Access

Access.Database

Access.Database(database as binary, optional options as record) as table

Описание

Возвращает структурное представление базы данных Access, `database`. Можно указать необязательный параметр записи `options` для управления следующими параметрами:

- `CreateNavigationProperties` : Логическое значение (true или false), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — false.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.

Пример указания параметра записи: [параметр1 = значение1, параметр2 = значение2...].

ActiveDirectory

ActiveDirectory.Domains

ActiveDirectory.Domains(optional forestRootDomainName as text) as table

Описание

Возвращает список доменов Active Directory в том же лесу, что и указанный домен, или из домена текущего компьютера, если значение не указано.

AdoDotNet

AdoDotNet.DataSource

AdoDotNet.DataSource(providerName as text, connectionString as any) as table

Описание

Возвращает коллекцию схем для источника данных ADO.NET с именем поставщика `providerName` и строкой подключения `connectionString`. `connectionString` может быть текстом или записью пар "свойство — значение". Значения свойств могут быть выражены текстом или числом.

AdoDotNet.Query

AdoDotNet.Query(providerName as text, connectionString as any, query as text) as table

Описание

Возвращает результат запуска "`query`" со строкой подключения "`connectionString`" с помощью поставщика ADO.NET `providerName`. "`connectionString`" может быть текстом или записью пар "значение-свойство". Значения свойств могут быть текстовыми или числовыми.

AnalysisServices

AnalysisServices.Database

AnalysisServices.Database(server as text, database as text, optional options as record) as table

Описание

Возвращает таблицу многомерных кубов или табличных моделей из базы данных служб Analysis Services `database` на сервере `server`. Можно указать необязательный параметр записи `options` для управления следующими параметрами:

- `Query` : Собственный запрос многомерных выражений для извлечения данных.
- `TypedMeasureColumns` : Логическое значение, указывающее, будут ли в качестве типов добавленных столбцов мер использоваться типы, указанные в многомерной или табличной модели. Если задано значение false, для всех столбцов мер будет использоваться тип "number". Значение по умолчанию — false.
- `Culture` : Имя языка и региональных параметров для данных. Это соответствует свойству строки подключения "Код языка".
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию зависит от драйвера.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `SubQueries` : Число (0, 1 или 2), задающее значение свойства "SubQueries" в строке подключения. Оно управляет поведением вычисляемых членов в подзапросах выборки или вложенных кубах. (Значение по умолчанию — 2).

AnalysisServices.Databases

AnalysisServices.Databases(server as text, optional options as record) as table

Описание

Возвращает базы данных в определенном экземпляре Analysis Services — `server`. Чтобы указать дополнительные свойства, можно предоставить необязательный параметр записи `options`. Запись может содержать следующие поля:

- `TypedMeasureColumns` : Логическое значение, указывающее, будут ли в качестве типов добавленных столбцов мер использоваться типы, указанные в многомерной или табличной модели. Если задано значение false, для всех столбцов мер будет использоваться тип "number". Значение по умолчанию — false.
- `Culture` : Имя языка и региональных параметров для данных. Это соответствует свойству строки подключения "Код языка".
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию зависит от драйвера.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `SubQueries` : Число (0, 1 или 2), задающее значение свойства "SubQueries" в строке подключения. Оно управляет поведением вычисляемых членов в подзапросах выборки или вложенных кубах. (Значение по умолчанию — 2).

AzureStorage

AzureStorage.Blobs

AzureStorage.Blobs(account as text) as table

Описание

Возвращает навигационную таблицу, содержащую одну строку для каждого контейнера, найденного по URL-адресу учетной записи `account` в хранилище Azure. Каждая строка содержит ссылку на BLOB-объекты в контейнере.

AzureStorage.Tables

AzureStorage.Tables(account as text) as table

Описание

Возвращает навигационную таблицу, содержащую одну строку для каждой таблицы, найденной по URL-адресу учетной записи `account` в хранилище Azure. Каждая строка содержит ссылку на таблицу Azure.

Binary

Binary.Buffer

Binary.Buffer(binary as binary) as binary

Описание

Помещает двоичное значение в буфер в памяти. Результат вызова - стабильное двоичное значение (то есть с детерминированной длиной и порядком байтов).

Пример №-1

Описание

Создать стабильную версию двоичного значения.

Код

```
Binary.Buffer(Binary.FromList({0..10}))
```

Результат

```
#binary({0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10})
```

Binary.Combine

Binary.Combine(binaries as list) as binary

Описание

Объединяет список двоичных значений в одно двоичное значение.

Binary.Compress

Binary.Compress(binary as binary, compressionType as number) as binary

Описание

Сжимает двоичное значение с помощью заданного типа сжатия. Результат такого вызова — сжатая копия входных данных. Типы сжатия:

- `Compression.GZip`
- `Compression.Deflate`

Пример №-1

Описание

Сжатие двоичного значения.

Код

```
Binary.Compress(Binary.FromList(List.Repeat({10}, 1000)), Compression.Deflate)
```

Результат

```
#binary({227, 226, 26, 5, 163, 96, 20, 12, 119, 0, 0})
```

Binary.Decompress

Binary.Decompress(binary as binary, compressionType as number) as binary

Описание

Распаковать двоичное значение с применением указанного типа сжатия. Результат этого вызова - распакованная копия входных данных. Типы сжатия:

- `Compression.GZip`
- `Compression.Deflate`

Пример №-1

Описание

Распаковать двоичное значение.

Код

```
Binary.Decompress(#binary({115, 103, 200, 7, 194, 20, 134, 36, 134, 74, 134, 84, 6,
0}), Compression.Deflate)
```

Результат

```
#binary({71, 0, 111, 0, 111, 0, 100, 0, 98, 0, 121, 0, 101, 0})
```

Binary.From

Binary.From(value as any, optional encoding as number) as binary

Описание

Возвращает значение `binary` из указанного `value`. Если данное `value` имеет значение `null`, `Binary.From` возвращает `null`. Если данное `value` - `binary`, возвращается `value`. Значения следующих типов можно преобразовать в значение `binary`:

- `text`: значение `binary` из текстового представления. Дополнительные сведения см. в разделе `Binary.FromText` .

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Получить значение `binary` для `"1011"`.

Код

```
Binary.From("1011")
```

Результат

```
Binary.FromText("1011", BinaryEncoding.Base64)
```

Binary.FromList

Binary.FromList(list as list) as binary

Описание

Преобразует список чисел в двоичное значение.

Binary.FromText

Binary.FromText(text as text, optional encoding as number) as binary

Описание

Возвращает результат преобразования текстового значения `text` в двоичное (список `number`). Можно задать параметр `encoding`, чтобы указать кодировку, используемую в текстовом значении. Следующие значения `BinaryEncoding` могут быть использованы для `encoding`.

- `BinaryEncoding.Base64`: кодировка Base 64
- `BinaryEncoding.Hex`: шестнадцатеричная кодировка

Пример №-1

Описание

Декодирование "1011" в двоичное значение.

Код

```
Binary.FromText("1011")
```

Результат

```
Binary.FromText("1011", BinaryEncoding.Base64)
```

Пример №-2

Описание

Декодирование "1011" в двоичное значение с шестнадцатеричной кодировкой.

Код

```
Binary.FromText("1011", BinaryEncoding.Hex)
```

Результат

```
Binary.FromText("EBE=", BinaryEncoding.Base64)
```

Binary.Length

Binary.Length(binary as binary) as number

Описание

Возвращает число символов.

Binary.ToList

Binary.ToList(binary as binary) as list

Описание

Преобразует двоичное значение в список чисел.

Binary.ToText

Binary.ToText(binary as binary, optional encoding as number) as text

Описание

Возвращает результат преобразования списка двоичных чисел `binary` в текстовое значение. При необходимости можно задать параметр `encoding`, чтобы указать кодировку, используемую в формируемом текстовом значении Следующие значения `BinaryEncoding` могут быть использованы для `encoding`.

- `BinaryEncoding.Base64`: кодировка Base 64
- `BinaryEncoding.Hex`: шестнадцатеричная кодировка

BinaryFormat

BinaryFormat.7BitEncodedSignedInteger

BinaryFormat.7BitEncodedSignedInteger(binary as binary) as any

Описание

Двоичный формат, который считывает 64-разрядное целое число со знаком, зашифрованное с использованием 7-разрядной кодировки с переменной длиной.

BinaryFormat.7BitEncodedUnsignedInteger

BinaryFormat.7BitEncodedUnsignedInteger(binary as binary) as any

Описание

Двоичный формат, который считывает 64-разрядное целое число без знака, зашифрованное с использованием 7-разрядной кодировки с переменной длиной.

BinaryFormat.Binary

BinaryFormat.Binary(optional length as any) as function

Описание

Возвращает двоичный формат, который считывает двоичное значение. Если задан параметр `length`, то двоичное значение будет содержать указанное число байтов. Если параметр `length` не указан, то двоичное значение будет содержать оставшиеся байты. `length` может быть указан в виде числа или как двоичный формат длины, предшествующей двоичным данным.

BinaryFormat.Byte

BinaryFormat.Byte(binary as binary) as any

Описание

Двоичный формат, который считывает 8-разрядное целое число без знака.

BinaryFormat.ByteOrder

BinaryFormat.ByteOrder(binaryFormat as function, byteOrder as number) as function

Описание

Возвращает двоичный формат с порядком следования байтов, указанным `binaryFormat`. Порядок байтов по умолчанию: `ByteOrder.BigEndian`.

BinaryFormat.Choice

BinaryFormat.Choice(binaryFormat as function, chooseFunction as function, optional type as type, optional combineFunction as function) as function

Описание

Возвращает двоичный формат, который выбирает следующий двоичный формат в зависимости от значения, которое уже было считано. Значение двоичного формата, сформированного этой функцией, обрабатывается в два этапа:

- Двоичный формат, указанный параметром `binaryFormat`, используется для считывания значения.
- Значение передается функции выбора, заданной параметром `chooseFunction`.
- Функция выбора проверяет значение и возвращает второй двоичный формат.
- Второй двоичный формат используется для чтения второго значения.
- Если указана функция объединения, то первое и второе значения передаются в эту функцию, а возвращается значение результата.
- Если функция объединения не указана, возвращается второе значение.
- Возвращается второе значение.

Необязательный параметр `type` указывает тип двоичного формата, который возвращается функцией выбора. Допустимые значения: `type any`, `type list` или `type binary`. Если параметр `type` не указан, то используется `type any`. Если используется параметр `type list` или `type binary`, система может вернуть значение `binary` или `list` из потока вместо значения из буфера, что может сократить объем памяти, необходимый для чтения формата.

Пример №-1

Описание

Считывает список байтов, где число элементов определяется первым байтом с использованием списка потоков.

Код

```
let
    binaryData = #binary({2, 3, 4, 5}),
    listFormat = BinaryFormat.Choice(
        BinaryFormat.Byte,
        (length) => BinaryFormat.List(BinaryFormat.Byte, length),
        type list)
in
    listFormat(binaryData)
```

Результат

```
{3, 4}
```

Пример №-2

Описание

Считывает список байтов, где число элементов определяется первым байтом.

Код

```
let
    binaryData = #binary({2, 3, 4, 5}),
    listFormat = BinaryFormat.Choice(
        BinaryFormat.Byte,
        (length) => BinaryFormat.List(BinaryFormat.Byte, length))
in
    listFormat(binaryData)
```

Результат

```
{3, 4}
```

Пример №-3

Описание

Считывает список байтов, где число элементов определяется первым байтом, а первый считанный байт сохраняется.

Код

```
let
    binaryData = #binary({2, 3, 4, 5}),
    listFormat = BinaryFormat.Choice(
        BinaryFormat.Byte,
        (length) => BinaryFormat.Record([
            length = length,
            list = BinaryFormat.List(BinaryFormat.Byte, length)
        ]))
in
    listFormat(binaryData)
```

Результат

```
[ length = 2, list = {3, 4} ]
```

BinaryFormat.Decimal

BinaryFormat.Decimal(binary as binary) as any

Описание

Двоичный формат, который считывает 16-байтовое десятичное значение .NET.

BinaryFormat.Double

BinaryFormat.Double(binary as binary) as any

Описание

Двоичный формат, который считывает 8-байтовое значение двойной точности с плавающей запятой в формате IEEE.

BinaryFormat.Group

BinaryFormat.Group(binaryFormat as function, group as list, optional extra as function, optional lastKey as any) as function

Описание

Используются следующие параметры.

- Параметр `binaryFormat` указывает двоичный формат значения ключа.
- Параметр `group` предоставляет сведения о группе известных элементов.
- Необязательный параметр `extra` можно использовать для определения функции, возвращающей значение двоичного формата для значения, идущего после любого непредвиденного ключа. Если параметр `extra` не указан, при наличии непредвиденных значений ключей возникнет ошибка.

Параметр `group` указывает список определений элементов. Каждое определение элемента - это список, содержащий 3-5 значений:

- Значение ключа. Значение ключа, который соответствует элементу. Это должно быть уникальное значение в наборе элементов.
- Формат элемента. Двоичный формат, соответствующий значению элемента. Это позволяет каждому элементу использовать свой формат.
- Вхождение элемента. Значение `BinaryOccurrence.Type`, которое определяет, сколько раз элемент будет указан в группе. Если обязательные элементы отсутствуют, возникает ошибка. Обязательные или необязательные повторяющиеся элементы обрабатываются как непредвиденные значения ключа.
- Значение элемента по умолчанию (необязательно). Если значение элемента по умолчанию отображается в списке определений элемента и не равно NULL, оно будет использоваться вместо значения по умолчанию. Значение по умолчанию для повторяющихся и необязательных элементов равно NULL, а значение по умолчанию для повторяющихся значений — пустой список { }.
- Преобразование значения элемента (необязательно). Если функция преобразования значения элемента присутствует в списке определений элемента и не равна NULL, она будет вызвана для преобразования значения элемента перед его возвращением. Функция преобразования вызывается, только если элемент присутствует в входных данных (она не вызывается со значением по умолчанию).

Пример №-1

Описание

В следующем примере показано преобразование значения элемента и значения элемента по умолчанию. Повторяющийся элемент с ключом 1 суммирует список значений, считанных с помощью List.Sum. Необязательный элемент с ключом 2 имеет значение по умолчанию 123, а не NULL.

