Крис Уэбб. Функция Csv.Document M Power Query

CSV-файлы часто используются в качестве источника данных для Power Query в Excel или Power BI Desktop. Однако документация Microsoft по функции *Csv.Document()* ограничена и в некоторых случаях неверна. В этой довольно длинной заметке я покажу вам столько возможностей этой функции, сколько мне удалось обнаружить.

> Это перевод заметки Криса Уэбба. Повествование ведется от лица Криса. Мои примечания набраны с отступом.

Синтаксис функции

```
Csv.Document(
   source as any,
   optional columns as any,
   optional delimiter as any,
   optional extraValues as nullable number,
   optional encoding as nullable number
) as table
```

> Оказалось, что заявленный в документации синтаксис не является единственно возможным. Второй вариант связан с использованием записи во втором аргументе:
>
> ```
> Csv.Document(
>    source as any,
>    optional […] as record,
> ) as table
> ```

Функция *Csv.Document()* возвращает таблицу и имеет один обязательный и четыре необязательных параметра.

Параметр Source (Источник)

Это единственный обязательный параметр – файл в формате CSV. Обычно это двоичное значение, возвращаемое функцией *File.Contents*. Например, возьмем следующий простой CSV-файл без заголовков столбцов и с одной строкой данных:

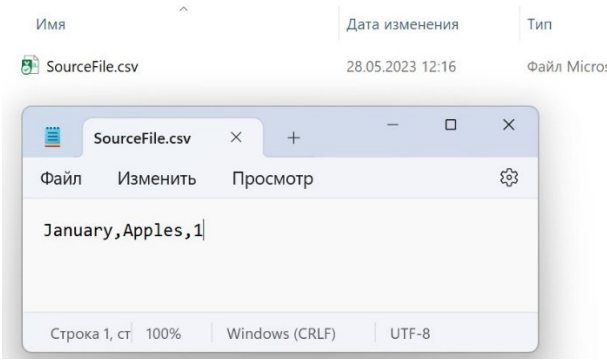


Рис. 1. Простой CSV-файл

Следующий код M использует *File.Contents* для чтения содержимого файла, а затем передает содержимое в *Csv.Document* для интерпретации как CSV-файла:

Запрос1[1]

```
let
   Источник = File.Contents("…\Files\SourceFile.csv"),
   B_CSV = Csv.Document(Источник)
in
   B_CSV
```

> Путь к CSV-файлу будет зависеть от того, где вы его разместите.

На выходе получается…

[1] Номер соответствует запросу в приложенном Excel-файле.

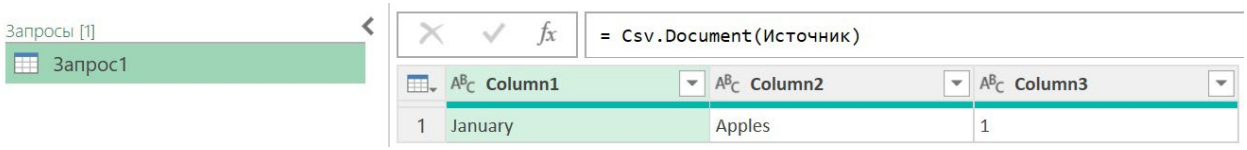


Рис. 2. Преобразование простого CSV-файла в таблицу

Также можно передать текст в первый параметр *Csv.Document()*, например:

Запрос2

```
let
   Источник = "February,Oranges,2",
   B_CSV = Csv.Document(Источник)
in
   B_CSV
```

Выходные данные этого запроса:

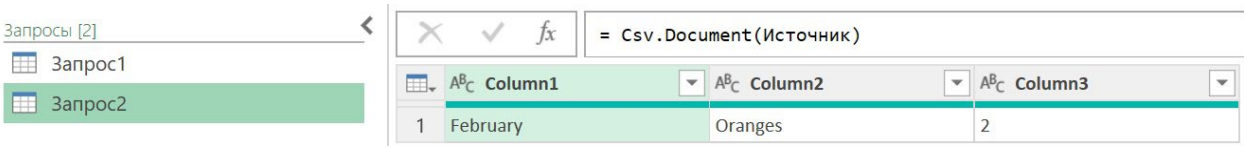


Рис. 3. Преобразование текста в таблицу

В обоих примерах я полагаюсь на поведение функции *Csv.Document()* по умолчанию в отношении разделителей и других свойств, которые я объясню более подробно ниже.

Использование записи во втором параметре

Второй параметр функции *Csv.Document()* можно использовать несколькими способами. В коде, созданном пользовательским интерфейсом редактора запросов, он обычно принимает форму записи, и различные поля записи указывают, как функция ведет себя в разных сценариях. Например, если вы подключитесь к CSV-файлу, показанному выше (см. рис. 1), пройдя в Excel *Данные –>Из текстового/CSV-файла*, то в пользовательском интерфейсе редактора запросов, вы увидите следующее окно с предварительным просмотром данных и тремя опциями для выбора:

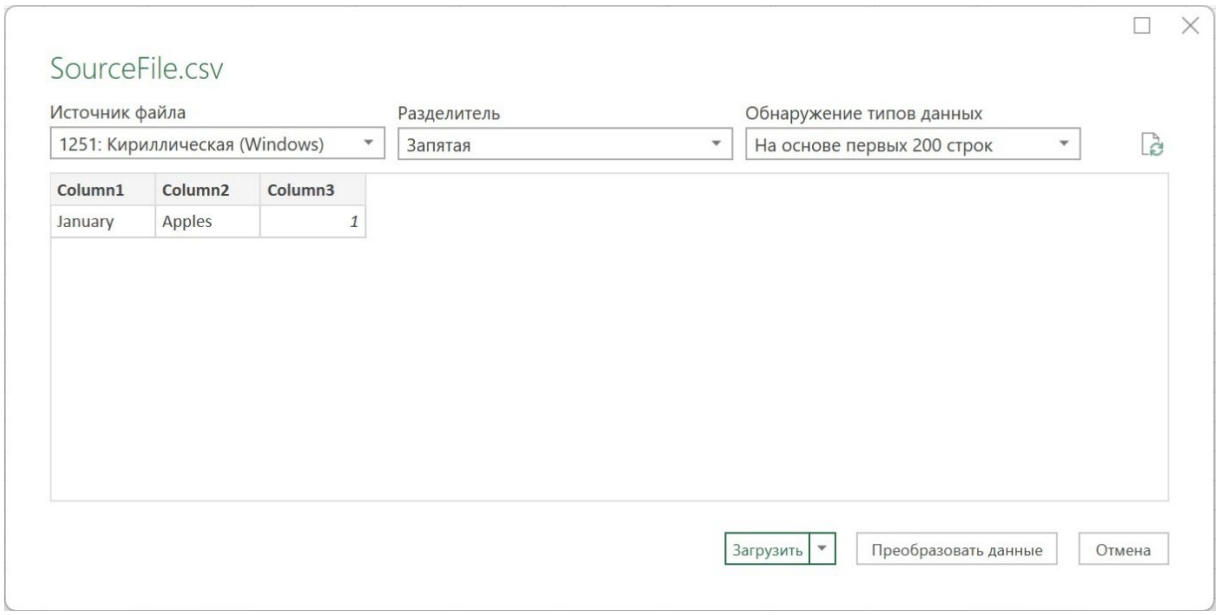


Рис. 4. Окно импорта CSV-файла

Если нажать *Преобразовать данные*, то в Расширенном редакторе вы увидите код:

Запрос3

```
let
   Источник = Csv.Document(
      File.Contents("…\Files\SourceFile.csv"),
      [
```

```
        Delimiter=",",
        Columns=3,
        Encoding=1251,
        QuoteStyle=QuoteStyle.None
      ]
  ),
  #"Измененный тип" = Table.TransformColumnTypes(
    Источник,
    {
      {"Column1", type text},
      {"Column2", type text},
      {"Column3", Int64.Type}
    }
  )
in
  #"Измененный тип"
```

Функция *Csv.Document()* во втором параметре представлена записью, содержащей четыре поля: *Delimiter*, *Columns*, *Encoding* и *QuoteStyle*. Существует также пятое поле, которое можно добавить в запись – *CsvStyle*, но его нельзя задать в пользовательском интерфейсе.

Опция *Обнаружение типов данных*, показанная на рис. 4, дает три варианта обнаружения типов данных в каждом столбце вашего файла. По умолчанию просматриваются первые 200 строк в наборе данных, но вы также можете попросить движок просмотреть весь набор данных (что может замедлить импорт) или вообще не определять типы данных. В последнем случае все столбцы будет рассматриваться как текстовые. В этом случае типы данных задаются не в функции *Csv.Document()*, а на шаге #"Измененный тип" с использованием функции *Table.TransformColumnTypes()*. Что и было сделано движком в Запросе 3. Как мы увидим позже, вместо этого можно задать имена и типы столбцов за один шаг с помощью *Csv.Document()*.

> Выбор варианта *Обнаружение типов данных* не отражается в коде. Похоже, выбор влияет только на поведение движка. Например, при выборе опции *Не обнаруживать типы данных* движок автоматически не использует функцию Table.PromoteHeaders() для поднятия заголовков столбцов. Выбор варианта *Обнаружение типов данных* также недоступен, если повторно обратиться к опциям импорта, нажав на шестеренку рядом с шагом *Источник* (см. рис. 11).

Поле Encoding (Кодировка)

Раскрывающееся меню *Источник файла* соответствует полю *Encoding* функции *Csv.Document()*. Это целочисленное значение указывает кодовую страницу, используемую для кодирования содержимого файла. По умолчанию 65001 (UTF-8).

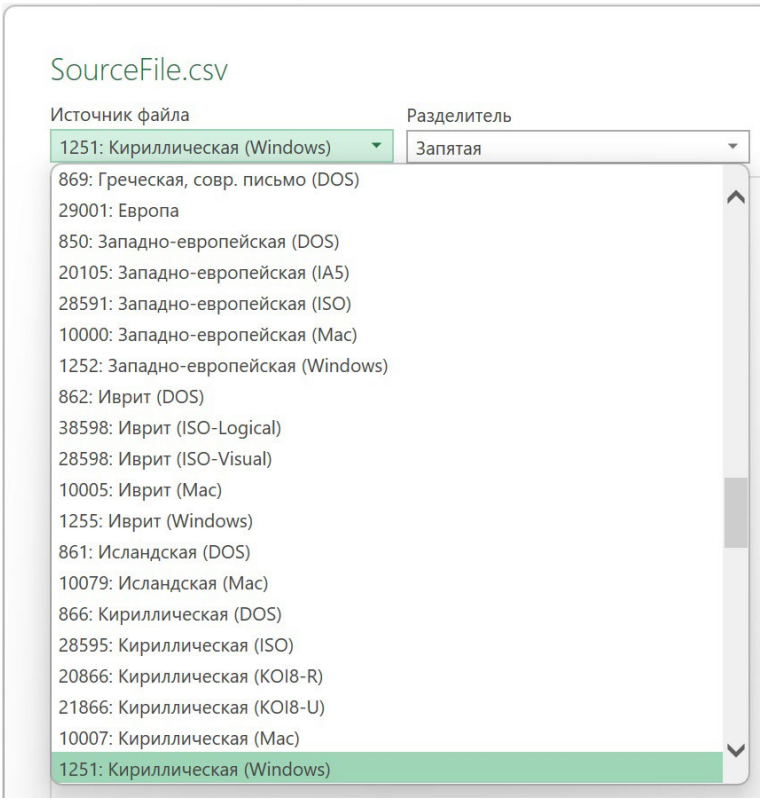


Рис. 5. Кодировки CSV-файла

В Запросе 3 кодовая страница 1251 выбрана движком. Кодировка распознана верно. Следующий код задает неверную кодовую страницу 1200 для того же CSV-файла (для упрощения иные поля записи во втором параметре опущены)…

Запрос4

```
let
   Источник = Csv.Document(
      File.Contents("…\Files\SourceFile.csv"),
      [Encoding=1200]
   )
in
   Источник
```

... со следующим результатом:

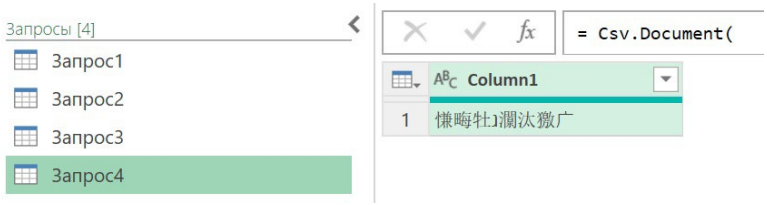


Рис. 6. Итог запроса при неверно заданной кодировке

Поле Delimiter (Разделитель)

