## 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](https://learn.microsoft.com/ru-ru/powerquery-m/csv-document) 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](https://blog.crossjoin.co.uk/2018/03/09/an-in-depth-look-at-the-csv-document-m-function/) 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:

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

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]](#footnote-1)

let

Источник = File.Contents("…\Files\SourceFile.csv"),

В\_CSV = Csv.Document(Источник)

in

В\_CSV

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

На выходе получается…

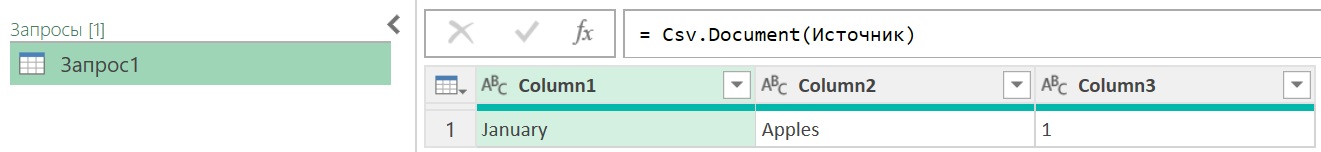


Рис. 2. Преобразование простого CSV-файла в таблицу

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

**Запрос2**

let

Источник = "February,Oranges,2",

В\_CSV = Csv.Document(Источник)

in

В\_CSV

Выходные данные этого запроса:

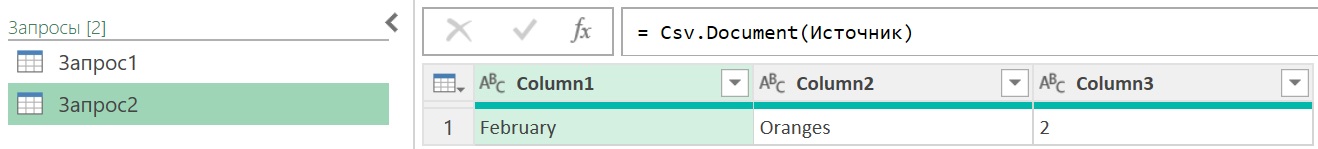


Рис. 3. Преобразование текста в таблицу

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:

Изображение выглядит как текст, снимок экрана, число

Автоматически созданное описание

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).

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

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:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

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:

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

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](https://learn.microsoft.com/ru-ru/powerquery-m/m-spec-lexical-structure#character-escape-sequences). 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",

В\_CSV = Csv.Document(Источник, [Delimiter="#(tab)"])

in

В\_CSV

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

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",

В\_CSV = Csv.Document(Source, [Columns=4])

in

В\_CSV

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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",

В\_CSV = Csv.Document(Source, [Columns=2])

in

В\_CSV

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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](https://t.me/PQ_ru/24316) buchlotnik, число столбцов не обязательно константа – его можно и вычислить. Например…

**Запрос7а**

let

Источник = {{"q"},{"a","s"},{"z","x","c"}},

В\_CSV = Table.FromList(

Источник,

(x)=>x,

List.Max(Источник,null,List.Count)

)

in

В\_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*:

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

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:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рис. 12. CSV-файл с экранированными кавычками

Код с использованием *QuoteStyle.None*…

**Запрос8**

let

Источник = File.Contents("…\Files\SourceFileWithQuotes.csv"),

В\_CSV = Csv.Document(

Источник,

[QuoteStyle=QuoteStyle.None]

)

in

В\_CSV

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

Изображение выглядит как текст, Шрифт, линия, снимок экрана

Автоматически созданное описание

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

А код с использованием *QuoteStyle.Csv*…

**Запрос9**

let

Источник = File.Contents("…\Files\SourceFileWithQuotes.csv"),

В\_CSV = Csv.Document(

Источник,

[QuoteStyle=QuoteStyle.Csv]

)

in

В\_CSV

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

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

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

#### Поле CsvStyle

В онлайн справке редактора Power Query сказано:

Изображение выглядит как текст, снимок экрана, Шрифт, алгебра

Автоматически созданное описание

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

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

Импортируем следующий CSV-файл:

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

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"),

В\_CSV = Csv.Document(

Источник,

[CsvStyle=CsvStyle.QuoteAlways])

in

В\_CSV

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

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

Рис. 17. Таблица с параметром CsvStyle.QuoteAlways

С другой стороны…

**Запрос11**

let

Источник = File.Contents("…\Files\SourceFileCsvStyle.csv"),

В\_CSV = Csv.Document(

Источник,

[CsvStyle=CsvStyle.QuoteAfterDelimiter])

in

В\_CSV

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

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

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](https://learn.microsoft.com/ru-ru/powerquery-m/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*.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

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"),

В\_CSV = Csv.Document(Источник, {"Month","Product","Sales"})

in

В\_CSV

… возвращает таблицу с заголовками:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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"),

В\_CSV = Csv.Document(

Источник ,

type table [Month=text, Product=text, Sales=number]

)

in

В\_CSV

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

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",

В\_CSV = Csv.Document(Source,{"first","second"},"c")

in

В\_CSV

… возвращает:

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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

Следующий запрос…

**Запрос15**

let

Источник = "abcdefg",

В\_CSV = Csv.Document(Source,{"first","second"},"cd")

in

В\_CSV

… возвращает:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

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

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:

Изображение выглядит как текст, линия, снимок экрана

Автоматически созданное описание

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",

В\_CSV = Csv.Document(

Источник,

{"first","second","third"},

{0,3,5}

)

in

В\_CSV

… вернет:

Изображение выглядит как текст, Шрифт, линия, число

Автоматически созданное описание

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](https://blog.crossjoin.co.uk/2016/02/16/working-with-csv-files-that-contain-rogue-line-breaks-in-power-query-and-power-bi/).

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

Рассмотрим…

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

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"),

В\_CSV = Csv.Document(

Источник,

{"Month","Product","Sales"}

)

in

В\_CSV

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

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"),

В\_CSV = Csv.Document(

Источник,

{"Month","Product","Sales"},

",",

ExtraValues.Error

)

in

В\_CSV

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

**Запрос19**

let

Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),

В\_CSV = Csv.Document(

Источник,

{"Month","Product","Sales"},

",",

ExtraValues.Ignore

)

in

В\_CSV

… лишние данные будут проигнорированы, и ошибка не появится:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

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"),

В\_CSV = Csv.Document(

Источник,

{"Month","Product","Sales","Extra Columns"},

",",

ExtraValues.List

)

in

В\_CSV

Изображение выглядит как текст, снимок экрана, линия, число

Автоматически созданное описание

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:*

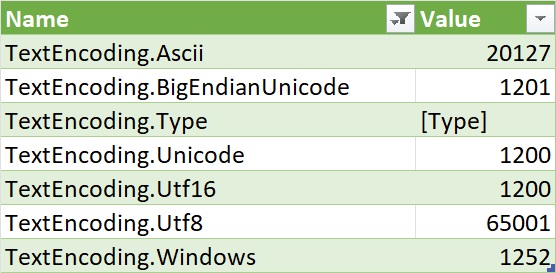


Рис. 29. Поддерживаемые значения типа *TextEncoding.Type*

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

**Запрос21**

let

Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),

В\_CSV = Csv.Document(

Источник,

{"Month","Product","Sales"},

",",

ExtraValues.List,

TextEncoding.Windows

)

in

В\_CSV

**Запрос22**

let

Источник = File.Contents("…\Files\SourceFileExtraValue.csv"),

В\_CSV = Csv.Document(

Источник,

{"Month","Product","Sales"},

",",

ExtraValues.List,

1252

)

in

В\_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…

Изображение выглядит как текст, снимок экрана, мультимедиа, программное обеспечение

Автоматически созданное описание

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"),

В\_CSV = Csv.Document(

Источник,

{"Month","Sales"},

",",

ExtraValues.Ignore,

1252

)

in

В\_CSV

… вернет:

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

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:

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

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

1. Номер соответствует запросу в приложенном Excel-файле. [↑](#footnote-ref-1)