Код

```
let
```

```
    b = #binary(
    {
        1, 101,
        1, 102
    }),
    f = BinaryFormat.Group(
        BinaryFormat.Byte,
        {
            { 1, BinaryFormat.Byte, BinaryOccurrence.Repeating,
              0, (list) => List.Sum(list) },
            { 2, BinaryFormat.Byte, BinaryOccurrence.Optional, 123 }
        })
in
    f(b)
```

Результат

```
{ 203, 123 }
```

Пример №-2

Описание

Далее предполагается, что значение ключа - это один байт, в группе 4 элемента, во всех из которых после ключа идет байт данных. Элементы отображаются во входных данных следующим образом:

- *Ключ 1 является обязательным и не отображается со значением 11.*
- *Ключ 2 повторяется и отображается дважды со значением 22 и приводит к формированию значения { 22, 22 }.*
- *Ключ 3 не является обязательным, не отображается и приводит к формированию значения NULL.*
- *Ключ 4 повторяется, но не отображается и приводит к формированию значения { }.*
- *Ключ 5 не является частью группы, но появляется один раз со значением 55. Дополнительная функция вызывается со значением ключа 5 и возвращает формат, соответствующий этому значению (BinaryFormat.Byte). Значение 55 считывается и удаляется.*

Код

```
let
    b = #binary(
    {
        1, 11,
        2, 22,
        2, 22,
        5, 55,
        1, 11
    }),
    f = BinaryFormat.Group(
        BinaryFormat.Byte,
        {
            { 1, BinaryFormat.Byte, BinaryOccurrence.Required },
            { 2, BinaryFormat.Byte, BinaryOccurrence.Repeating },
            { 3, BinaryFormat.Byte, BinaryOccurrence.Optional },
            { 4, BinaryFormat.Byte, BinaryOccurrence.Repeating }
        },
        (extra) => BinaryFormat.Byte)
in
    f(b)
```

Результат

```
{ 11, { 22, 22 }, null, { } }
```

BinaryFormat.Length

BinaryFormat.Length(binaryFormat as function, length as any) as function

Описание

Возвращает двоичный формат, который ограничивает объем данных, который может быть считан. И `BinaryFormat.List`, и `BinaryFormat.Binary` можно использовать для считывания до конца данных. `BinaryFormat.Length` можно использовать для ограничения числа считываемых байтов. Параметр `binaryFormat` указывает двоичный формат, который нужно ограничить. Параметр `length` указывает число байтов для считывания. Параметр `length` может быть числовым значением или значением двоичного формата, указывающим формат значения длины, предшествующей считанному значению.

Пример №-1

Описание

Ограничьте число считанных байтов при чтении списка байтов до байтового значения, предшествующего списку.

Код

```
let
    binaryData = #binary({1, 2, 3}),
    listFormat = BinaryFormat.Length(
        BinaryFormat.List(BinaryFormat.Byte), BinaryFormat.Byte)
in
    listFormat(binaryData)
```

Результат

```
{2}
```

Пример №-2

Описание

Ограничьте число считанных байтов до 2 при чтении списка байтов.

Код

```
let
    binaryData = #binary({1, 2, 3}),
    listFormat = BinaryFormat.Length(
        BinaryFormat.List(BinaryFormat.Byte), 2)
in
    listFormat(binaryData)
```

Результат

```
{1, 2}
```

BinaryFormat.List

BinaryFormat.List(binaryFormat as function, optional countOrCondition as any) as function

Описание

Возвращает двоичный формат, который считывает последовательность элементов и возвращает `list`. Параметр `binaryFormat` указывает двоичный формат каждого элемента. Определить число считанных элементов можно тремя способами.

- Если параметр `countOrCondition` не указан, то двоичный формат будет считываться, пока не будут обработаны все элементы.
- Если `countOrCondition` - это число, то двоичный формат будет считывать только указанное число элементов.
- Если `countOrCondition` - это функция, она будет вызвана для чтения каждого элемента. Функция возвращает значение true для продолжения и значение false, чтобы остановить считывание элементов. Последний элемент включается в список.
- Если countOrCondition - это двоичный формат, ожидается, что указанное количество элементов будет предшествовать списку, а указанный формат будет использоваться для чтения этого количества.

Пример №-1

Описание

Считывает байты, пока байтовое значение меньше 2.

Код

```
let
    binaryData = #binary({1, 2, 3}),
    listFormat = BinaryFormat.List(BinaryFormat.Byte, (x) => x < 2)
in
    listFormat(binaryData)
```

Результат

```
{1, 2}
```

Пример №-2

Описание

Считывает 2 байта.

Код

```
let
    binaryData = #binary({1, 2, 3}),
    listFormat = BinaryFormat.List(BinaryFormat.Byte, 2)
in
```

```
    listFormat(binaryData)
```

Результат

```
{1, 2}
```

Пример №-3

Описание

Считывает байты до конца данных.

Код

```
let
    binaryData = #binary({1, 2, 3}),
    listFormat = BinaryFormat.List(BinaryFormat.Byte)
in
    listFormat(binaryData)
```

Результат

```
{1, 2, 3}
```

BinaryFormat.Null

BinaryFormat.Null(binary as binary) as any

Описание

Двоичный формат, который считывает нуль байт и возвращает NULL.

BinaryFormat.Record

BinaryFormat.Record(record as record) as function

Описание

Возвращает двоичный формат, который считывает запись. Параметр `record` указывает формат записи. Каждое поле записи может содержать различные двоичные форматы. Если поле содержит значение, которое не является значением двоичного формата, то данные не для этого поля не считываются, а значение поля переносится в результат.

Пример №-1

Описание

Считывает запись, содержащую одно 16-разрядное целое число и одно 32-разрядное целое число.

Код

```
let
```

```
    binaryData = #binary({
        0x00, 0x01,
        0x00, 0x00, 0x00, 0x02}),
    recordFormat = BinaryFormat.Record([
        A = BinaryFormat.UnsignedInteger16,
        B = BinaryFormat.UnsignedInteger32
    ])
in
    recordFormat(binaryData)
```

Результат

```
[A = 1, B = 2]
```

BinaryFormat.SignedInteger16

BinaryFormat.SignedInteger16(binary as binary) as any

Описание

Двоичный формат, который считывает 16-разрядное целое число со знаком.

BinaryFormat.SignedInteger32

BinaryFormat.SignedInteger32(binary as binary) as any

Описание

Двоичный формат, который считывает 32-разрядное целое число со знаком.

BinaryFormat.SignedInteger64

BinaryFormat.SignedInteger64(binary as binary) as any

Описание

Двоичный формат, который считывает 64-разрядное целое число со знаком.

BinaryFormat.Single

BinaryFormat.Single(binary as binary) as any

Описание

Двоичный формат, который считывает 4-байтовое значение одиночной точности с плавающей запятой в формате IEEE.

BinaryFormat.Text

BinaryFormat.Text(length as any, optional encoding as number) as function

Описание

Возвращает двоичный формат, который считывает текстовое значение. `length` указывает число байтов для расшифровки или двоичный формат длины, предшествующей тексту. Необязательное

значение `encoding` указывает кодировку текста. Если параметр `encoding` не указан, то кодировка определяется по меткам порядка байтов Юникода. Если метки порядка байтов отсутствуют, то используется `TextEncoding.Utf8`.

Пример №-1

Описание

Расшифровка 2 байтов в виде текста ASCII.

Код

```
let
    binaryData = #binary({65, 66, 67}),
    textFormat = BinaryFormat.Text(2, TextEncoding.Ascii)
in
    textFormat(binaryData)
```

Результат

```
"AB"
```

Пример №-2

Описание

Расшифровка текста ASCII, где длина текста в байтах указывается перед текстом в виде байта.

Код

```
let
    binaryData = #binary({2, 65, 66}),
    textFormat = BinaryFormat.Text(BinaryFormat.Byte,
        TextEncoding.Ascii)
in
    textFormat(binaryData)
```

Результат

```
"AB"
```

BinaryFormat.Transform

BinaryFormat.Transform(binaryFormat as function, function as function) as function

Описание

Возвращает двоичный формат, который преобразует значения, считанные другим двоичным форматом. Параметр `binaryFormat` указывает двоичный формат, который будет использоваться для считывания значений. `function` вызывается со считанным значением и возвращает преобразованное значение.

Пример №-1

Описание

Считывает байт и добавляет к нему единицу.

Код

```
let
    binaryData = #binary({1}),
    transformFormat = BinaryFormat.Transform(
        BinaryFormat.Byte,
        (x) => x + 1)
in
    transformFormat(binaryData)
```

Результат

```
2
```

BinaryFormat.UnsignedInteger16

BinaryFormat.UnsignedInteger16(binary as binary) as any

Описание

Двоичный формат, который считывает 16-разрядное целое число без знака.

BinaryFormat.UnsignedInteger32

BinaryFormat.UnsignedInteger32(binary as binary) as any

Описание

Двоичный формат, который считывает 32-разрядное целое число без знака.

BinaryFormat.UnsignedInteger64

BinaryFormat.UnsignedInteger64(binary as binary) as any

Описание

Двоичный формат, который считывает 64-разрядное целое число без знака.

Byte

Byte.From

Byte.From(value as any, optional culture as text, optional roundingMode as number) as number

Описание

Возвращает 8-разрядное целое значение `number` из данного значения `value`. Если данное значение `value` является значением `null`, `Byte.From` возвращает `null`. Если данное значение `value` является

значением `number` в диапазоне 8-разрядных целых чисел без дробной части, возвращается `value`. Если есть дробная часть, число округляется в заданном режиме. Режим округления по умолчанию — `RoundingMode.ToEven`. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в 8-разрядное целое значение `number`. См. в описании `Number.Round` допустимые режимы округления.

Пример №-1

Описание

Получить 8-разрядное целое значение `number` для `"4"`.

Код

```
Byte.From("4")
```

Результат

```
4
```

Пример №-2

Описание

Получить 8-разрядное целое значение `number` для `"4.5"`, используя `RoundingMode.AwayFromZero`.

Код

```
Byte.From("4.5", null, RoundingMode.AwayFromZero)
```

Результат

```
5
```

Character

Character.FromNumber

Character.FromNumber(number as number) as text

Описание

Возвращает символ, эквивалентный значению.

Пример №-1

Описание

Найти значение символа для числа 9.

Код

```
Character.FromNumber(9)
```

Результат

```
"#(tab)"
```

Character.ToNumber

Character.ToNumber(character as text) as number

Описание

Возвращает число, эквивалентное символу, `character`.

Пример №-1

Описание

Найти числовое значение для символа "#(tab)", 9.

Код

```
Character.ToNumber("#(tab)")
```

Результат

```
9
```

Combiner

Combiner.CombineTextByDelimiter

Combiner.CombineTextByDelimiter(delimiter as text, optional quoteStyle as number) as function

Описание

Возвращает функцию, которая объединяет список текстовых значений в один текст с помощью заданного разделителя.

Combiner.CombineTextByEachDelimiter

Combiner.CombineTextByEachDelimiter(delimiters as list, optional quoteStyle as number) as function

Описание

Возвращает функцию, которая объединяет список текстовых значений в один текст с помощью каждого разделителя в последовательности.

Combiner.CombineTextByLengths

Combiner.CombineTextByLengths(lengths as list, optional template as text) as function

Описание

Возвращает функцию, которая объединяет список текстовых значений в один текст с помощью заданной длиной.

Combiner.CombineTextByPositions

Combiner.CombineTextByPositions(positions as list, optional template as text) as function

Описание

Возвращает функцию, которая объединяет список текстовых значений в один текст с использованием заданных положений.

Combiner.CombineTextByRanges

Combiner.CombineTextByRanges(ranges as list, optional template as text) as function

Описание

Возвращает функцию, которая объединяет список текстовых значений в один текст с использованием заданных положений и длин.

Comparer

Comparer.Equals

Comparer.Equals(comparer as function, x as any, y as any) as logical

Описание

Возвращает значение `logical` на основании проверки равенства двух заданных значений, `x` и `y`, с использованием предоставленного `comparer`.

`comparer` — это модуль `Comparer`, который используется для управления сравнением. Функции сравнения можно использовать для сравнений, учитывающих или не учитывающих регистр, язык и региональные параметры, а также локаль.
В языке формул доступны следующие встроенные функции сравнения:

- `Comparer.Ordinal` — используется для точного сравнения по порядковому номеру
- `Comparer.OrdinalIgnoreCase`: используется для точного сравнения по порядковому номеру без учета регистра
- `Comparer.FromCulture` — используется для сравнения с учетом языка и региональных параметров

Пример №-1

Описание

Сравнение "1" и "A" с использованием языка "en-US" для определения того, равны ли значения.

Код

```
Comparer.Equals(Comparer.FromCulture("en-us"), "1", "A")
```

Результат

```
false
```

Comparer.FromCulture

Comparer.FromCulture(culture as text, optional ignoreCase as logical) as function

Описание

Возвращает функцию сравнения с учетом `culture` и логическое значение `ignoreCase`, определяющее чувствительность к регистру для операции сравнения. Значением по умолчанию для `ignoreCase` является false. Значения для языка и региональных стандартов представляют собой общепринятые текстовые представления языков, используемые в .NET Framework.

Пример №-1

Описание

Сравнение "a" и "A" с использованием языка "en-US" для определения того, равны ли значения.

Код

```
Comparer.FromCulture("en-us")("a", "A")
```

Результат

```
-1
```

Пример №-2

Описание

Сравнение "a" и "A" с использованием языка "en-US" без учета регистра для определения того, равны ли значения.

Код

```
Comparer.FromCulture("en-us", true)("a", "A")
```

Результат

```
0
```

Comparer.Ordinal

Comparer.Ordinal(x as any, y as any) as number

Описание

Возвращает функцию сравнения, использующую порядковые правила для сравнения указанных значений `x` и `y`.

Пример №-1

Описание

Используя порядковые правила, выяснить, эквивалентны ли значения "encyclop?dia" и "encyclopaedia". Обратите внимание, что они эквивалентны при использовании `Comparer.FromCulture("en-us")`.

Код

```
Comparer.Equals(Comparer.Ordinal, "encyclop?dia", "encyclopaedia")
```

Результат

```
false
```

Comparer.OrdinalIgnoreCase

Comparer.OrdinalIgnoreCase(x as any, y as any) as number

Описание

Возвращает функцию сравнения без учета регистра, использующую при сравнении указанных значений `x` и `y` правила Ordinal.

Пример №-1

Описание

Используя правила Ordinal без учета регистра, сравним "Abc" с "abc". Обратите внимание, что "Abc" считается меньше "abc", если использовать `Comparer.Ordinal`.

Код

```
Comparer.OrdinalIgnoreCase("Abc", "abc")
```

Результат

```
0
```

Csv

Csv.Document

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

Описание

Возвращает содержимое CSV-документа в виде таблицы.

- `columns`: если указана запись, а значения `delimiter`, `extraValues`, `encoding` равны NULL, все параметры `Delimiter`, `Columns`, `Encoding`, `CsvStyle` и `QuoteStyle` извлекаются из записи.
- `delimiter` может получить один символ или список; если значение не указано, используется запятая или NULL.
- Поддерживаемые значения дополнительного параметра `extraValues` см. в `ExtraValues.Type`.
- `encoding` указывает тип кодировки текста.

Пример №-1

Описание

Обработка текста в формате CSV с заголовками столбцов

Код

```
Table.PromoteHeaders(Csv.Document("OrderID,Item
1,Fishing rod
2,1 lb. worms"))
```

Результат

```
Table.FromRecords({
    [ OrderID = "1", Item = "Fishing rod" ],
    [ OrderID = "2", Item = "1 lb. worms" ]
})
```

Cube

Cube.AddAndExpandDimensionColumn

Cube.AddAndExpandDimensionColumn(cube as table, dimensionSelector as any, attributeNames as list, optional newColumnNames as any) as table

Описание

Совмещает указанную таблицу измерения `dimensionSelector` в контексте фильтра куба `cube` и изменяет степень гранулярности измерения контекста фильтра, расширяя указанный набор атрибутов измерения `attributeNames`. Атрибуты измерения добавляются в табличное представление со столбцами `newColumnNames` или `attributeNames`, если это значение не задано.

