05.2015	0	01.05.2015	0
2	1235	45	12.05.2015	1	01.05.2015	11
3	1236	82	13.05.2015	2	12.05.2015	1
4	1237	32	22.05.2015	3	13.05.2015	9

Рис. 20.26. Рассчитана разница дат

> Напоминаем, что порядок выбора столбцов важен. Если вы сначала выберете столбец *PreviousOrder*, а затем столбец *OrderDate*, новый столбец покажет разницу *PreviousOrder* минус *OrderDate.*

Щелкните правой кнопкой мыши столбец Индекс –> Удалить. Щелкните правой кнопкой мыши столбец *PreviousOrder –> Тип изменения –> Дата*. Щелкните правой кнопкой мыши столбец *Вычитание –> Переименовать –> DaysSinceLastOrder*. *Главная –> Закрыть и загрузить*.

Можете протестировать работоспособность запроса, открыв файл *Varying Header Rows.txt* и добавив строки до *ID Number* и после *ID Total*. Далее обновите запрос в Excel-файле.