Раскрывающийся список *Delimiter* позволяет указать разделитель, используемый для разделения на столбцы в каждой строке данных. В пользовательском интерфейсе доступен ряд параметров, а выбор *Пользовательский* даст возможность ввести собственный разделитель:

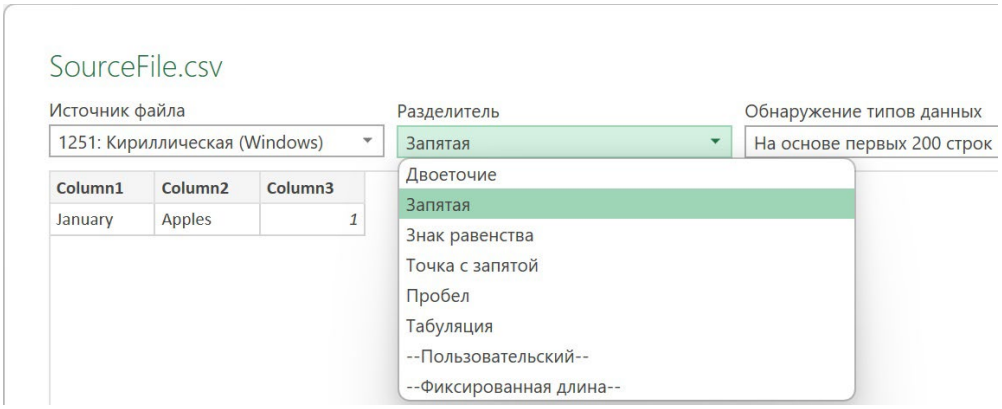


Рис. 7. Разделители в интерфейсе Power Query

Если на этом этапе выбрать разделитель из списка или выбрать *Пользовательский* и указать разделитель из одного символа, то поле *Delimiter* устанавливается в рамках записи во втором параметре *Csv.Document().* Если же после выбора *Пользовательский* указать разделитель из нескольких символов, или выбрать *Фиксированная длина*, будет использован иной синтаксис, описанный ниже.

Если *Delimiter* не установлен, по умолчанию используется запятая. Если вы хотите указать специальный символ, воспользуйтесь Escape-последовательностью. Например, чтобы применить в качестве разделителя символ табуляции, укажите текстовую строку "#(tab)". Движок вернет текстовое значение, содержащее только один символ табуляции.

Запрос5

```
let
   Источник = "789#(tab)456#(tab)123",
   B_CSV = Csv.Document(Источник, [Delimiter="#(tab)"])
in
   B_CSV
```

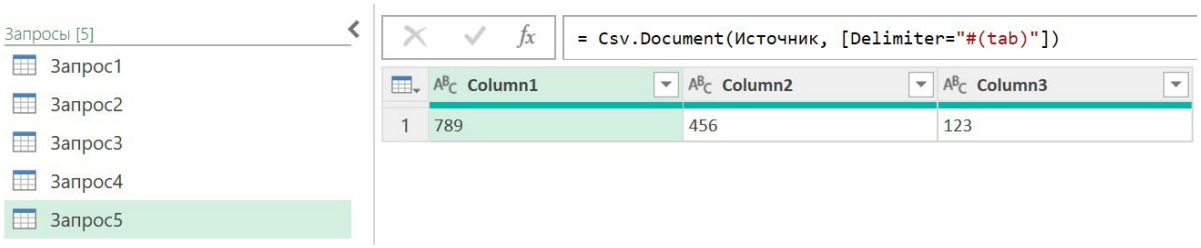


Рис. 8. Escape-последовательность #(tab) в качестве разделителя

Поле Columns (Столбцы)

В поле *Columns* указывается количество столбцов в таблице, возвращаемой функцией *Csv.Document()*, независимо от того, на сколько столбцов могут быть разделены исходные данные. Следующий запрос возвращает таблицу с четырьмя столбцами:

Запрос6

```
let
   Источник = "a,b,c",
   B_CSV = Csv.Document(Source, [Columns=4])
in
   B_CSV
```

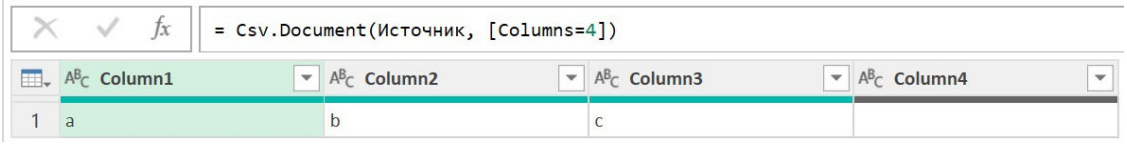


Рис. 9. Число столбцов определяется не исходными данными, а задается в коде

Следующий запрос возвращает таблицу с двумя столбцами, отбрасывая третий столбец, присутствующий в данных:

Запрос7
```
let
   Источник = "a,b,c",
   B_CSV = Csv.Document(Source, [Columns=2])
in
   B_CSV
```

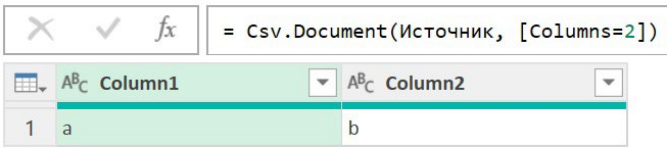


Рис. 10. Установленное число столбцов может вместить не все данные

Поле *Columns* при первом подключении к CSV-файлу через пользовательский интерфейс задать нельзя. Интерфейс укажет в коде столько столбцов, сколько найдет в CSV-файле. Это может стать проблемой, если количество столбцов в исходных данных в будущем увеличится. Однажды установленное фиксированное значение (при первом импорте) определит размер таблицы при последующих обновлениях. В результате некоторые столбцы в правой части таблицы не будут отображаться. Возможно, лучше удалить поле *Columns*, чтобы избежать этого. Если поле *Columns* не задано, то *Csv.Document()* возвращает таблицу с количеством столбцов, присутствующих в первой строке исходных данных.

> Как заметил buchlotnik, число столбцов не обязательно константа – его можно и вычислить. Например…
>
> **Запрос7a**
> ```
> let
>    Источник = {{"q"},{"a","s"},{"z","x","c"}},
>    B_CSV = Table.FromList(
>       Источник,
>       (x)=>x,
>       List.Max(Источник,null,List.Count)
>    )
> in
>    B_CSV
> ```
>
> … вернет таблицу с 4 столбцами – максимальное значение, на которое может быть разбита третья строка.