Cube.AddMeasureColumn

Cube.AddMeasureColumn(cube as table, column as text, measureSelector as any) as table

Описание

Добавляет столбец с именем `column` к `cube`, который содержит результаты меры `measureSelector`, примененной в контексте каждой строки. На применение меры оказывают влияние изменения в степени гранулярности измерения и параметры срезов. Значения меры будут поправлены после выполнения некоторых операций куба.

Cube.ApplyParameter

Cube.ApplyParameter(cube as table, parameter as any, optional arguments as list) as table

Описание

Возвращает куб после применения "`parameter`" с "`arguments`" к "`cube`".

Cube.AttributeMemberId

Cube.AttributeMemberId(value as any) as any

Описание

Возвращает уникальный идентификатор члена из значения свойства члена. `value`. Возвращает значение NULL для всех остальных значений.

Cube.CollapseAndRemoveColumns

Cube.CollapseAndRemoveColumns(cube as table, columnNames as list) as table

Описание

Изменяет степень гранулярности измерения контекста фильтра `cube`, сворачивая атрибуты, сопоставленные с указанными столбцами `columnNames`. Столбцы также удаляются из табличного представления куба.

Cube.Dimensions

Cube.Dimensions(cube as table) as table

Описание

Возвращает таблицу, содержащую набор доступных измерений в пределах `cube`. Каждое измерение - это таблица с набором атрибутов измерения. Каждый атрибут измерения представлен как столбец в этой таблице измерения. Измерения в кубе можно развернуть с помощью метода Cube.AddAndExpandDimensionColumn.

Cube.DisplayFolders

Cube.DisplayFolders(cube as table) as table

Описание

Возвращает составное дерево таблиц, представляющих отображаемую иерархию папок объектов (т. е. измерений и мер), доступных для использования в `cube`.

Cube.Measures

Cube.Measures(cube as any) as table

Описание

Возвращает таблицу, содержащую набор доступных мер в пределах `cube`. Каждая мера представлена функцией. Меры могут применяться к кубу с помощью метода Cube.AddMeasureColumn.

Cube.Parameters

Cube.Parameters(cube as table) as table

Описание

Возвращает таблицу, содержащую набор параметров, которые можно применить к "`cube`". Каждый параметр представляет из себя функцию, которую можно вызвать для того, чтобы получить "`cube`" с примененными параметром и агрументами.

Cube.Transform

Cube.Transform(cube as table, transforms as list) as table

Описание

Применяет список функций куба `transforms` к `cube`.

Currency

Currency.From

Currency.From(value as any, optional culture as text, optional roundingMode as number) as number

Описание

Возвращает значение `currency` из данного `value`. Если данный `value` — `null`, `Currency.From` возвращает `null`. Если данный `value` — `number` в пределах валюты, дробная часть `value` округляется до 4 десятичных цифр и возвращается. Если данный `value` — любого другого типа, см. `Number.FromText` для преобразования его в значение-`number`, затем применяется приведенное ранее утверждение о преобразовании значения `number` к значению `currency`. Допустимый диапазон для валют — от `-922 337 203 685 477,5808` до `922 337 203 685 477,5807`. См. в описании `Number.Round` допустимые режимы округления. По умолчанию используется `RoundingMode.ToEven`.

Пример №-1

Описание

Получим значение `currency` для `"1,23455"`, используя `RoundingMode.Down`.

Код

```
Currency.From("1.23455", "en-Us", RoundingMode.Down)
```

Результат

```
1.2345
```

Пример №-2

Описание

Получить значение `валюты` *из значения* `"1.23455"`.

Код

```
Currency.From("1.23455")
```

Результат

```
1.2346
```

DB2

DB2.Database

DB2.Database(server as text, database as text, optional options as record) as table

Описание

Возвращает список таблиц и представлений SQL, доступных в базе данных DB2 на сервере `server` в экземпляре базы данных `database`. Дополнительно к имени сервера через двоеточие может быть указан порт. Необязательный параметр записи `options` может быть указан для управления следующими параметрами:

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.
- `Implementation` : Указывает используемую реализацию поставщика внутренней базы данных. Допустимые значения: "IBM" и "Microsoft".
- `BinaryCodePage` : Номер CCSID (идентификатор набора кодированных знаков) для расшифровки двоичных данных DB2 FOR BIT в строки символов. Применяется к Implementation = "Microsoft". Задайте значение 0, чтобы отключить преобразование (по

умолчанию). Задайте значение 1, чтобы преобразовать на основе кодировки базы данных. Задайте другой номер CCSID, чтобы преобразовать в кодировку приложения.
- `PackageCollection` : Указывает значение строки для коллекции пакетов (значение по умолчанию — "NULLID"), чтобы разрешить использование общих пакетов, необходимых для обработки инструкций SQL. Применяется к Implementation = "Microsoft".

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

Date

Date.AddDays

Date.AddDays(dateTime as any, numberOfDays as number) as any

Описание

Возвращает результат `date`, `datetime` или `datetimezone` после добавления `numberOfDays` дней к значению `datetime`, `dateTime`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, к которому добавляются дни.
- `numberOfDays`: количество дней, которые нужно добавить.

Пример №-1

Описание

Добавляет 5 дней к значению `date`, `datetime` или `datetimezone`, представляющему дату 14.05.2011.

Код

```
Date.AddDays(#date(2011, 5, 14), 5)
```

Результат

```
#date(2011, 5, 19)
```

Date.AddMonths

Date.AddMonths(dateTime as any, numberOfMonths as number) as any

Описание

Возвращает результат `date`, `datetime` или `datetimezone` после добавления `numberOfMonths` месяцев к значению `datetime`, `dateTime`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, к которому добавляются месяцы.
- `numberOfMonths`: число месяцев, которое нужно добавить.

Пример №-1

Описание

Добавляет 5 месяцев к значению `date`, `datetime` *или* `datetimezone`, *представляющему дату 14.05.2011.*

Код

```
Date.AddMonths(#date(2011, 5, 14), 5)
```

Результат

```
#date(2011, 10, 14)
```

Пример №-2

Описание

Добавляет 18 месяцев к значению `date`, `datetime` *или* `datetimezone`, *представляющее дату и время 14.05.2011 08:15:22.*

Код

```
Date.AddMonths(#datetime(2011, 5, 14, 8, 15, 22), 18)
```

Результат

```
#datetime(2012, 11, 14, 8, 15, 22)
```

Date.AddQuarters

Date.AddQuarters(dateTime as any, numberOfQuarters as number) as any

Описание

Возвращает результат `date`, `datetime` или `datetimezone` после добавления `numberOfQuarters` кварталов к значению `datetime`, `dateTime`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, к которому добавляются кварталы.
- `numberOfQuarters`: число кварталов, которое нужно добавить.

Пример №-1

Описание

Добавляет 1 квартал к значению `date`, `datetime` *или* `datetimezone`, *представляющему дату 14.05.2011.*

Код

```
Date.AddQuarters(#date(2011, 5, 14), 1)
```

Результат

```
#date(2011, 8, 14)
```

Date.AddWeeks

Date.AddWeeks(dateTime as any, numberOfWeeks as number) as any

Описание

Возвращает результат `date`, `datetime` или `datetimezone` после добавления `numberOfWeeks` недель к значению `datetime`, `dateTime`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, к которому добавляются недели.
- `numberOfWeeks`: число недель, которое нужно добавить.

Пример №-1

Описание

Добавляет 2 недели к значению `date`, `datetime` или `datetimezone`, представляющему дату 14.05.2011.

Код

```
Date.AddWeeks(#date(2011, 5, 14), 2)
```

Результат

```
#date(2011, 5, 28)
```

Date.AddYears

Date.AddYears(dateTime as any, numberOfYears as number) as any

Описание

Возвращает результат `date`, `datetime` или `datetimezone` после добавления `numberOfYears` к значению `datetime`, `dateTime`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, к которому добавляются годы.
- `numberOfYears`: число лет, которое нужно добавить.

Пример №-1

Описание

Добавляет 10 лет к значению `date`, `datetime` или `datetimezone`, представляющему дату и время 14.05.2011 08:15:22.

Код

```
Date.AddYears(#datetime(2011, 5, 14, 8, 15, 22), 10)
```

Результат

```
#datetime(2021, 5, 14, 8, 15, 22)
```

Пример №-2

Описание

Добавляет 4 года к значению `date`, `datetime` *или* `datetimezone`, *представляющему дату 14.05.2011.*

Код

```
Date.AddYears(#date(2011, 5, 14), 4)
```

Результат

```
#date(2015, 5, 14)
```

Date.Day

Date.Day(dateTime as any) as number

Описание

Возвращает компонент дня значения `date`, `datetime` или `datetimezone`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, из которого извлекается компонент дня.

Пример №-1

Описание

Получает компонент дня значения `date`, `datetime` *или* `datetimezone`, *представляющего дату 14.05.2011 и время 17:00:00.*

Код

```
Date.Day(#datetime(2011, 5, 14, 17, 0, 0))
```

Результат

```
14
```

Date.DayOfWeek

Date.DayOfWeek(dateTime as any, optional firstDayOfWeek as number) as any

Описание

Возвращает число от 0 до 6, представляющее день недели в заданном значении даты и времени (`dateTime`). Эта функция принимает необязательное значение дня (`firstDayOfWeek`) для установки первого дня недели для относительного вычисления. Значение firstDay по умолчанию — Day.Sunday. Допустимые значения: Day.Sunday, Day.Monday, Day.Tuesday, Day.Wednesday, Day.Thursday, Day.Friday и Day.Saturday.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, на основе которого определяется день недели.
- `firstDayOfWeek`: тип `Day`, представляющий первый день недели для данного вычисления.

Пример №-1

Описание

Возвращает день недели, на который выпадает 21 февраля 2011 г., если воскресенье является первым днем недели (по умолчанию).

Код

```
Date.DayOfWeek(#date(2011, 02, 21))
```

Результат

```
1
```

Пример №-2

Описание

Возвращает день недели, на который выпадает 21 февраля 2011 г., если понедельник является первым днем недели.

Код

```
Date.DayOfWeek(#date(2011, 02, 21), Day.Monday)
```

Результат

```
0
```

Date.DayOfWeekName

Date.DayOfWeekName(date as any, optional culture as text) as text

Описание

Возвращает название дня недели для указанного `date`, а при необходимости — язык и региональные параметры `culture`.

Пример №-1

Описание

Получение названия дня недели.

Код

```
Date.DayOfWeekName(#date(2011, 12, 31), "en-US")
```

Результат

```
"Saturday"
```

Date.DayOfYear

Date.DayOfYear(dateTime as any) as number

Описание

Возвращает число, представляющее день года в соответствующем значении `date`, `datetime` или `datetimezone`: `dateTime`.

Пример №-1

Описание

Номер дня для 1 марта 2011 г. (`#date(2011, 03, 01)`).

Код

```
Date.DayOfYear(#date(2011, 03, 01))
```

Результат

```
60
```

Date.DaysInMonth

Date.DaysInMonth(dateTime as any) as number

Описание

Возвращает число дней в месяце в значении `date`, `datetime` или `datetimezone`: `dateTime`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, для которого возвращается число дней в месяце.

Пример №-1

Описание

Число дней в декабре, представленное как `#date(2011, 12, 01)`.

Код

```
Date.DaysInMonth(#date(2011, 12, 01))
```

Результат

```
31
```

Date.EndOfDay

Date.EndOfDay(dateTime as any) as any

Описание

Возвращает значение `date`, `datetime` или `datetimezone`, представляющее конец дня в `dateTime`. Данные о часовом поясе сохраняются.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, на основе которого вычисляется конец дня.

Пример №-1

Описание

Возвращает конец дня для 14.05.2011 17:00:00.

Код

```
Date.EndOfDay(#datetime(2011, 5, 14, 17, 0, 0))
```

Результат

```
#datetime(2011, 5, 14, 23, 59, 59.9999999)
```

Пример №-2

Описание

Возвращает конец дня для 17.05.2011 17:00:00-19:00.

Код

```
Date.EndOfDay(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0))
```

Результат

```
#datetimezone(2011, 5, 17, 23, 59, 59.9999999, -7, 0)
```

Date.EndOfMonth

Date.EndOfMonth(dateTime as any) as any

Описание

Возвращает последний день месяца в `dateTime`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, на основе которого вычисляется конец месяца.

Пример №-1

Описание

Возвращает конец месяца для 14.05.2011.

Код

```
Date.EndOfMonth(#date(2011, 5, 14))
```

Результат

```
#date(2011, 5, 31)
```

Пример №-2

Описание

Возвращает конец месяца для 17.05.2011 17:00:00-7:00.

Код

```
Date.EndOfMonth(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0))
```

Результат

```
#datetimezone(2011, 5, 31, 23, 59, 59.9999999, -7, 0)
```

Date.EndOfQuarter

Date.EndOfQuarter(dateTime as any) as any

Описание

Возвращает значение `date`, `datetime` или `datetimezone`, представляющее конец квартала в `dateTime`. Данные о часовом поясе сохраняются.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, на основе которого вычисляется конец квартала.

Пример №-1

Описание

Поиск конца квартала для 10 октября 2011 г., 8:00 (`#datetime(2011, 10, 10, 8, 0, 0)`).

Код

```
Date.EndOfQuarter(#datetime(2011, 10, 10, 8, 0, 0))
```

Результат

```
#datetime(2011, 12, 31, 23, 59, 59.9999999)
```

Date.EndOfWeek

Date.EndOfWeek(dateTime as any, optional firstDayOfWeek as number) as any

Описание

Возвращает последний день недели в заданном значении `date`, `datetime` или `datetimezone` `dateTime`. Эта функция принимает необязательное значение `Day`, `firstDayOfWeek`, для установки первого дня недели для относительного вычисления. Значение по умолчанию равно `Day.Sunday`.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, на основе которого вычисляется последний день недели.
- `firstDayOfWeek`: *[Необязательно]* Значение `Day.Type`, представляющее первый день недели. Допустимые значения: `Day.Sunday`, `Day.Monday`, `Day.Tuesday`, `Day.Wednesday`, `Day.Thursday`, `Day.Friday` и `Day.Saturday`.. Значение по умолчанию - `Day.Sunday`.

  .

Пример №-1

Описание

Получает конец недели для 17.05.2011 17:00:00-7:00, если воскресенье - первый день недели.

Код

```
Date.EndOfWeek(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0), Day.Sunday)
```

Результат

```
#datetimezone(2011, 5, 21, 23, 59, 59.9999999, -7, 0)
```

Пример №-2

Описание

Возвращает конец недели для 14.05.2011.

Код

```
Date.EndOfWeek(#date(2011, 5, 14))
```

Результат

```
#date(2011, 5, 14)
```

Date.EndOfYear

Date.EndOfYear(dateTime as any) as any

Описание

Возвращает значение, представляющее конец года в `dateTime`, включая доли секунды. Данные о часовом поясе сохраняются.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, на основе которого вычисляется конец года.

Пример №-1

Описание

Возвращает конец года для 14.05.2011 17:00:00.

Код

```
Date.EndOfYear(#datetime(2011, 5, 14, 17, 0, 0))
```

Результат

```
#datetime(2011, 12, 31, 23, 59, 59.9999999)
```

Пример №-2

Описание

Получает конец часа для 17.05.2011 17:00:00-7:00.

Код

```
Date.EndOfYear(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0))
```

Результат

```
#datetimezone(2011, 12, 31, 23, 59, 59.9999999, -7, 0)
```

Date.From

Date.From(value as any, optional culture as text) as date

Описание

Возвращает значение `date` из указанного `value`. Если данное `value` имеет значение `null`, `Date.From` возвращает `null`. Если данное `value` - `date`, возвращается `value`. Значения следующих типов можно преобразовать в значение `date`:

- `text`: значение `date` из текстового представления. Дополнительные сведения см. в разделе `Date.FromText`.
- `datetime`: компонент даты `value`.
- `datetimezone`: компонент даты локального эквивалента datetime `value`.
- `number`: компонент даты эквивалента даты и времени OLE-автоматизации, выраженный `value`.

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Преобразовать `43910` в значение `date`.

Код

```
Date.From(43910)
```

Результат

```
#date(2020, 3, 20)
```

Пример №-2

Описание

Преобразовать `#datetime(1899, 12, 30, 06, 45, 12)` в значение `date`.

Код

```
Date.From(#datetime(1899, 12, 30, 06, 45, 12))
```

Результат

```
#date(1899, 12, 30)
```

Date.FromText

Date.FromText(text as text, optional culture as text) as date

Описание

Создает значение `date` из текстового представления `text` согласно стандарту формата ISO 8601.

- `Date.FromText("2010-02-19")` // Дата, гггг-ММ-дд

Пример №-1

Описание

Преобразует `"December 31, 2010"` в значение даты.

Код

```
Date.FromText("2010-12-31")
```

Результат

```
#date(2010, 12, 31)
```

Пример №-2

Описание

Преобразование `"December, 2010"` в значение date.

Код

```
Date.FromText("2010, 12")
```

Результат

```
#date(2010, 12, 1)
```

Пример №-3

Описание

Преобразование `"December 31, 2010"` в значение date с другим форматом

Код

```
Date.FromText("2010, 12, 31")
```

Результат

```
#date(2010, 12, 31)
```

Пример №-4

Описание

Преобразование `"2010"` в значение date.

Код

```
Date.FromText("2010")
```

Результат

```
#date(2010, 1, 1)
```

Date.IsInCurrentDay

Date.IsInCurrentDay(dateTime as any) as logical

Описание

Указывает, приходится заданное значение datetime `dateTime` на текущий день, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли текущее системное время на текущий день.

Код

```
Date.IsInCurrentDay(DateTime.FixedLocalNow())
```

Результат

```
true
```

Date.IsInCurrentMonth

Date.IsInCurrentMonth(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на текущий месяц, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли текущее системное время на текущий месяц.

Код

```
Date.IsInCurrentMonth(DateTime.FixedLocalNow())
```

Результат

```
true
```

Date.IsInCurrentQuarter

Date.IsInCurrentQuarter(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на текущий квартал, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли текущее системное время на текущий квартал.

Код

```
Date.IsInCurrentQuarter(DateTime.FixedLocalNow())
```

Результат

```
true
```

Date.IsInCurrentWeek

Date.IsInCurrentWeek(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на текущую неделю, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли текущее системное время на текущую неделю.

Код

```
Date.IsInCurrentWeek(DateTime.FixedLocalNow())
```

Результат

```
true
```

Date.IsInCurrentYear

Date.IsInCurrentYear(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на текущий год, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли текущее системное время на текущий год.

Код

```
Date.IsInCurrentYear(DateTime.FixedLocalNow())
```

Результат

```
true
```

Date.IsInNextDay

Date.IsInNextDay(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на следующий день, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли день, следующий после текущего системного времени, на следующий день.

Код

```
Date.IsInNextDay(Date.AddDays(DateTime.FixedLocalNow(), 1))
```

Результат

```
true
```

Date.IsInNextMonth

Date.IsInNextMonth(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на следующий месяц, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли месяц, следующий после текущего системного времени, на следующий месяц.

Код

```
Date.IsInNextMonth(Date.AddMonths(DateTime.FixedLocalNow(), 1))
```

Результат

```
true
```

Date.IsInNextNDays

Date.IsInNextNDays(dateTime as any, days as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение следующего количества дней, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `days`: количество дней.

Пример №-1

Описание

Определить, находится ли день после текущего системного времени в течение следующих двух дней.

Код

```
Date.IsInNextNDays(Date.AddDays(DateTime.FixedLocalNow(), 1), 2)
```

Результат

```
true
```

Date.IsInNextNMonths

Date.IsInNextNMonths(dateTime as any, months as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение следующего количества месяцев, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `months`: количество месяцев.

Пример №-1

Описание

Определить, находится ли месяц после текущего системного времени в течение следующих двух месяцев.

Код

```
Date.IsInNextNMonths(Date.AddMonths(DateTime.FixedLocalNow(), 1), 2)
```

Результат

```
true
```

Date.IsInNextNQuarters

Date.IsInNextNQuarters(dateTime as any, quarters as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение следующего количества кварталов, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `quarters`: количество кварталов.

Пример №-1

Описание

Определить, находится ли квартал после текущего системного времени в течение следующих двух кварталов.

Код

```
Date.IsInNextNQuarters(Date.AddQuarters(DateTime.FixedLocalNow(), 1), 2)
```

Результат

```
true
```

Date.IsInNextNWeeks

Date.IsInNextNWeeks(dateTime as any, weeks as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение следующего количества недель, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `weeks`: количество недель.

Пример №-1

Описание

Определить, находится ли неделя после текущего системного времени в течение следующих двух недель.

Код

```
Date.IsInNextNWeeks(Date.AddDays(DateTime.FixedLocalNow(), 7), 2)
```

Результат

```
true
```

Date.IsInNextNYears

Date.IsInNextNYears(dateTime as any, years as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение следующего количества лет, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.

- `years`: количество лет.

Пример №-1

Описание

Определить, находится ли год после текущего системного времени в течение следующих двух лет.

Код

```
Date.IsInNextNYears(Date.AddYears(DateTime.FixedLocalNow(), 1), 2)
```

Результат

```
true
```

Date.IsInNextQuarter

Date.IsInNextQuarter(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на следующий квартал, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли квартал, следующий после текущего системного времени, на следующий квартал.

Код

```
Date.IsInNextQuarter(Date.AddQuarters(DateTime.FixedLocalNow(), 1))
```

Результат

```
true
```

Date.IsInNextWeek

Date.IsInNextWeek(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на следующую неделю, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли неделя, идущая после текущего системного времени, на следующую неделю.

Код

```
Date.IsInNextWeek(Date.AddDays(DateTime.FixedLocalNow(), 7))
```

Результат

```
true
```

Date.IsInNextYear

Date.IsInNextYear(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на следующий год, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли год, идущий после текущего системного времени, на следующий год.

Код

```
Date.IsInNextYear(Date.AddYears(DateTime.FixedLocalNow(), 1))
```

Результат

```
true
```

Date.IsInPreviousDay

Date.IsInPreviousDay(dateTime as any) as logical

Описание

Указывает, приходится заданное значение datetime `dateTime` на предыдущий день, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли день, идущий до текущего системного времени, на предыдущий день.

Код

```
Date.IsInPreviousDay(Date.AddDays(DateTime.FixedLocalNow(), -1))
```

Результат

```
true
```

Date.IsInPreviousMonth

Date.IsInPreviousMonth(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на предыдущий месяц, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли месяц, идущий до текущего системного времени, на предыдущий месяц.

Код

```
Date.IsInPreviousMonth(Date.AddMonths(DateTime.FixedLocalNow(), -1))
```

Результат

```
true
```

Date.IsInPreviousNDays

Date.IsInPreviousNDays(dateTime as any, days as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение предыдущего количества дней, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `days`: количество дней.

Пример №-1

Описание

Определить, находится ли день до текущего системного времени в течение следующих двух дней.

Код

```
Date.IsInPreviousNDays(Date.AddDays(DateTime.FixedLocalNow(), -1), 2)
```

Результат

```
true
```

Date.IsInPreviousNMonths

Date.IsInPreviousNMonths(dateTime as any, months as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение предыдущего количества месяцев, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `months`: количество месяцев.

Пример №-1

Описание

Определить, находится ли месяц до текущего системного времени в течение следующих двух месяцев.

Код

```
Date.IsInPreviousNMonths(Date.AddMonths(DateTime.FixedLocalNow(), -1), 2)
```

Результат

```
true
```

Date.IsInPreviousNQuarters

Date.IsInPreviousNQuarters(dateTime as any, quarters as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение предыдущего количества кварталов, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `quarters`: количество кварталов.

Пример №-1

Описание

Определить, находится ли квартал до текущего системного времени в течение следующих двух кварталов.

Код

```
Date.IsInPreviousNQuarters(Date.AddQuarters(DateTime.FixedLocalNow(), -1), 2)
```

Результат

```
true
```

Date.IsInPreviousNWeeks

Date.IsInPreviousNWeeks(dateTime as any, weeks as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение предыдущего количества недель, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `weeks`: количество недель.

Пример №-1

Описание

Определить, находится ли неделя до текущего системного времени в течение следующих двух недель.

Код

```
Date.IsInPreviousNWeeks(Date.AddDays(DateTime.FixedLocalNow(), -7), 2)
```

Результат

```
true
```

Date.IsInPreviousNYears

Date.IsInPreviousNYears(dateTime as any, years as number) as logical

Описание

Указывает, настанет ли данное значение даты и времени `dateTime` в течение предыдущего количества лет, определенного текущей датой и временем системы.

- `dateTime`: значения `date`, `datetime` или `datetimezone` для оценки.
- `years`: количество лет.

Пример №-1

Описание

Определить, находится ли год до текущего системного времени в течение следующих двух лет.

Код

```
Date.IsInPreviousNYears(Date.AddYears(DateTime.FixedLocalNow(), -1), 2)
```

Результат

```
true
```

Date.IsInPreviousQuarter

Date.IsInPreviousQuarter(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на предыдущий квартал, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли квартал, идущий до текущего системного времени, на предыдущий квартал.

Код

```
Date.IsInPreviousQuarter(Date.AddQuarters(DateTime.FixedLocalNow(), -1))
```

Результат

```
true
```

Date.IsInPreviousWeek

Date.IsInPreviousWeek(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на предыдущую неделю, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли неделя, идущая до текущего системного времени, на предыдущую неделю.

Код

```
Date.IsInPreviousWeek(Date.AddDays(DateTime.FixedLocalNow(), -7))
```

Результат

```
true
```

Date.IsInPreviousYear

Date.IsInPreviousYear(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на предыдущий год, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, приходится ли год, идущий до текущего системного времени, на предыдущий год.

Код

```
Date.IsInPreviousYear(Date.AddYears(DateTime.FixedLocalNow(), -1))
```

Результат

```
true
```

Date.IsInYearToDate

Date.IsInYearToDate(dateTime as any) as logical

Описание

Указывает, приходится ли заданное значение datetime `dateTime` на текущий год и идет ли оно в течение текущего дня или после него, что определяется текущей датой и временем в системе.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, относится ли текущее системное время к периоду, прошедшему с начала года.

Код

```
Date.IsInYearToDate(DateTime.FixedLocalNow())
```

Результат

```
true
```

Date.IsLeapYear

Date.IsLeapYear(dateTime as any) as logical

Описание

Указывает, приходится ли указанное значение datetime `dateTime` на високосный год.

- `dateTime`: значение `date`, `datetime` или `datetimezone`, которое необходимо оценить.

Пример №-1

Описание

Определить, является ли 2012 год, представленный `#date(2012, 01, 01)`, високосным.

Код

```
Date.IsLeapYear(#date(2012, 01, 01))
```

Результат

```
true
```

Date.Month

Date.Month(dateTime as any) as number

Описание

Возвращает компонент месяца заданного значения `datetime`, `dateTime`.

Пример №-1

Описание

Найти компонент месяца в #datetime(2011, 12, 31, 9, 15, 36).

Код

```
Date.Month(#datetime(2011, 12, 31, 9, 15, 36))
```

Результат

```
12
```

Date.MonthName

Date.MonthName(date as any, optional culture as text) as text

Описание

Возвращает название компонента месяца для указанного `date`, а при необходимости — язык и региональные параметры `culture`.

Пример №-1

Описание

Получение названия месяца.

Код

```
Date.MonthName(#datetime(2011, 12, 31, 5, 0, 0), "en-US")
```

Результат

```
"December"
```

Date.QuarterOfYear

Date.QuarterOfYear(dateTime as any) as number

Описание

Возвращает число от 1 до 4, указывающее квартал года, на который приходится дата `dateTime`. `dateTime` может иметь значение `date`, `datetime` или `datetimezone`.

Пример №-1

Описание

Поиск квартала года для даты в #date(2011, 12, 31).

Код

```
Date.QuarterOfYear(#date(2011, 12, 31))
```

Результат

```
4
```

Date.StartOfDay

Date.StartOfDay(dateTime as any) as any

Описание

Возвращает первое значение дня `dateTime`. `dateTime` должно быть значением `date`, `datetime` или `datetimezone`.

Пример №-1

Описание

Поиск начала дня для 10 октября 2011 г., 8:00 (`#datetime(2011, 10, 10, 8, 0, 0)`).

Код

```
Date.StartOfDay(#datetime(2011, 10, 10, 8, 0, 0))
```

Результат

```
#datetime(2011, 10, 10, 0, 0, 0)
```

Date.StartOfMonth

Date.StartOfMonth(dateTime as any) as any

Описание

Возвращает первое значение месяца для заданного типа `date` или `datetime`.

Пример №-1

Описание

Поиск начала месяца для 10 октября 2011 г., 8:10:32 (`#datetime(2011, 10, 10, 8, 10, 32)`).

Код

```
Date.StartOfMonth(#datetime(2011, 10, 10, 8, 10, 32))
```

Результат

```
#datetime(2011, 10, 1, 0, 0, 0)
```

Date.StartOfQuarter

Date.StartOfQuarter(dateTime as any) as any

Описание

Возвращает первое значение квартала `dateTime`. `dateTime` должно быть значением `date`, `datetime` или `datetimezone`.

Пример №-1

Описание

Поиск начала квартала для 10 октября 2011 г., 8:00 (`#datetime(2011, 10, 10, 8, 0, 0)`).

Код

```
Date.StartOfQuarter(#datetime(2011, 10, 10, 8, 0, 0))
```

Результат

```
#datetime(2011, 10, 1, 0, 0, 0)
```

Date.StartOfWeek

Date.StartOfWeek(dateTime as any, optional firstDayOfWeek as number) as any

Описание

Возвращает первое значение недели для данного значения `date`, `datetime` или `datetimezone`.

Пример №-1

Описание

Поиск начала недели для 10 октября 2011 г., 8:10:32 (`#datetime(2011, 10, 10, 8, 10, 32)`).

Код

```
Date.StartOfWeek(#datetime(2011, 10, 10, 8, 10, 32))
```

Результат

```
#datetime(2011, 10, 9, 0, 0, 0)
```

Date.StartOfYear

Date.StartOfYear(dateTime as any) as any

Описание

Возвращает первое значение года для данного значения `date`, `datetime` или `datetimezone`.