Поле QuoteStyle

Поле *QuoteStyle* может принимать два значения типа *QuoteStyle.Type*: *QuoteStyle.None* и *QuoteStyle.Csv*. Вот что говорится в документации о типе *QuoteStyle.Type*:

Определяет обработку разрывов строк в кавычках. QuoteStyle.None (по умолчанию): все разрывы строк рассматриваются как конец текущей строки, даже если они находятся внутри значения в кавычках. QuoteStyle.Csv: разрывы строк в кавычках рассматриваются как часть данных, а не как конец текущей строки.

Хотя значение *QuoteStyle.Type* задается автоматически при подключении к файлу, при редактировании шага в редакторе запросов, это значение можно изменить в пользовательском интерфейсе в раскрывающемся списке *Разрывы строк*:

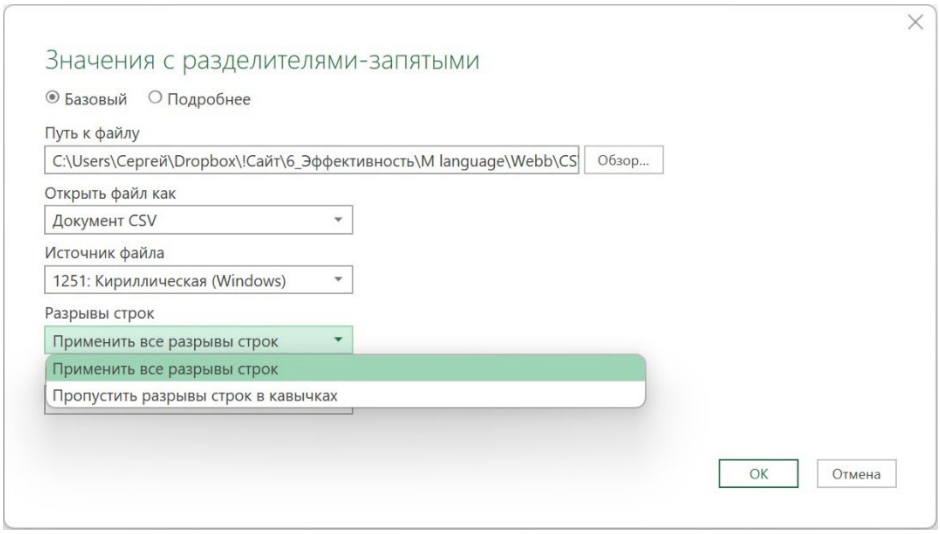


Рис. 11. Выбор значения *QuoteStyle.Type* в редакторе запросов

Это поле определяет, соблюдаются ли разрывы строк внутри текстовых значений. Как для *QuoteStyle.None*, так и для *QuoteStyle.Csv*, если текстовое значение заключено в двойные кавычки ", эти двойные кавычки используются для указания начала и конца текстового значения и не отображаются в выходных данных. Если вы хотите, чтобы двойная кавычка присутствовала в тексте, вы должны экранировать ее "". Однако, для *QuoteStyle.None* разрывы строк всегда соблюдаются, даже если они появляются в двойных кавычках. Если задан *QuoteStyle.Csv*, то разрывы строк внутри двойных кавычек игнорируются. Рассмотрим следующий CSV-файл:

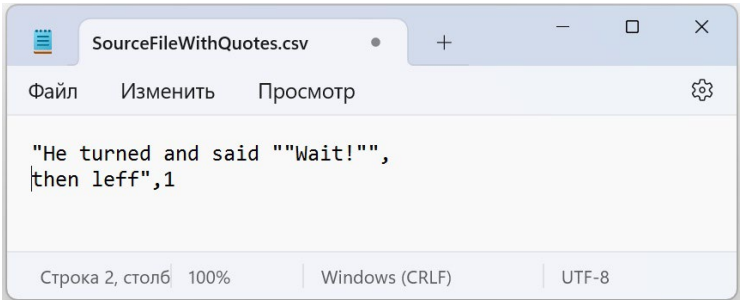


Рис. 12. CSV-файл с экранированными кавычками

Код с использованием *QuoteStyle.None*…

Запрос8

```
let
   Источник = File.Contents("…\Files\SourceFileWithQuotes.csv"),
   B_CSV = Csv.Document(
      Источник,
      [QuoteStyle=QuoteStyle.None]
   )
in
   B_CSV
```

... возвращает таблицу с двумя строками и одним столбцом:

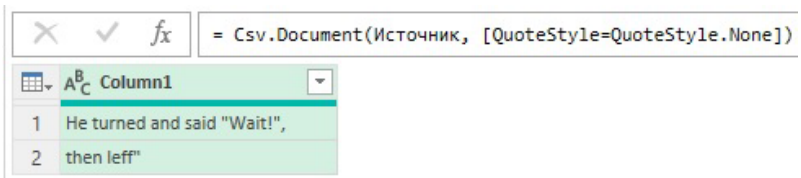


Рис. 13. Таблица с параметром *QuoteStyle.None*

А код с использованием *QuoteStyle.Csv*…

Запрос9

```
let
   Источник = File.Contents("…\Files\SourceFileWithQuotes.csv"),
   B_CSV = Csv.Document(
      Источник,
      [QuoteStyle=QuoteStyle.Csv]
   )
in
   B_CSV
```

... возвращает таблицу из одной строки и двух столбцов:

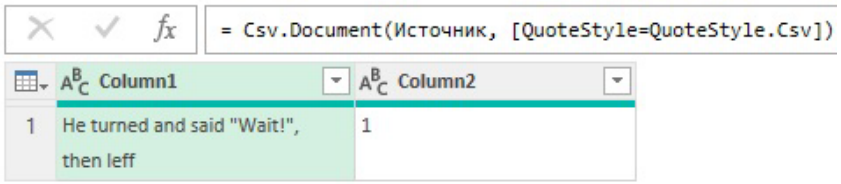


Рис. 14. Таблица с параметром *QuoteStyle.Csv*

Поле CsvStyle

В онлайн справке редактора Power Query сказано:

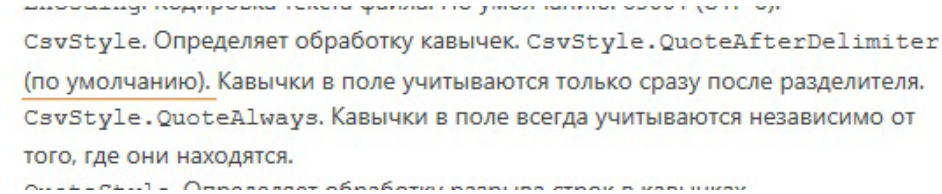


Рис. 15. Фрагмент справки редактора Power Query по функции *Csv.Document*

На самом деле по умолчанию используется значение *CsvStyle.QuoteAlways*.