Пример №-1

Описание

Поиск начала года для 10 октября 2011 г., 8:10:32 (`#datetime(2011, 10, 10, 8, 10, 32)`).

Код

```
Date.StartOfYear(#datetime(2011, 10, 10, 8, 10, 32))
```

Результат

```
#datetime(2011, 1, 1, 0, 0, 0)
```

Date.ToRecord

Date.ToRecord(date as date) as record

Описание

Возвращает запись, содержащую части заданного значения даты, `date`.

- `date`: значение `date`, для которого необходимо вычислить запись частей.

Пример №-1

Описание

Преобразовать значение `#date(2011, 12, 31)` в запись, содержащую состоит значения даты.

Код

```
Date.ToRecord(#date(2011, 12, 31))
```

Результат

```
[
      Year = 2011,
      Month = 12,
      Day = 31
]
```

Date.ToText

Date.ToText(date as date, optional format as text, optional culture as text) as text

Описание

Возвращает текстовое представление `date`, значения даты, `date`. Эта функция принимает необязательный параметр формата `format`. Полный список поддерживаемых форматов см. в спецификации библиотеки.

Пример №-1

Описание

Получение текстового представления #date (2010, 12, 31).

Код

```
Date.ToText(#date(2010, 12, 31))
```

Результат

```
"12/31/2010"
```

Пример №-2

Описание

Получение текстового представления #date(2010, 12, 31) с возможностью форматирования.

Код

```
Date.ToText(#date(2010, 12, 31), "yyyy/MM/dd")
```

Результат

```
"2010/12/31"
```

Date.WeekOfMonth

Date.WeekOfMonth(dateTime as any, optional firstDayOfWeek as number) as number

Описание

Возвращает число от 1 до 5, указывающее неделю в году, на которую приходится эта дата `dateTime`.

- `dateTime`: значение `datetime`, для которой определяется неделя в месяце.

Пример №-1

Описание

Определить, на какую неделю марта приходится 15 марта 2011 г. (`#date(2011, 03, 15)`).

Код

```
Date.WeekOfMonth(#date(2011, 03, 15))
```

Результат

```
3
```

Date.WeekOfYear

Date.WeekOfYear(dateTime as any, optional firstDayOfWeek as number) as number

Описание

Возвращает число от 1 до 54, указывающее неделю в году, на которую приходится дата `dateTime`.

- `dateTime`: значение `datetime`, для которой определяется неделя в году.

Пример №-1

Описание

Определить, на какую неделю в году приходится 23 марта 2011 г. (`#date(2011, 03, 23)`).

Код

```
Date.WeekOfYear(#date(2011, 03, 23))
```

Результат

```
13
```

Date.Year

Date.Year(dateTime as any) as number

Описание

Возвращает компонент года заданного значения `datetime`, `dateTime`.

Пример №-1

Описание

Поиск компонента года в значении #datetime(2011, 12, 31, 9, 15, 36).

Код

```
Date.Year(#datetime(2011, 12, 31, 9, 15, 36))
```

Результат

```
2011
```

DateTime

DateTime.AddZone

DateTime.AddZone(dateTime as datetime, timezoneHours as number, optional timezoneMinutes as number) as datetimezone

Описание

Задает сведения о часовом поясе в значении даты и времени `dateTime`. В сведения о часовом поясе входит `timezoneHours` и `timezoneMinutes` (второе не обязательно).

Пример №-1

Описание

Задает в поле сведений о часовом поясе для #datetime(2010, 12, 31, 11, 56, 02) значение "7 часов 30 минут".

Код

```
DateTime.AddZone(#datetime(2010, 12, 31, 11, 56, 02), 7, 30)
```

Результат

```
#datetimezone(2010, 12, 31, 11, 56, 2, 7, 30)
```

DateTime.Date

DateTime.Date(dateTime as any) as date

Описание

Возвращает компонент даты `dateTime` для указанного значения `date`, `datetime` или `datetimezone`.

Пример №-1

Описание

Поиск значение даты #datetime (2010, 12, 31, 11, 56, 02).

Код

```
DateTime.Date(#datetime(2010, 12, 31, 11, 56, 02))
```

Результат

```
#date(2010, 12, 31)
```

DateTime.FixedLocalNow

DateTime.FixedLocalNow() as datetime

Описание

Возвращает значение `datetime`, равное текущей дате и времени в системе. Это значение зафиксировано и не меняется при последующих вызовах в отличие от DateTime.LocalNow, которое может возвращать разные значения по мере выполнения выражения.

DateTime.From

DateTime.From(value as any, optional culture as text) as datetime

Описание

Возвращает значение `datetime` из указанного `value`. Если данное `value` имеет значение `null`, `DateTime.From` возвращает `null`. Если данное `value` - `datetime`, возвращается `value`. Значения следующих типов можно преобразовать в значение `datetime`:

- `text`: значение `datetime` из текстового представления. Дополнительные сведения см. в разделе `DateTime.FromText` .
- `date`: `datetime` с `value` в качестве компонента даты и `12:00:00 AM` в качестве компонента времени.
- `datetimezone`: `datetime`, локальный эквивалент `value`.
- `time`: `datetime` с эквивалентом даты OLE-автоматизации, `0` в качестве компонента даты и `value` в качестве компонента времени.
- `number`: `datetime`, эквивалент даты OLE-автоматизации выраженным `value`.

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Преобразовать `#date(1975, 4, 4)` в значение `datetime`.

Код

```
DateTime.From(#date(1975, 4, 4))
```

Результат

```
#datetime(1975, 4, 4, 0, 0, 0)
```

Пример №-2

Описание

Преобразовать `#time(06, 45, 12)` *в значение* `datetime`*.*

Код

```
DateTime.From(#time(06, 45, 12))
```

Результат

```
#datetime(1899, 12, 30, 06, 45, 12)
```

DateTime.FromFileTime

DateTime.FromFileTime(fileTime as number) as datetime

Описание

Создает значение `datetime` на основе значения `fileTime` и преобразует его в соответствии с местным часовым поясом. Значение filetime - время файла Windows, представляющее количество 100-наносекундных интервалов, прошедших с 12:00 1 января 1601 г. нашей эры (C.E.) по Гринвичу (UTC).

Пример №-1

Описание

Преобразование `129876402529842245` *в значение datetime.*

Код

```
DateTime.FromFileTime(129876402529842245)
```

Результат

```
#datetime(2012, 7, 24, 14, 50, 52.9842245)
```

DateTime.FromText

DateTime.FromText(text as text, optional culture as text) as datetime

Описание

Создает значение `datetime` из текстового представления `text`, согласно стандарту формата ISO 8601.

- `DateTime.FromText("2010-12-31T01:30:00")` // гггг-ММ-ддТчч:мм:сс

Пример №-1

Описание

Преобразование `"20101231T01:30:25"` в значение datetime.

Код

```
DateTime.FromText("20101231T01:30:25")
```

Результат

```
#datetime(2010, 12, 31, 1, 30, 25)
```

Пример №-2

Описание

Преобразование `"20101231T013025"` в значение datetime.

Код

```
DateTime.FromText("20101231T013025")
```

Результат

```
#datetime(2010, 12, 31, 1, 30, 25)
```

Пример №-3

Описание

Преобразование `"2010-12-31T01:30"` в значение datetime.

Код

```
DateTime.FromText("2010-12-31T01:30")
```

Результат

```
#datetime(2010, 12, 31, 1, 30, 0)
```

Пример №-4

Описание

Преобразование `"2010-12-31T01:30:25"` в значение datetime.

Код

```
DateTime.FromText("2010-12-31T01:30:25")
```

Результат

```
#datetime(2010, 12, 31, 1, 30, 25)
```

Пример №-5

Описание

Преобразование `"20101231T01:30:25.121212"` в значение datetime.

Код

```
DateTime.FromText("20101231T01:30:25.121212")
```

Результат

```
#datetime(2010, 12, 31, 1, 30, 25.121212)
```

DateTime.IsInCurrentHour

DateTime.IsInCurrentHour(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение текущего часа, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли текущее системное время в текущем часе.

Код

```
DateTime.IsInCurrentHour(DateTime.FixedLocalNow())
```

Результат

```
true
```

DateTime.IsInCurrentMinute

DateTime.IsInCurrentMinute(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение текущей минуты, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли текущее системное время в текущей минуте.

Код

```
DateTime.IsInCurrentMinute(DateTime.FixedLocalNow())
```

Результат

```
true
```

DateTime.IsInCurrentSecond

DateTime.IsInCurrentSecond(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение текущей секунды, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли текущее системное время в текущей секунде.

Код

```
DateTime.IsInCurrentSecond(DateTime.FixedLocalNow())
```

Результат

```
true
```

DateTime.IsInNextHour

DateTime.IsInNextHour(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение следующего часа, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли час, следующий за текущим системным временем, в следующем часе.

Код

```
DateTime.IsInNextHour(DateTime.FixedLocalNow() + #duration(0,1,0,0))
```

Результат

```
true
```

DateTime.IsInNextMinute

DateTime.IsInNextMinute(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение следующей минуты, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли минута, следующая за текущим системным временем, в следующей минуте.

Код

```
DateTime.IsInNextMinute(DateTime.FixedLocalNow() + #duration(0,0,1,0))
```

Результат

```
true
```

DateTime.IsInNextNHours

DateTime.IsInNextNHours(dateTime as any, hours as number) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение следующих нескольких часов, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.
- `hours`: количество часов.

Пример №-1

Описание

Определяет, находится ли час, следующий за текущим системным временем, в следующих двух часах.

Код

```
DateTime.IsInNextNHours(DateTime.FixedLocalNow() + #duration(0,2,0,0), 2)
```

Результат

```
true
```

DateTime.IsInNextNMinutes

DateTime.IsInNextNMinutes(dateTime as any, minutes as number) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение следующих нескольких минут, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.
- `minutes`: количество минут.

Пример №-1

Описание

Определяет, находится ли минута, следующая за текущим системным временем, в следующих двух минутах.

Код

```
DateTime.IsInNextNMinutes(DateTime.FixedLocalNow() + #duration(0,0,2,0), 2)
```

Результат

```
true
```

DateTime.IsInNextNSeconds

DateTime.IsInNextNSeconds(dateTime as any, seconds as number) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение следующих нескольких секунд, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.
- `seconds`: количество секунд.

Пример №-1

Описание

Определяет, находится ли секунда, следующая за текущим системным временем, в следующих двух секундах.

Код

```
DateTime.IsInNextNSeconds(DateTime.FixedLocalNow() + #duration(0,0,0,2), 2)
```

Результат

```
true
```

DateTime.IsInNextSecond

DateTime.IsInNextSecond(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение следующей секунды, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли секунда, следующая за текущим системным временем, в следующей секунде.

Код

```
DateTime.IsInNextSecond(DateTime.FixedLocalNow() + #duration(0,0,0,1))
```

Результат

```
true
```

DateTime.IsInPreviousHour

DateTime.IsInPreviousHour(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение предыдущего часа, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли час, предшествующий текущему системному времени, в предыдущем часе.

Код

```
DateTime.IsInPreviousHour(DateTime.FixedLocalNow() - #duration(0,1,0,0))
```

Результат

```
true
```

DateTime.IsInPreviousMinute

DateTime.IsInPreviousMinute(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение предыдущей минуты, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли минута, предшествующая текущему системному времени, в предыдущей минуте.

Код

```
DateTime.IsInPreviousMinute(DateTime.FixedLocalNow() - #duration(0,0,1,0))
```

Результат

```
true
```

DateTime.IsInPreviousNHours

DateTime.IsInPreviousNHours(dateTime as any, hours as number) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение предыдущих нескольких часов, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.
- `hours`: количество часов.

Пример №-1

Описание

Определяет, находится ли час, предшествующий текущему системному времени, в предыдущих двух часах.

Код

```
DateTime.IsInPreviousNHours(DateTime.FixedLocalNow() - #duration(0,2,0,0), 2)
```

Результат

```
true
```

DateTime.IsInPreviousNMinutes

DateTime.IsInPreviousNMinutes(dateTime as any, minutes as number) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение предыдущих нескольких минут, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.
- `minutes`: количество минут.

Пример №-1

Описание

Определяет, находится ли минута, предшествующая текущему системному времени, в предыдущих двух минутах.

Код

```
DateTime.IsInPreviousNMinutes(DateTime.FixedLocalNow() - #duration(0,0,2,0), 2)
```

Результат

```
true
```

DateTime.IsInPreviousNSeconds

DateTime.IsInPreviousNSeconds(dateTime as any, seconds as number) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение предыдущих нескольких секунд, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.
- `seconds`: количество секунд.

Пример №-1

Описание

Определяет, находится ли секунда, предшествующая текущему системному времени, в предыдущих двух секундах.

Код

```
DateTime.IsInPreviousNSeconds(DateTime.FixedLocalNow() - #duration(0,0,0,2), 2)
```

Результат

```
true
```

DateTime.IsInPreviousSecond

DateTime.IsInPreviousSecond(dateTime as any) as logical

Описание

Указывает, наступает ли данный момент времени (`dateTime`) в течение предыдущей секунды, по расчету на основе текущей системной даты и времени.

- `dateTime`: вычисляемое значение `datetime` или `datetimezone`.

Пример №-1

Описание

Определяет, находится ли секунда, предшествующая текущему системному времени, в предыдущей секунде.

Код

```
DateTime.IsInPreviousSecond(DateTime.FixedLocalNow() - #duration(0,0,0,1))
```

Результат

```
true
```

DateTime.LocalNow

DateTime.LocalNow() as datetime

Описание

Возвращает значение `datetime`, равное текущей дате и времени в системе.

DateTime.Time

DateTime.Time(dateTime as any) as time

Описание

Возвращает компонент времени заданного значения datetime, `dateTime`.

Пример №-1

Описание

Поиск значения времени #datetime (2010, 12, 31, 11, 56, 02).

Код

```
DateTime.Time(#datetime(2010, 12, 31, 11, 56, 02))
```

Результат

```
#time(11, 56, 2)
```

DateTime.ToRecord

DateTime.ToRecord(dateTime as datetime) as record

Описание

Возвращает запись, содержащую части заданного значения datetime, `dateTime`.

- `dateTime`: значение `datetime`, для которого необходимо вычислить запись частей.

Пример №-1

Описание

Преобразование значения `#datetime(2011, 12, 31, 11, 56, 2)` *в запись, содержащую значения даты и времени.*

Код

```
DateTime.ToRecord(#datetime(2011, 12, 31, 11, 56, 2))
```

Результат

```
[
      Year = 2011,
      Month = 12,
      Day = 31,
      Hour = 11,
      Minute = 56,
      Second = 2
]
```

DateTime.ToText

DateTime.ToText(dateTime as datetime, optional format as text, optional culture as text) as text

Описание

Возвращает текстовое представление значения `dateTime`, значения datetime, `dateTime`. Эта функция принимает необязательный параметр формата `format`. Полный список поддерживаемых форматов см. в спецификации библиотеки.

Пример №-1

Описание

Получение текстового представления #datetime (2011, 12, 31, 11, 56, 2).