Импортируем следующий CSV-файл:

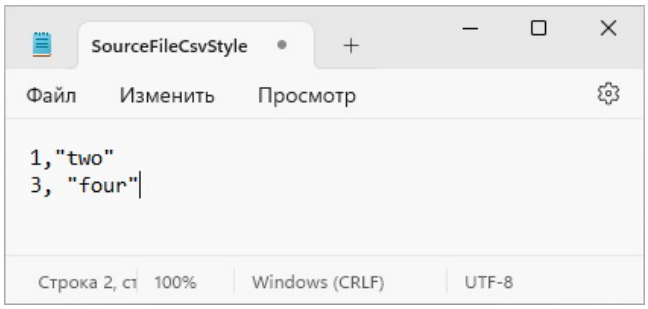


Рис. 16. Пример CSV-файла

Обратите внимание, что во второй строке после запятой стоит пробел. Следующий запрос M…

Запрос10

```
let
   Источник = File.Contents("…\Files\SourceFileCsvStyle.csv"),
   B_CSV = Csv.Document(
      Источник,
      [CsvStyle=CsvStyle.QuoteAlways])
in
   B_CSV
```

… справляется с кавычками, так как пробел после запятой не считается значимым:

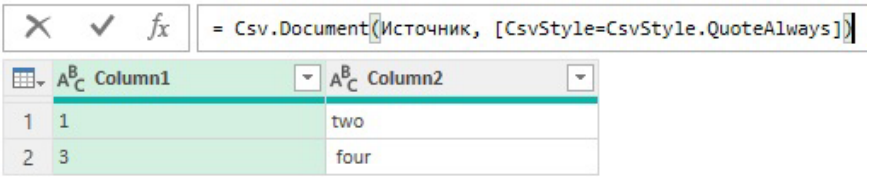


Рис. 17. Таблица с параметром CsvStyle.QuoteAlways

С другой стороны…

Запрос11

```
let
   Источник = File.Contents("…\Files\SourceFileCsvStyle.csv"),
   B_CSV = Csv.Document(
      Источник,
      [CsvStyle=CsvStyle.QuoteAfterDelimiter])
in
   B_CSV
```

… возвращает текст во второй строке в кавычках, так как пробел после запятой изменяет способ обработки кавычек:

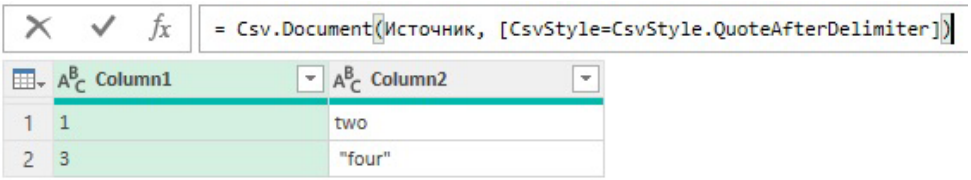


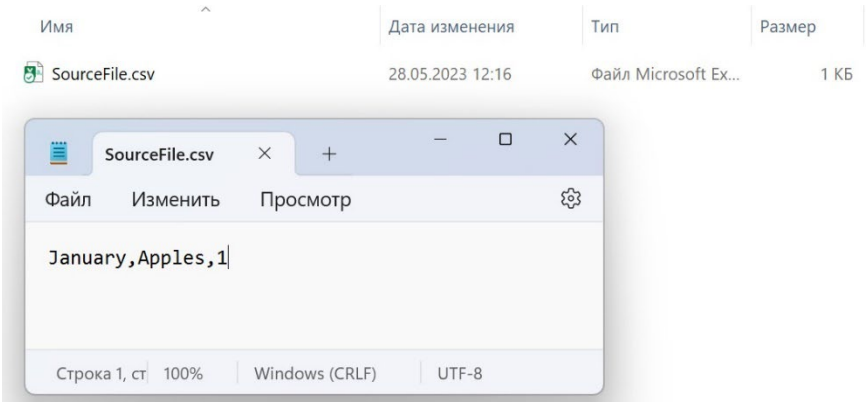
Рис. 18. Таблица с параметром CsvStyle.QuoteAfterDelimiter

Использование списка или table type во втором параметре

Если первая строка CSV-файла содержит заголовки столбцов и вы подключаетесь к файлу с помощью пользовательского интерфейса редактора PQ, в большинстве случаев наличие заголовков будет обнаружено. В запрос будет добавлен шаг с функцией Table.PromoteHeaders() для использования значений в первой строке в качестве заголовков столбцов. Однако, если у вас нет заголовков столбцов в CSV-файле, вместо записи во втором параметре функции *Csv.Document()* можно указать список имен столбцов или, что еще лучше, тип таблицы для одновременного задания имен и типов столбцов.

Если задать имена столбцов *списком* или *table type*, то можно использовать еще три опциональных параметра внутри функции *Csv.Document()*: *Delimiter, ExtraValues* и *Encoding*.

Например, в следующем CSV-файле есть три столбца: *Month*, *Product* и *Sales*.



Использование списка текстовых значений, содержащих эти имена столбцов, во втором параметре *Csv.Document()*, как в следующем запросе M…

Запрос12

```
let
   Источник = File.Contents("…\Files\SourceFile.csv"),
   B_CSV = Csv.Document(Источник, {"Month","Product","Sales"})
in
   B_CSV
```

… возвращает таблицу с заголовками:

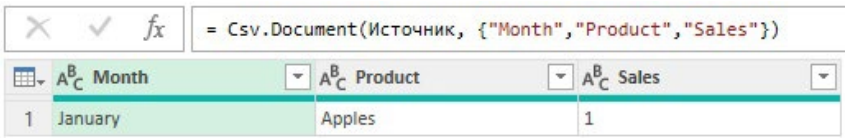


Рис. 19. Таблица с заголовками, заданными списком

Здесь имена столбцов заданы правильно, но типы данных трех столбцов установлены по умолчанию, как текст. Использование *table type* вместо списка имен столбцов позволяет решить и эту проблему:

Запрос13

```
let
   Источник = File.Contents("…\Files\SourceFile.csv"),
   B_CSV = Csv.Document(
      Источник ,
      type table [Month=text, Product=text, Sales=number]
   )
in
   B_CSV
```

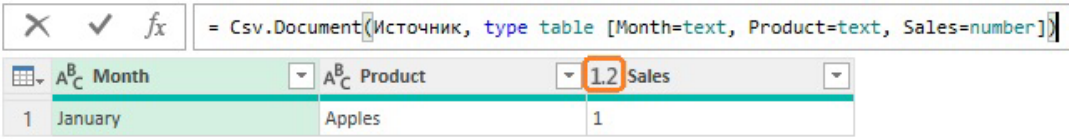


Рис. 20. Таблица с заголовками, заданными *table type*

> Обратите внимание, что теперь в столбце Sales тип данных = *Десятичное число*. Если имена столбцом состоят из нескольких слов используется идентификатор с кавычками. Например:
>
> type table [Month=text, Product=text, #"Total Sales"=number]

Параметр Delimiter (Разделитель)

Если вы использовали список имен столбцов или тип таблицы во втором параметре *Csv.Document()*, вы можете добавить третий параметр для управления тем, как каждая строка данных разбивается на столбцы. Это можно сделать двумя способами.

Прежде всего, вы можете передать любой фрагмент текста в третий параметр, чтобы указать разделитель. Это может быть один или несколько символов. Например, следующий запрос M…

Запрос14

```
let
   Источник = "abcdefg",
   B_CSV = Csv.Document(Source,{"first","second"},"c")
in
   B_CSV
```

… возвращает:

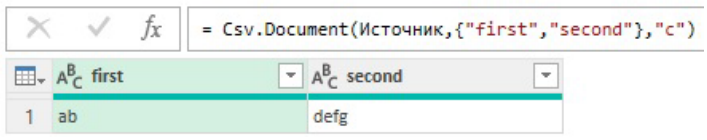


Рис. 21. Разделитель из одного символа – *c*

Следующий запрос…

Запрос15

```
let
   Источник = "abcdefg",
   B_CSV = Csv.Document(Source,{"first","second"},"cd")
in
```

B_CSV

… возвращает:

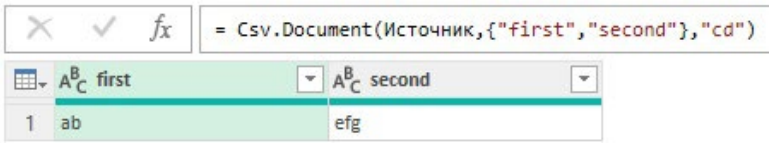


Рис. 22. Разделитель из двух символов – *cd*

Параметр *Delimiter* также может принимать список целочисленных значений, чтобы можно было обрабатывать строки с подстроками фиксированной ширины. Эта функция доступна в пользовательском интерфейсе при выборе параметра *Фиксированная ширина* в раскрывающемся списке *Разделитель* при первом подключении к CSV-файлу:

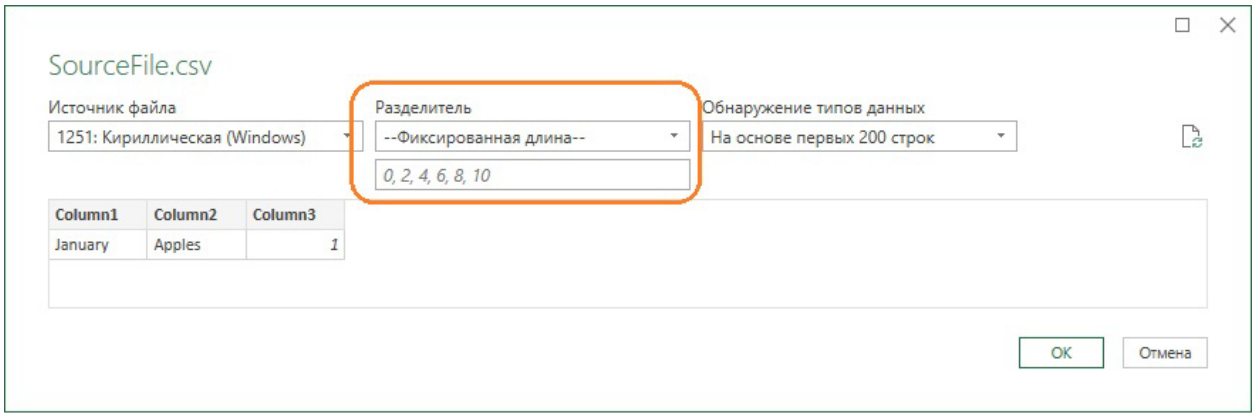


Рис. 23. Разделитель, как список целочисленных значений

Каждое целое число в списке представляет собой количество символов от начала строки, которое отмечает начало каждого столбца. Каждое целое число в списке должно быть больше предыдущего. Помните, что счет начинается с 0. Например, запрос M…

Запрос16

```
let
   Источник = "abcdefg",
   B_CSV = Csv.Document(
      Источник,
      {"first","second","third"},
      {0,3,5}
   )
in
   B_CSV
```

… вернет:

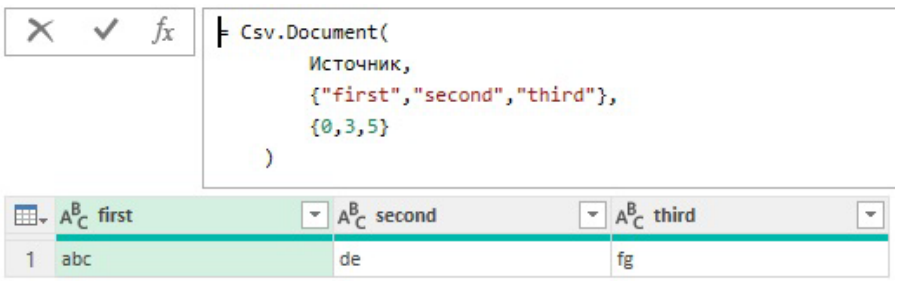


Рис. 24. Таблица на основе разделителя в виде списка целочисленных значений

Параметр ExtraValues

Параметр *ExtraValues* позволяет обрабатывать сценарии, в которых в конце строк есть дополнительные столбцы. Однако это не так полезно, как кажется: в большинстве случаев, когда количество столбцов в CSV-файле меняется, это происходит из-за того, что в текстовых столбцах

есть разрывы строк без кавычек, и в этом случае вы должны убедиться, что ваши исходные данные всегда переносят текст в двойные кавычки, и использовать опцию *QuoteStyle*, описанную выше. Если вы не можете исправить источник данных, см. этот пост.

Параметр *ExtraValues* может принимать одно из трех значений типа *ExtraValues.Type*: *ExtraValues.List*, *ExtraValues.Ignore* и *ExtraValues.Error* (по умолчанию).

Рассмотрим…

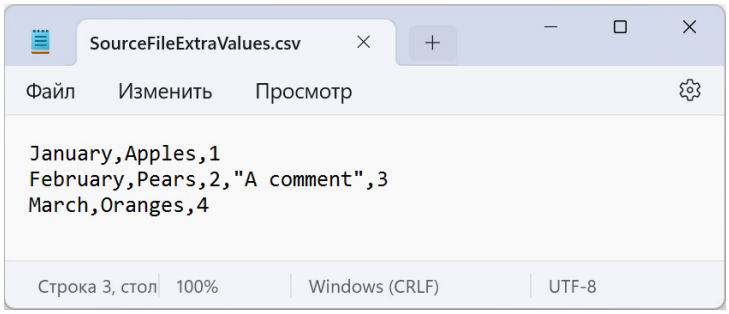


Рис. 25. CSV-файл с двумя дополнительными столбцами во второй строке

Следующий запрос считывает данные из файла SourceFileExtraValue.csv:

Запрос17

```
let
   Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),
   B_CSV = Csv.Document(
      Источник,
      {"Month","Product","Sales"}
   )
in
   B_CSV
```

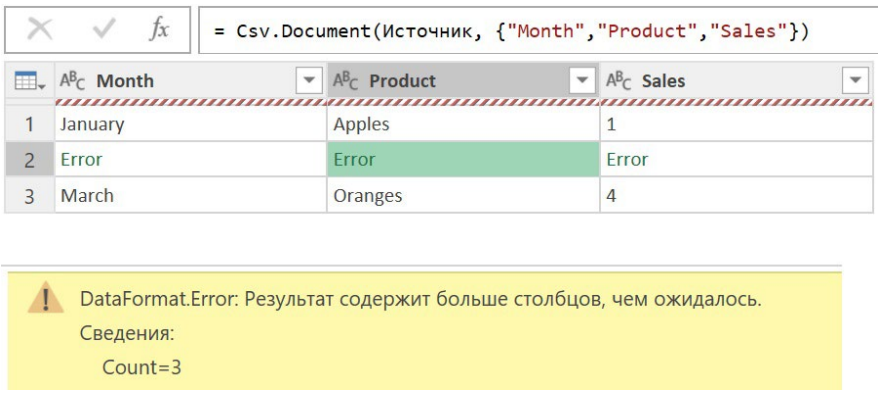


Рис. 26. Во второй строке отображаются ошибки

Поскольку мы указали, что в таблице три столбца, для каждой ячейки во второй строке возвращается ошибка *Результат содержит больше столбцов, чем ожидалось*.

Аналогичный результат будет получен, если явно указать четвертым параметром *ExtraValues.Error*:

Запрос18

```
let
   Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),
   B_CSV = Csv.Document(
      Источник,
      {"Month","Product","Sales"},
      ",",
      ExtraValues.Error
   )
in
   B_CSV
```

Однако, если вместо этого вы установите *ExtraValues.Ignore*…

Запрос19

```
let
   Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),
   B_CSV = Csv.Document(
      Источник,
      {"Month","Product","Sales"},
      ",",
      ExtraValues.Ignore
   )
in
   B_CSV
```

… лишние данные будут проигнорированы, и ошибка не появится:

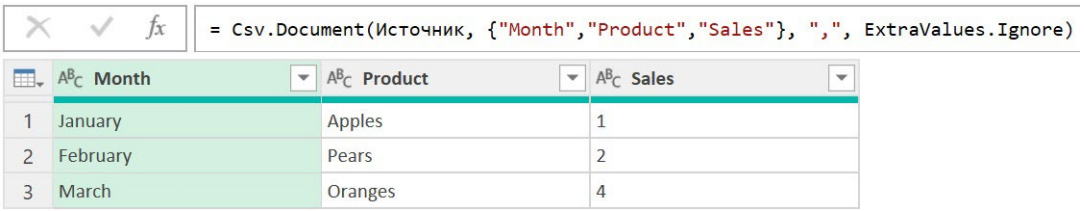


Рис. 27. Усечение данных при использовании параметра *ExtraValues.Ignore*

ExtraValues.List позволяет поместить дополнительные значения в список. Вам нужно предусмотреть дополнительный столбец в таблице для хранения этих значений. Обратите внимание, что в следующем запросе определены четыре столбца, а не три:

Запрос20

```
let
   Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),
   B_CSV = Csv.Document(
      Источник,
      {"Month","Product","Sales","Extra Columns"},
      ",",
      ExtraValues.List
   )
in
   B_CSV
```

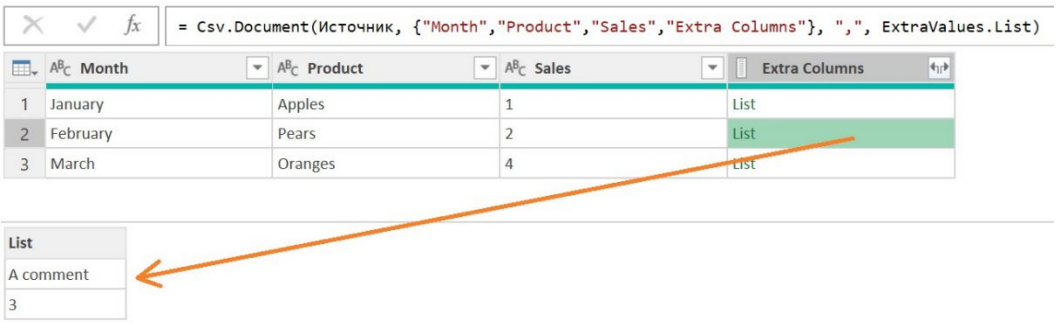


Рис. 28. Помещение избыточных данных в список в дополнительном столбце

В первой и третьей строках столбец *Extra Columns* содержит пустые списки. Во второй строке список содержит два значения.