Код

```
DateTime.ToText(#datetime(2010, 12, 31, 11, 56, 2))
```

Результат

```
"12/31/2010 11:56:02 AM"
```

Пример №-2

Описание

Получение текстового представления #datetime(2011, 12, 31, 11, 56, 2) с возможностью форматирования.

Код

```
DateTime.ToText(#datetime(2010, 12, 31, 11, 56, 2), "yyyy/MM/ddThh:mm:ss")
```

Результат

```
"2010/12/31T11:56:02"
```

DateTimeZone

DateTimeZone.FixedLocalNow

DateTimeZone.FixedLocalNow() as datetimezone

Описание

Возвращает значение `datetime`, равное текущей дате и времени в системе. Возвращенное значение содержит сведения о часовом поясе, представляющем местный часовой пояс. Это значение зафиксировано и не меняется при последующих вызовах в отличие от DateTimeZone.LocalNow, которое может возвращать разные значения по мере выполнения выражения.

DateTimeZone.FixedUtcNow

DateTimeZone.FixedUtcNow() as datetimezone

Описание

Возвращает текущую дату и время в формате UTC (часовой пояс GMT). Это значение зафиксировано и не меняется при последующих вызовах.

DateTimeZone.From

DateTimeZone.From(value as any, optional culture as text) as datetimezone

Описание

Возвращает значение `datetimezone` из указанного `value`. Если данное `value` имеет значение `null`, `DateTimeZone.From` возвращает `null`. Если данное `value` - `datetimezone`, возвращается `value`. Значения следующих типов можно преобразовать в значение `datetimezone`:

- `text`: значение `datetimezone` из текстового представления. Дополнительные сведения см. в разделе `DateTimeZone.FromText`.
- `date`: `datetimezone` с `value` в качестве компонента даты, `12:00:00 AM` в качестве компонента времени и смещения, соответствующего местному часовому поясу.
- `datetime`: `datetimezone` с `value` в качестве даты, времени и смещения, соответствующего местному часовому поясу.
- `time`: `datetimezone` с эквивалентом даты OLE-автоматизации, `0` в качестве компонента даты, `value` в качестве компонента времени и смещения, соответствующего местному часовому поясу.

- `number`: `datetimezone` с эквивалентом даты и времени даты OLE-автоматизации, выраженным `value`, и смещением, соответствующим местному часовому поясу.

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Преобразовать `"2020-10-30T01:30:00-08:00"` в значение `datetimezone`.

Код

```
DateTimeZone.From("2020-10-30T01:30:00-08:00")
```

Результат

```
#datetimezone(2020, 10, 30, 01, 30, 00, -8, 00)
```

DateTimeZone.FromFileTime

DateTimeZone.FromFileTime(fileTime as number) as datetimezone

Описание

Создает значение `datetimezone` на основе значения `fileTime` и преобразует его в соответствии с местным часовым поясом. Значение filetime — время файла Windows, представляющее количество 100-наносекундных интервалов, прошедших с 12:00 1 января 1601 г. нашей эры (C.E.) по Гринвичу (UTC).

Пример №-1

Описание

Преобразование `129876402529842245` в значение datetimezone.

Код

```
DateTimeZone.FromFileTime(129876402529842245)
```

Результат

```
#datetimezone(2012, 7, 24, 14, 50, 52.9842245, -7, 0)
```

DateTimeZone.FromText

DateTimeZone.FromText(text as text, optional culture as text) as datetimezone

Описание

Создает значение `datetimezone` из текстового представления `text` согласно стандарту формата ISO 8601.

- `DateTimeZone.FromText("2010-12-31T01:30:00-08:00")` // гггг-ММ-ддТчч:мм:ссZ

Пример №-1

Описание

Преобразование `"2010-12-31T01:30:00-08:00"` в значение datetimezone.

Код

```
DateTimeZone.FromText("2010-12-31T01:30:00-08:00")
```

Результат

```
#datetimezone(2010, 12, 31, 1, 30, 0, -8, 0)
```

Пример №-2

Описание

Преобразование `"2010-12-31T01:30:00.121212-08:00"` в значение datetimezone.

Код

```
DateTimeZone.FromText("2010-12-31T01:30:00.121212-08:00")
```

Результат

```
#datetimezone(2010, 12, 31, 1, 30, 0.121212, -8, 0)
```

Пример №-3

Описание

Преобразование `"2010-12-31T01:30:00Z"` в значение datetimezone.

Код

```
DateTimeZone.FromText("2010-12-31T01:30:00Z")
```

Результат

```
#datetimezone(2010, 12, 31, 1, 30, 0, 0, 0)
```

Пример №-4

Описание

Преобразование `"20101231T013000+0800"` в значение datetimezone.

Код

```
DateTimeZone.FromText("20101231T013000+0800")
```

Результат

```
#datetimezone(2010, 12, 31, 1, 30, 0, 8, 0)
```

DateTimeZone.LocalNow

DateTimeZone.LocalNow() as datetimezone

Описание

Возвращает значение `datetimezone`, равное текущей дате и времени в системе. Возвращенное значение содержит сведения о часовом поясе, представляющем местный часовой пояс.

DateTimeZone.RemoveZone

DateTimeZone.RemoveZone(dateTimeZone as datetimezone) as datetime

Описание

Возвращает значение #datetime из `dateTimeZone` с удаленными данными часового пояса.

Пример №-1

Описание

Удалить данные часового пояса из значения #datetimezone (2011, 12, 31, 9, 15, 36, -7, 0).

Код

```
DateTimeZone.RemoveZone( #datetimezone(2011, 12, 31, 9, 15, 36,-7, 0))
```

Результат

```
#datetime(2011, 12, 31, 9, 15, 36)
```

DateTimeZone.SwitchZone

DateTimeZone.SwitchZone(dateTimeZone as datetimezone, timezoneHours as number, optional timezoneMinutes as number) as datetimezone

Описание

Изменяет данные часового пояса в значении datetimezone `dateTimeZone` на новые данные о часовом поясе из `timezoneHours` и при необходимости `timezoneMinutes`. Если `dateTimeZone` не содержит компонент часового пояса, возникает исключение.

Пример №-1

Описание

Изменяет данные о часовом поясе для #datetimezone (2010, 12, 31, 11, 56, 02, 7, 30) на -30 минут.

Код

```
DateTimeZone.SwitchZone(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30), 0, -30)
```

Результат

```
#datetimezone(2010, 12, 31, 3, 56, 2, 0, -30)
```

Пример №-2

Описание

Изменяет данные о часовом поясе для #datetimezone (2010, 12, 31, 11, 56, 02, 7, 30) на 8 часов.

Код

```
DateTimeZone.SwitchZone(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30), 8)
```

Результат

```
#datetimezone(2010, 12, 31, 12, 26, 2, 8, 0)
```

DateTimeZone.ToLocal

DateTimeZone.ToLocal(dateTimeZone as datetimezone) as datetimezone

Описание

Изменяет данные о часовом поясе значения datetimezone `dateTimeZone` на локальный часовой пояс. Если `dateTimeZone` не содержит компонента часового пояса, будут добавлены данные локальные часового пояса.

Пример №-1

Описание

Изменение сведений о часовом поясе для #datetimezone (2010, 12, 31, 11, 56, 02, 7, 30) на локальный часовой пояс (по тихоокеанскому времени).

Код

```
DateTimeZone.ToLocal(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30))
```

Результат

```
#datetimezone(2010, 12, 31, 12, 26, 2, -8, 0)
```

DateTimeZone.ToRecord

DateTimeZone.ToRecord(dateTimeZone as datetimezone) as record

Описание

Возвращает запись, содержащую части заданного значения datetimezone, `dateTimeZone`.

- `dateTimeZone`: значение `datetimezone`, для которого необходимо вычислить запись частей.

Пример №-1

Описание

Преобразование значения `#datetimezone(2011, 12, 31, 11, 56, 2, 8, 0)` в запись, содержащую значения даты, времени и часового пояса.

Код

```
DateTimeZone.ToRecord(#datetimezone(2011, 12, 31, 11, 56, 2, 8, 0))
```

Результат

```
[
      Year = 2011,
      Month = 12,
      Day = 31,
      Hour = 11,
      Minute = 56,
      Second = 2,
      ZoneHours = 8,
      ZoneMinutes = 0
]
```

DateTimeZone.ToText

DateTimeZone.ToText(dateTimeZone as datetimezone, optional format as text, optional culture as text) as text

Описание

Возвращает текстовое представление `dateTimeZone`, значения datetimezone `dateTimeZone`. Эта функция принимает необязательный параметр формата `format`. Полный список поддерживаемых форматов см. в спецификации библиотеки.

Пример №-1

Описание

Получение текстового представления #datetimezone(2010, 12, 31, 11, 56, 2, 10, 12) с возможностью форматирования.

Код

```
DateTimeZone.ToText(#datetimezone(2010, 12, 31, 11, 56, 2, 10, 12),
"yyyy/MM/ddThh:mm:sszzz")
```

Результат

```
"2010/12/31T11:56:02+10:12"
```

Пример №-2

Описание

Получение текстового представления #datetimezone (2011, 12, 31, 11, 56, 2, 8, 0).

Код

```
DateTimeZone.ToText(#datetimezone(2010, 12, 31, 11, 56, 2, 8, 0))
```

Результат

```
"12/31/2010 11:56:02 AM +08:00"
```

DateTimeZone.ToUtc

DateTimeZone.ToUtc(dateTimeZone as datetimezone) as datetimezone

Описание

Изменяет данные о часовом значения даты и времени `dateTimeZone` в данные часового пояса в формате UTC или всемирного времени. Если `dateTimeZone` не содержит компонента часового пояса, будут добавлены данные часового пояса UTC.

Пример №-1

Описание

Изменяет данные о часовом поясе для #datetimezone (2010, 12, 31, 11, 56, 02, 7, 30) на UTC.

Код

```
DateTimeZone.ToUtc(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30))
```

Результат

```
#datetimezone(2010, 12, 31, 4, 26, 2, 0, 0)
```

DateTimeZone.UtcNow

DateTimeZone.UtcNow() as datetimezone

Описание

Возвращает текущую дату и время в формате UTC (часовой пояс GMT).

Пример №-1

Описание

Возвращает текущую дату и время в формате UTC.

Код

```
DateTimeZone.UtcNow()
```

Результат

```
#datetimezone(2011, 8, 16, 23, 34, 37.745, 0, 0)
```

DateTimeZone.ZoneHours

DateTimeZone.ZoneHours(dateTimeZone as datetimezone) as number

Описание

Изменяет часовой пояс значения.

DateTimeZone.ZoneMinutes

DateTimeZone.ZoneMinutes(dateTimeZone as datetimezone) as number

Описание

Изменяет часовой пояс значения.

Decimal

Decimal.From

Decimal.From(value as any, optional culture as text) as number

Описание

Возвращает значение Decimal `number` из данного значения `value`. Если данное значение `value` является значением `null`, `Decimal.From` возвращает `null`. Если данное значение `value` является значением `number` в диапазоне Decimal, возвращается `value`, иначе возвращается ошибка. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в значение Decimal `number`.

Пример №-1

Описание

Получить значение Decimal `number` *для* `"4.5"`*.*

Код

```
Decimal.From("4.5")
```

Результат

```
4.5
```

Diagnostics

Diagnostics.ActivityId

Diagnostics.ActivityId() as text

Описание

Возвращает непрозрачный идентификатор для текущего вычисления.

Diagnostics.Trace

Diagnostics.Trace(traceLevel as number, message as text, value as any, optional delayed as logical) as any

Описание

Записывает трассировку `message`, если трассировка включена, и возвращает `value`. Необязательный параметр `delayed` указывает, следует ли откладывать оценку `value` до трассировки сообщения. `traceLevel` может иметь одно из следующих значений: `TraceLevel.Critical`, `TraceLevel.Error`, `TraceLevel.Warning`, `TraceLevel.Information`, `TraceLevel.Verbose`.

Пример №-1

Описание

Трассировка сообщения перед вызовом функции Text.From и возврат результата.

Код

```
Diagnostics.Trace(TraceLevel.Information, "TextValueFromNumber", () => Text.From(123), true)
```

Результат

```
"123"
```

DirectQueryCapabilities

DirectQueryCapabilities.From

DirectQueryCapabilities.From(value as any) as table

Описание

DirectQueryCapabilities.From

Double

Double.From

Double.From(value as any, optional culture as text) as number

Описание

Возвращает значение Double `number` из данного значения `value`. Если данное значение `value` является значением `null`, `Double.From` возвращает `null`. Если данное значение `value` является значением `number` в диапазоне Double, возвращается `value`, иначе возвращается ошибка. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в значение Double `number`.

Пример №-1

Описание

Получить значение Double `number` для `"4"`.

Код

```
Double.From("4.5")
```

Результат

```
4.5
```

Duration

Duration.Days

Duration.Days(duration as duration) as number

Описание

Возвращает компонент дня заданного значения `duration`, `duration`.

Пример №-1

Описание

Поиск компонента дня в значении #duration(5, 4, 3, 2).

Код

```
Duration.Days(#duration(5, 4, 3, 2))
```

Результат

```
5
```

Duration.From

Duration.From(value as any) as duration

Описание

Возвращает значение `duration` из указанного `value`. Если данное `value` имеет значение `null`, `Duration.From` возвращает `null`. Если данное `value` - `duration`, возвращается `value`. Значения следующих типов можно преобразовать в значение `duration`:

- `text`: значение `duration` из текстового формата прошедшего времени (d.h:m:s). Дополнительные сведения см. в разделе `Duration.FromText` .
- `number`: значение `duration`, эквивалентное числу полных и неполных дней, выраженное `value`.

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Преобразование `2.525` в значение `duration`.

Код

```
Duration.From(2.525)
```

Результат

```
#duration(2, 12, 36, 0)
```

Duration.FromText

Duration.FromText(text as text) as duration

Описание

Возвращает значение длительности из указанного текста, `text`. Эта функция принимает следующие форматы :

- (-)hh:mm(:ss(.ff))
- (-)ddd(.hh:mm(:ss(.ff)))

(все диапазоны задаются с включением границ)
ddd: число дней.
hh: число часов от 0 до 23.
mm: число минут от 0 до 59.
ss: число секунд от 0 до 59.
ff: число долей секунды от 0 до 9999999.

Пример №-1

Описание

Преобразование `"2.05:55:20"` в значение `duration`.

Код

```
Duration.FromText("2.05:55:20")
```

Результат

```
#duration(2, 5, 55, 20)
```

Duration.Hours

Duration.Hours(duration as duration) as number

Описание

Возвращает компонент часов заданного значения `duration`, `duration`.

Пример №-1

Описание

Поиск компонента часов в значении #duration(5, 4, 3, 2).

Код

```
Duration.Hours(#duration(5, 4, 3, 2))
```

Результат

```
4
```

Duration.Minutes

Duration.Minutes(duration as duration) as number

Описание

Возвращает компонент минут заданного значения `duration`, `duration`.

Пример №-1

Описание

Поиск компонента минут в значении #duration(5, 4, 3, 2).

Код

```
Duration.Minutes(#duration(5, 4, 3, 2))
```

Результат

```
3
```

Duration.Seconds

Duration.Seconds(duration as duration) as number

Описание

Возвращает компонент секунд заданного значения `duration`, `duration`.

Пример №-1

Описание

Поиск компонента секунд в значении #duration(5, 4, 3, 2).

Код

```
Duration.Seconds(#duration(5, 4, 3, 2))
```

Результат

```
2
```

Duration.ToRecord

Duration.ToRecord(duration as duration) as record

Описание

Возвращает запись, содержащую части длительности значения `duration`.

- `duration`: значение `duration`, на основе которого создается запись.

Пример №-1

Описание

Преобразование `#duration(2, 5, 55, 20)` *в запись его частей, включая дни, часы, минуты и секунды, если применимо.*

Код

```
Duration.ToRecord(#duration(2, 5, 55, 20))
```

Результат

```
[Days = 2,
    Hours = 5,
    Minutes = 55,
    Seconds = 20]
```

Duration.ToText

Duration.ToText(duration as duration, optional format as text) as text

Описание

Возвращает текстовое представление в виде "day.hour:mins:sec" для данного значения длительности `duration`. Текстовое значение, указывающее формат, который может быть задан как необязательный второй параметр `format`.

- `duration`: значение `duration`, для которого формируется текстовое представление.
- `format`: *[Необязательно]* Значение `text`, которое указывает формат.

Пример №-1

Описание

Преобразовать `#duration(2, 5, 55, 20)` *в текстовое значение.*

Код

```
Duration.ToText(#duration(2, 5, 55, 20))
```

Результат

```
"2.05:55:20"
```

Duration.TotalDays

Duration.TotalDays(duration as duration) as number

Описание

Возвращает общее количество дней, охваченных значением `duration`, `duration`.

Пример №-1

Описание

Поиск общего количества дней, охваченных #duration(5, 4, 3, 2).

Код

```
Duration.TotalDays(#duration(5, 4, 3, 2))
```

Результат

```
5.1687731481481478
```

Duration.TotalHours

Duration.TotalHours(duration as duration) as number

Описание

Возвращает общее количество часов, охваченных значением `duration`, `duration`.

Пример №-1

Описание

Поиск общего количества часов, охваченных #duration(5, 4, 3, 2).

Код

```
Duration.TotalHours(#duration(5, 4, 3, 2))
```

Результат

```
124.05055555555555
```

Duration.TotalMinutes

Duration.TotalMinutes(duration as duration) as number

Описание

Возвращает общее количество минут, охваченных значением `duration`, `duration`.

Пример №-1

Описание

Поиск общего количества минут, охваченных #duration(5, 4, 3, 2).

Код

```
Duration.TotalMinutes(#duration(5, 4, 3, 2))
```

Результат

```
7443.0333333333338
```

Duration.TotalSeconds

Duration.TotalSeconds(duration as duration) as number

Описание

Возвращает общее количество секунд, охваченных значением `duration`, `duration`.

Пример №-1

Описание

Поиск общего количества секунд, охваченных #duration(5, 4, 3, 2).

Код

```
Duration.TotalSeconds(#duration(5, 4, 3, 2))
```

Результат

```
446582
```

Embedded

Embedded.Value

Embedded.Value(value as any, path as text) as any

Описание

Доступ к значению по имени во внедренном гибридном веб-приложении.

Error

Error.Record

Error.Record(reason as text, optional message as text, optional detail as any) as record

Описание

Возвращает ошибочную запись из указанных текстовых значений для причины, сообщения и сведений.

Excel

Excel.CurrentWorkbook

Excel.CurrentWorkbook() as table

Описание

Возвращает таблицы в текущей книге Excel.

Excel.Workbook

Excel.Workbook(workbook as binary, optional useHeaders as logical, optional delayTypes as logical) as table

Описание

Возвращает запись из листов книги Excel.

Exchange

Exchange.Contents

Exchange.Contents(optional mailboxAddress as text) as table

Описание

Возвращает содержание из учетной записи "`mailboxAddress`" Microsoft Exchange. Если "`mailboxAddress`" не указана, будет использоваться учетная запись по умолчанию для учетных данных.

Expression

Expression.Constant

Expression.Constant(value as any) as text

Описание

Expression.Constant

Expression.Evaluate

Expression.Evaluate(document as text, optional environment as record) as any

Описание

Expression.Evaluate

Expression.Identifier

Expression.Identifier(name as text) as text

Описание

Expression.Identifier

Facebook

Facebook.Graph

Facebook.Graph(url as text) as any

Описание

Возвращает запись, содержащую набор таблиц, найденных в диаграмме Facebook по указанному URL-адресу, `url`.

File

File.Contents

File.Contents(path as text) as binary

Описание

Возвращает содержимое файла `path` в двоичном виде.

Folder

Folder.Contents

Folder.Contents(path as text) as table

Описание

Возвращает таблицу, содержащую одну строку для каждой папки и файла, найденных в папке `path`. Каждая строка представляет свойства файла или папки и ссылку на содержимое.

Folder.Files

Folder.Files(path as text) as table

Описание

Возвращает таблицу, содержащую одну строку для каждой папки и файла, найденных в папке `path` и во вложенных папках. Каждая строка содержит свойства файла и ссылки на его содержимое.

Function

Function.Invoke

Function.Invoke(function as function, args as list) as any

Описание

Вызывает данную функцию, используя указанный список аргументов, и возвращает результат.

Пример №-1

Описание

Вызывает Record.FieldNames с одним аргументом [A=1,B=2]

Код

```
Function.Invoke(Record.FieldNames, {[A=1,B=2]})
```

Результат

```
{  "A", "B" }
```

Function.InvokeAfter

Function.InvokeAfter(function as function, delay as duration) as any

Описание

Возвращает результат вызова `function` спустя период времени `delay`.

Function.IsDataSource

Function.IsDataSource(function as function) as logical

Описание

Определяет, считается ли `function` источником данных.

HdInsight

HdInsight.Containers

HdInsight.Containers(account as text) as table

Описание

Возвращает навигационную таблицу, содержащую одну строку для каждого контейнера, найденного по URL-адресу учетной записи `account` в хранилище Azure. Каждая строка содержит ссылку на BLOB-объекты в контейнере.

HdInsight.Contents

HdInsight.Contents(account as text) as table

Описание

Возвращает навигационную таблицу, содержащую одну строку для каждого контейнера, найденного по URL-адресу учетной записи `account` в хранилище Azure. Каждая строка содержит ссылку на BLOB-объекты в контейнере.

HdInsight.Files

HdInsight.Files(account as text, containerName as text) as table

Описание

Возвращает таблицу, содержащую строку для каждого BLOB-файла, обнаруженного по URL-адресу контейнера, `account`, из хранилища Azure. Каждая строка содержит свойства файла и ссылку на его содержимое.

Hdfs

Hdfs.Contents

Hdfs.Contents(url as text) as table

Описание

Возвращает таблицу, содержащую одну строку для каждой папки и каждого файла, найденного по URL-адресу папки `url` из файловой системы Hadoop. Каждая строка представляет свойства файла или папки и ссылку на содержимое.

Hdfs.Files

Hdfs.Files(url as text) as table

Описание

Возвращает таблицу, содержащую одну строку для каждого файла, найденного по URL-адресу папки `url` и во вложенных папках из файловой системы Hadoop. Каждая строка содержит свойства файла и ссылки на его содержимое.

Informix

Informix.Database

Informix.Database(server as text, database as text, optional options as record) as table

Описание

Возвращает список таблиц и представлений SQL, доступных в базе данных Informix на сервере `server` в экземпляре базы данных `database`. Дополнительно к имени сервера через двоеточие может быть указан порт. Необязательный параметр записи `options` может быть указан для управления следующими параметрами:

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

Int16

Int16.From

Int16.From(value as any, optional culture as text, optional roundingMode as number) as number

Описание

Возвращает 16-разрядное целое значение `number` из данного значения `value`. Если данное значение `value` является значением `null`, `Int16.From` возвращает `null`. Если данное значение `value` является значением `number` в диапазоне 16-разрядных целых чисел без дробной части, возвращается `value`. Если есть дробная часть, число округляется в заданном режиме. Режим округления по умолчанию — `RoundingMode.ToEven`. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в 16-разрядное целое значение `number`. См. в описании `Number.Round` допустимые режимы округления.

Пример №-1

Описание

Получить 16-разрядное целое значение `number` для `"4.5"`, используя `RoundingMode.AwayFromZero`.

Код

```
Int16.From("4.5", null, RoundingMode.AwayFromZero)
```

Результат

```
5
```

Пример №-2

Описание

Получить 16-разрядное целое значение `number` *для* `"4"`*.*

Код

```
Int64.From("4")
```

Результат

```
4
```

Int32

Int32.From

Int32.From(value as any, optional culture as text, optional roundingMode as number) as number

Описание

Возвращает 32-разрядное целое значение `number` из данного значения `value`. Если данное значение `value` является значением `null`, `Int32.From` возвращает `null`. Если данное значение `value` является значением `number` в диапазоне 32-разрядных целых чисел без дробной части, возвращается `value`. Если есть дробная часть, число округляется в заданном режиме. Режим округления по умолчанию — `RoundingMode.ToEven`. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в 32-разрядное целое значение `number`. См. в описании `Number.Round` допустимые режимы округления.

Пример №-1

Описание

Получить 32-разрядное целое значение `number` *для* `"4.5"`*, используя* `RoundingMode.AwayFromZero`*.*

Код

```
Int32.From("4.5", null, RoundingMode.AwayFromZero)
```

Результат

```
5
```

Пример №-2

Описание

Получить 32-разрядное целое значение `number` *для* `"4"`*.*

Код

```
Int32.From("4")
```

Результат

```
4
```

Int64

Int64.From

Int64.From(value as any, optional culture as text, optional roundingMode as number) as number

Описание

Возвращает 64-разрядное целое значение `number` из данного значения `value`. Если данное значение `value` является значением `null`, `Int64.From` возвращает `null`. Если данное значение `value` является значением `number` в диапазоне 64-разрядных целых чисел без дробной части, возвращается `value`. Если есть дробная часть, число округляется в заданном режиме. Режим округления по умолчанию — `RoundingMode.ToEven`. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в 64-разрядное целое значение `number`. См. в описании `Number.Round` допустимые режимы округления.

Пример №-1

Описание

Получить 64-разрядное целое значение `number "4"`.

Код

```
Int64.From("4")
```

Результат

```
4
```

Пример №-2

Описание

Получим 64-разрядное целочисленное значение `number` для `"4.5"`, используя `RoundingMode.AwayFromZero`.

Код

```
Int64.From("4.5", null, RoundingMode.AwayFromZero)
```

Результат

```
5
```

Int8

Int8.From

Int8.From(value as any, optional culture as text, optional roundingMode as number) as number

Описание

Возвращает 8-разрядное целое значение `number` со знаком из данного значения `value`. Если данное значение `value` является значением `null`, `Int8.From` возвращает `null`. Если данное значение `value` является значением `number` в диапазоне 8-разрядных целых чисел со знаком без дробной части, возвращается `value`. Если есть дробная часть, число округляется в заданном режиме. Режим округления по умолчанию — `RoundingMode.ToEven`. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в 8-разрядное целое значение `number` со знаком. См. в описании `Number.Round` допустимые режимы округления.

Пример №-1

Описание

Получить 8-разрядное целое значение `number` со знаком для `"4.5"`, используя `RoundingMode.AwayFromZero`.

Код

```
Int8.From("4.5", null, RoundingMode.AwayFromZero)
```

Результат

```
5
```

Пример №-2

Описание

Получить 8-разрядное целое значение `number` со знаком для `"4"`.

Код

```
Int8.From("4")
```

Результат

```
4
```

Json

Json.Document

Json.Document(jsonText as any, optional encoding as number) as any

Описание

Возвращает содержимое документа JSON.

Json.FromValue

Json.FromValue(value as any, optional encoding as number) as binary

Описание

Создает представление JSON заданного значения (`value`) с кодировкой текста, указанной "`encoding`". Если "`encoding`" пропускается, используется UTF8. Значения представлены следующим образом:

- Значения NULL, текстовые и логические значения представлены как соответствующие типы JSON.
- Числа представлены как числа в JSON, за исключением `#infinity`, `-#infinity` и `#nan`, которые преобразуются в NULL.
- Списки представлены как массивы JSON.
- Записи представлены как объекты JSON.
- Таблицы представлены как массив объектов.
- Значения date, time, datetime, datetimezone и duration представлены как текст ISO-8601.
- Двоичные значения представлены как текст в кодировке base-64.
- Типы и функции создают ошибку.

Пример №-1

Описание

Преобразование сложного значения в JSON.

Код

```
Text.FromBinary(Json.FromValue([A={1, true, "3"}, B=#date(2012, 3, 25)]))
```

Результат

```
"{""A"":[1,true,""3""],""B"":""2012-03-25""}"
```

Lines

Lines.FromBinary

Lines.FromBinary(binary as binary, optional quoteStyle as number, optional includeLineSeparators as logical, optional encoding as number) as list

Описание

Преобразует двоичное значение в список текстовых значений, разделенных разрывами строк. Если указан стиль кавычек, разрывы строк могут отображаться между кавычками. Если значение includeLineSeparators равно true, то символы разрыва строк включаются в текст.

Lines.FromText

Lines.FromText(text as text, optional quoteStyle as number, optional includeLineSeparators as logical) as list

Описание

Преобразует текстовое значение в список текстовых значений, разделенных разрывами строк. Если значение includeLineSeparators равно true, то символы разрыва строки включаются в текст.

- `QuoteStyle.None`: (по умолчанию) обработка специальных символов не требуется.
- `QuoteStyle.Csv`: обработка специальных символов как для формата CSV. Символ двойной кавычки используется для разграничения таких областей, пара символов двойной кавычки используется для представления одного символа двойной кавычки в пределах такой области.

Lines.ToBinary

Lines.ToBinary(lines as list, optional lineSeparator as text, optional encoding as number, optional includeByteOrderMark as logical) as binary

Описание

Преобразует список текстовых значений в двоичное значение с использованием указанной кодировки и разделителя lineSeparator. Указанный lineSeparator добавляется к каждой строке. Если разделитель не указан, используются символы возврата каретки и перевода строки.

Lines.ToText

Lines.ToText(lines as list, optional lineSeparator as text) as text

Описание

Преобразует список текстовых значений в одну текстовую строку. Указанный разделитель lineSeparator добавляется к каждой строке. Если разделитель не указан, используются символы возврата каретки и перевода строки.

List

List.Accumulate

List.Accumulate(list as list, seed as any, accumulator as function) as any

Описание

Накапливает сводное значения из элементов в списке `list` с использованием `accumulator`. Может быть установлен необязательный параметр начального значения, `seed`.

Пример №-1

Описание

Накапливает сводное значения из элементов в списке {1, 2, 3, 4, 5}, используя выражение ((состояние, текущее значение) => состояние + текущее значение).

Код

```
List.Accumulate({1, 2, 3, 4, 5}, 0, (state, current) => state + current)
```

Результат

```
15
```

List.AllTrue

List.AllTrue(list as list) as logical

Описание

Возвращает значение true, если все выражения в списке `list` равны true.

Пример №-1

Описание

Определить, все ли выражения в списке {true, true, 2 > 0} истинны.