Параметр Encoding

Параметр *Encoding* напрямую соответствует полю Encoding, используемому при передаче записи во второй параметр, как описано ранее. Единственное отличие состоит в том, что здесь *Encoding* помимо целых чисел может принимать значение типа *TextEncoding.Type:*

Name	Value
TextEncoding.Ascii	20127
TextEncoding.BigEndianUnicode	1201
TextEncoding.Type	[Type]
TextEncoding.Unicode	1200
TextEncoding.Utf16	1200
TextEncoding.Utf8	65001
TextEncoding.Windows	1252

Рис. 29. Поддерживаемые значения типа *TextEncoding.Type*

Единственная причина использовать *TextEncoding.Type* – удобство чтения кода M. Следующие два запроса возвращают одну и туже таблицу:

Запрос21

```
let
   Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),
   B_CSV = Csv.Document(
      Источник,
      {"Month","Product","Sales"},
      ",",
      ExtraValues.List,
      TextEncoding.Windows
   )
in
   B_CSV
```

Запрос22

```
let
   Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),
   B_CSV = Csv.Document(
      Источник,
      {"Month","Product","Sales"},
      ",",
      ExtraValues.List,
      1252
   )
in
   B_CSV
```

А как насчет *CsvStyle* и *QuoteStyle*? Если во втором параметре *Csv.Document()* указан список имен столбцов или тип таблицы, задать *CsvStyle* или *QuoteStyle* невозможно. При этом вы будете наблюдать поведение движка, характерное для *CsvStyle.QuoteAlways* и *QuoteStyle.Csv*. Например, со следующими исходными данными…

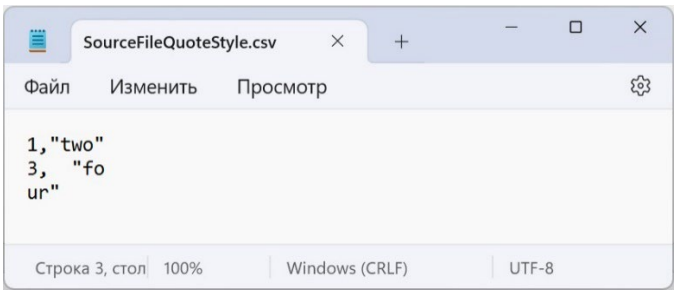


Рис. 30. CSV-файл для исследования поведения, когда во втором параметре указан список имен столбцов

… запрос M…

Запрос23

```
let
   Источник = File.Contents("…\Files\SourceFileQuoteStyle.csv"),
```

```
    B_CSV = Csv.Document(
        Источник,
        {"Month","Sales"},
        ",",
        ExtraValues.Ignore,
        1252
    )
in
    B_CSV
```

… вернет:

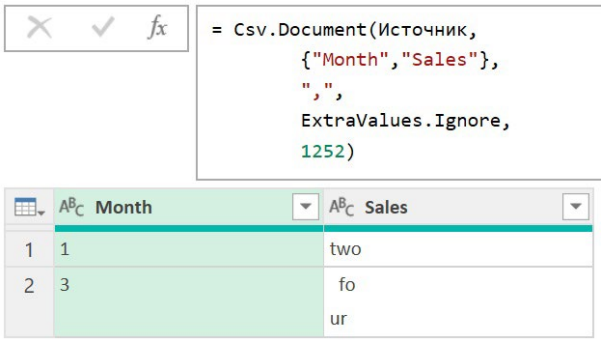


Рис. 31. Поведение движка соответствует параметрам *CsvStyle.QuoteAlways* и *QuoteStyle.Csv*

Сравните 4 варианта поведения, когда параметрам *CsvStyle.QuoteAlways* и *QuoteStyle.Csv* мы можем явно управлять в записи:

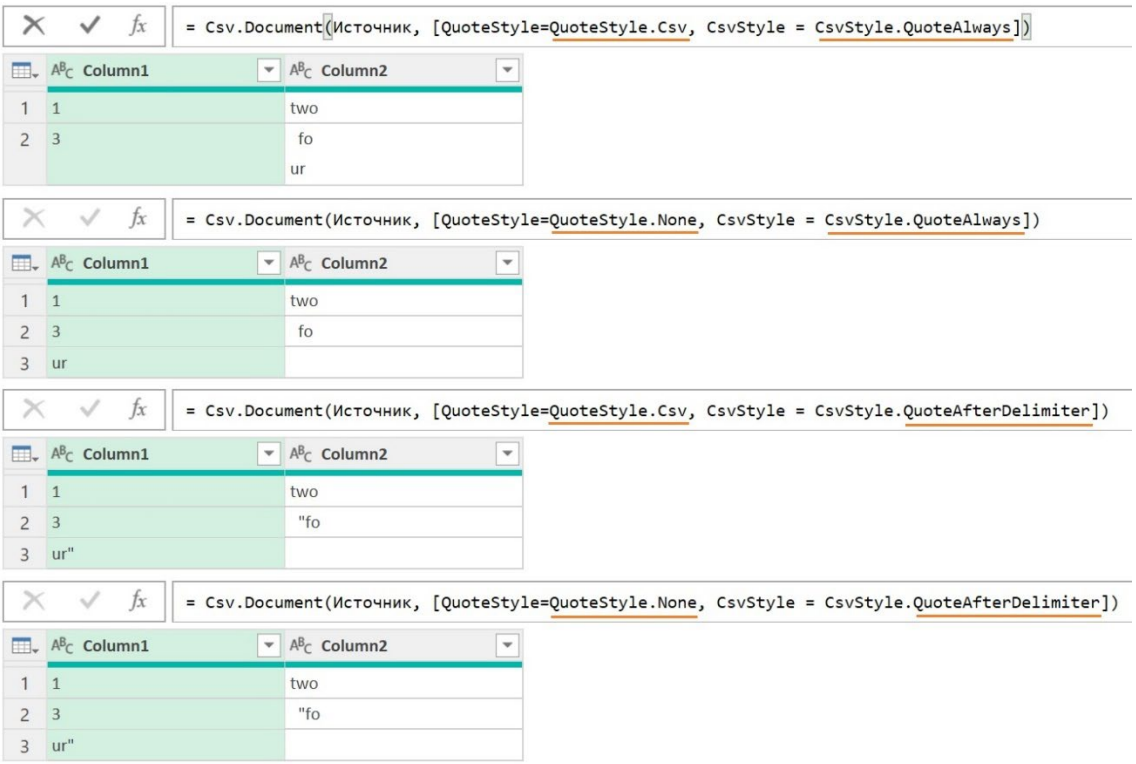


Рис. 32. Четыре сочетания параметров *CsvStyle.Type* и *QuoteStyle.Type*