Код

```
List.AllTrue({true, true, 2 > 0})
```

Результат

```
true
```

Пример №-2

Описание

Определить, все ли выражения в списке {true, true, 2 < 0} истинны.

Код

```
List.AllTrue({true, false, 2 < 0})
```

Результат

```
false
```

List.Alternate

List.Alternate(list as list, count as number, optional repeatInterval as number, optional offset as number) as list

Описание

Возвращает список, состоящий из всех нечетных элементов смещения в списке. Попеременно принимает и пропускает значения из списка `list` в зависимости от параметров.

- `count`: указывает число значений, которые пропускаются каждый раз.
- `repeatInterval`: необязательный интервал повтора, указывающий, сколько значений добавляются между пропущенными значениями.
- `offset`: параметр смещения, с которого начинается пропуск значений.

Пример №-1

Описание

Создать список из {1..10}, который начинается с 1, пропускает одно значение, сохраняет два значения и т. д.

Код

```
List.Alternate({1..10}, 1, 2, 1)
```

Результат

```
{1, 3, 4, 6, 7, 9, 10}
```

Пример №-2

Описание

Создать список из {1..10}, в котором пропускается каждое второе значение.

Код

```
List.Alternate({1..10}, 1, 1)
```

Результат

```
{2, 4, 6, 8, 10}
```

Пример №-3

Описание

Создать список из {1..10}, который начинается с 1 и пропускает каждое второе значение.

Код

```
List.Alternate({1..10}, 1, 1, 1)
```

Результат

```
{1, 3, 5, 7, 9}
```

Пример №-4

Описание

Создать список из {1..10}, в котором первое число пропускается.

Код

```
List.Alternate({1..10}, 1)
```

Результат

```
{2, 3, 4, 5, 6, 7, 8, 9, 10}
```

List.AnyTrue

List.AnyTrue(list as list) as logical

Описание

Возвращает значение true, если любое выражение в списке `list` истинно.

Пример №-1

Описание

Определить, есть ли в списке {2 = 0, false, 2 < 0} истинные выражения.

Код

```
List.AnyTrue({2 = 0, false, 2 < 0})
```

Результат

```
false
```

Пример №-2

Описание

Определить, есть ли в списке {true, false, 2 > 0} истинные выражения.

Код

```
List.AnyTrue({true, false, 2>0})
```

Результат

```
true
```

List.Average

List.Average(list as list, optional precision as number) as any

Описание

Возвращает среднее значение для элементов в списке `list`. Результат указывается с тем же типом данных, что и значения в списке. Работает только с числовыми значениями, значениями даты, времени, datetime, datetimezone и длительности. Если список пуст, возвращается значение NULL.

Пример №-1

Описание

Найдите среднее значение из дат: 1 января 2011 г., 2 января 2011 г. и 3 января 2011 г.

Код

```
List.Average({#date(2011, 1, 1), #date(2011, 1, 2), #date(2011, 1, 3)})
```

Результат

```
#date(2011, 1, 2)
```

Пример №-2

Описание

Найти среднее значение списка чисел `{3, 4, 6}`.

Код

```
List.Average({3, 4, 6})
```

Результат

```
4.333333333333333
```

List.Buffer

List.Buffer(list as list) as list

Описание

Помещает список `list` в буфер в памяти. Результат вызова - стабильный список.

Пример №-1

Описание

Создание стабильной копии списка {1..10}.

Код

```
List.Buffer({1..10})
```

Результат

```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

List.Combine

List.Combine(lists as list) as list

Описание

Принимает список списков `lists` и объединяет их в один новый список.

Пример №-1

Описание

Объединение двух простых списков {1, 2} и {3, 4}.

Код

```
List.Combine({{1, 2}, {3, 4}})
```

Результат

```
{
    1,
    2,
    3,
    4
}
```

Пример №-2

Описание

Объединение двух списков {1, 2} и {3, {4, 5}}, один из которых содержит вложенный список.

Код

```
List.Combine({{1, 2}, {3, {4, 5}}})
```

Результат

```
{
    1,
```

```
    2,
    3, {4,
        5}
}
```

List.Contains

List.Contains(list as list, value as any, optional equationCriteria as any) as logical

Описание

Указывает, содержит ли список `list` значение `value`. Возвращает значение true, если значение найдено в списке, и значение false в противном случае. Необязательное значение критерия уравнения `equationCriteria` можно указать для управления проверкой на равенство.

Пример №-1

Описание

Определить, содержит ли список {1, 2, 3, 4, 5} значение 3.

Код

```
List.Contains({1, 2, 3, 4, 5}, 3)
```

Результат

```
true
```

Пример №-2

Описание

Определить, содержит ли список {1, 2, 3, 4, 5} значение 6.

Код

```
List.Contains({1, 2, 3, 4, 5}, 6)
```

Результат

```
false
```

List.ContainsAll

List.ContainsAll(list as list, values as list, optional equationCriteria as any) as logical

Описание

Указывает, содержит ли список `list` все значения из другого списка `values`. Возвращает значение true, если значение найдено в списке, и значение false в противном случае. Необязательное значение критерия уравнения `equationCriteria` можно указать для управления проверкой на равенство.

Пример №-1

Описание

Определить, содержит ли список {1, 2, 3, 4, 5} значения 3 и 4.

Код

```
List.ContainsAll({1, 2, 3, 4, 5}, {3, 4})
```

Результат

```
true
```

Пример №-2

Описание

Определить, содержит ли список {1, 2, 3, 4, 5} значения 5 и 6.

Код

```
List.ContainsAll({1, 2, 3, 4, 5}, {5, 6})
```

Результат

```
false
```

List.ContainsAny

List.ContainsAny(list as list, values as list, optional equationCriteria as any) as logical

Описание

Указывает, содержит ли список `list` какие-либо значения из другого списка `values`. Возвращает значение true, если значение найдено в списке, и значение false в противном случае. Необязательное значение критерия уравнения `equationCriteria` можно указать для управления проверкой на равенство.

Пример №-1

Описание

Определить, содержит список {1, 2, 3, 4, 5} значение 3 или 9.

Код

```
List.ContainsAny({1, 2, 3, 4, 5}, {3, 9})
```

Результат

```
true
```

Пример №-2

Описание

Определить, содержит список {1, 2, 3, 4, 5} значение 6 или 7.

Код

```
List.ContainsAny({1, 2, 3, 4, 5}, {6, 7})
```

Результат

```
false
```

List.Count

List.Count(list as list) as number

Описание

Возвращает число элементов в списке `list`.

Пример №-1

Описание

Определить количество значений в списке {1, 2, 3}.

Код

```
List.Count({1, 2, 3})
```

Результат

```
3
```

List.Covariance

List.Covariance(numberList1 as list, numberList2 as list) as number

Описание

Возвращает ковариацию между двумя списками, `numberList1` и `numberList2`. `numberList1` и `numberList2` должны содержать одинаковое количество значений `number`.

Пример №-1

Описание

Вычислить ковариацию между двумя списками.

Код

```
List.Covariance({1, 2, 3},{1, 2, 3})
```

Результат

```
0.6666666666666607
```

List.DateTimeZones

List.DateTimeZones(start as datetimezone, count as number, step as duration) as list

Описание

Возвращает список значений `datetimezone` размера `count`, начиная с `start`. Данное значение приращения `step` является значением `duration`, которое добавляется к каждому значению.

Пример №-1

Описание

Создать список из 10 значений начиная за 5 минут до наступления Нового года (#datetimezone(2011, 12, 31, 23, 55, 0, -8, 0)) с увеличением на 1 минуту (#duration(0, 0, 1, 0)).

Код

```
List.DateTimeZones(#datetimezone(2011, 12, 31, 23, 55, 0, -8, 0), 10, #duration(0, 0, 1, 0))
```

Результат

```
{
    #datetimezone(2011, 12, 31, 23, 55, 0, -8, 0),
    #datetimezone(2011, 12, 31, 23, 56, 0, -8, 0),
    #datetimezone(2011, 12, 31, 23, 57, 0, -8, 0),
    #datetimezone(2011, 12, 31, 23, 58, 0, -8, 0),
    #datetimezone(2011, 12, 31, 23, 59, 0, -8, 0),
    #datetimezone(2012, 1, 1, 0, 0, 0, -8, 0),
    #datetimezone(2012, 1, 1, 0, 1, 0, -8, 0),
    #datetimezone(2012, 1, 1, 0, 2, 0, -8, 0),
    #datetimezone(2012, 1, 1, 0, 3, 0, -8, 0),
    #datetimezone(2012, 1, 1, 0, 4, 0, -8, 0)
}
```

List.DateTimes

List.DateTimes(start as datetime, count as number, step as duration) as list

Описание

Возвращает список значений `datetime` размера `count`, начиная с `start`. Данное значение приращения `step` является значением `duration`, которое добавляется к каждому значению.

Пример №-1

Описание

Создать список из 10 значений, начиная за 5 минут до наступления Нового года (#datetime(2011, 12, 31, 23, 55, 0)) с увеличением на 1 минуту (#duration(0, 0, 1, 0)).

Код

```
List.DateTimes(#datetime(2011, 12, 31, 23, 55, 0), 10, #duration(0, 0, 1, 0))
```

Результат

```
{
    #datetime(2011, 12, 31, 23, 55, 0),
    #datetime(2011, 12, 31, 23, 56, 0),
    #datetime(2011, 12, 31, 23, 57, 0),
    #datetime(2011, 12, 31, 23, 58, 0),
    #datetime(2011, 12, 31, 23, 59, 0),
    #datetime(2012, 1, 1, 0, 0, 0),
    #datetime(2012, 1, 1, 0, 1, 0),
    #datetime(2012, 1, 1, 0, 2, 0),
    #datetime(2012, 1, 1, 0, 3, 0),
    #datetime(2012, 1, 1, 0, 4, 0)
    }
```

List.Dates

List.Dates(start as date, count as number, step as duration) as list

Описание

Возвращает список значений `date` размера `count`, начиная с `start`. Данное значение приращения `step` является значением `duration`, которое добавляется к каждому значению.

Пример №-1

Описание

Создать список из пяти значений, начиная с новогодней ночи (#date(2011, 12, 31)), с увеличением значения на один день (#duration(1, 0, 0, 0)).

Код

```
List.Dates(#date(2011, 12, 31), 5, #duration(1, 0, 0, 0))
```

Результат

```
{
    #date(2011, 12, 31),
    #date(2012, 1, 1),
    #date(2012, 1, 2),
    #date(2012, 1, 3),
    #date(2012, 1, 4)
}
```

List.Difference

List.Difference(list1 as list, list2 as list, optional equationCriteria as any) as list

Описание

Возвращает элементы в списке `list1`, которых нет в списке `list2`. Повторяющиеся значения поддерживаются. Необязательное значение критерия уравнения `equationCriteria` можно указать для управления проверкой на равенство.

Пример №-1

Описание

Найти элементы в списке {1, 2, 3, 4, 5}, которых нет в списке {4, 5, 3}.

Код

```
List.Difference({1, 2, 3, 4, 5},{4, 5, 3})
```

Результат

```
{1, 2}
```

Пример №-2

Описание

Найти элементы в списке {1, 2}, которых нет в списке {1, 2, 3}.

Код

```
List.Difference({1, 2}, {1, 2, 3})
```

Результат

```
{}
```

List.Distinct

List.Distinct(list as list, optional equationCriteria as any) as list

Описание

Возвращает список, содержащий все значения в списке `list` с удаленными повторениями. Если этот список пуст, то результатом является пустой список.

Пример №-1

Описание

Удалить повторяющиеся элементы из списка {1, 1, 2, 3, 3, 3}.

Код

```
List.Distinct({1, 1, 2, 3, 3, 3})
```

Результат

```
{1, 2, 3}
```

List.Durations

List.Durations(start as duration, count as number, step as duration) as list

Описание

Возвращает список значений `count duration`, начиная с `start`, с добавлением заданного значения `duration step`.

Пример №-1

Описание

Создать список из пяти значений, начиная с первого часа, с добавлением одного часа.

Код

```
List.Durations(#duration(0, 1, 0, 0), 5, #duration(0, 1, 0, 0))
```

Результат

```
{#duration(0, 1, 0, 0),
    #duration(0, 2, 0, 0),
    #duration(0, 3, 0, 0),
    #duration(0, 4, 0, 0),
    #duration(0, 5, 0, 0)}
```

List.FindText

List.FindText(list as list, text as text) as list

Описание

Возвращает список значений из списка `list`, содержащих значение `text`.

Пример №-1

Описание

Поиск текстовых значений в списке {"a", "b", "ab"}, совпадающих с "a".

Код

```
List.FindText({"a", "b", "ab"}, "a")
```

Результат

```
{"a", "ab"}
```

List.First

List.First(list as list, optional defaultValue as any) as any

Описание

Возвращает первый элемент в списке `list` или необязательное значение по умолчанию `defaultValue`, если список пуст. Если список пуст, а значение по умолчанию не указано, функция возвращает значение `null`.

Пример №-1

Описание

Найти первое значение в списке {1, 2, 3}.

Код

```
List.First({1, 2, 3})
```

Результат

```
1
```

Пример №-2

Описание

Найти первое значение в списке {}. Если этот список пуст, возвращается значение -1.

Код

```
List.First({}, -1)
```

Результат

```
-1
```

List.FirstN

List.FirstN(list as list, countOrCondition as any) as any

Описание

- Если указано число, возвращается количество элементов до указанного.
- Если указано условие, возвращаются все элементы, которые изначально соответствуют условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.

Пример №-1

Описание

Найти первые значения в списке {3, 4, 5, -1, 7, 8, 2}, превышающие 0.

Код

```
List.FirstN({3, 4, 5, -1, 7, 8, 2},each _ > 0)
```

Результат

```
{3, 4, 5}
```

List.Generate

List.Generate(initial as function, condition as function, next as function, optional selector as function) as list

Описание

Создает список значений с четырьмя функциями, которые создают начальное значение `initial`, проверяют выполнение условия `condition` и в случае успеха выбирают результат и формируют следующее значение `next`. Необязательный параметр `selector` может также быть указан.

Пример №-1

Описание

Создать список записей, содержащих x и y, где x - это значение, а y - список. Значение x должно быть меньше 10 и представлять число элементов в списке y. После того как список сформирован, возвращает только значения x.

Код

```
List.Generate(()=> [ x = 1 , y = {}] , each [x] < 10 , each [x = List.Count([y]), y =
[y] & {x}] , each [x])
```

Результат

```
{1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

Пример №-2

Описание

Создать список, который начинается с 10, остается больше 0 и уменьшается на 1.

Код

```
List.Generate(()=>10, each _ > 0, each _ - 1)
```

Результат

```
{10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
```

List.InsertRange

List.InsertRange(list as list, index as number, values as list) as list

Описание

Возвращает новый список, созданный путем вставки значений из `values` в `list` с `index`. Первая позиция в списке находится по индексу 0.

- `list`: целевой список, в который будут вставлены значения.
- `index`: индекс целевого списка (`list`), в который будут вставлены значения. Первая позиция в списке находится по индексу 0.
- `values`: список значений, которые будут вставлены в `list`.

  .

Пример №-1

Описание

Вставка списка с вложенным списком ({1, {1.1, 1.2}}) в целевой список ({2, 3, 4}) по индексу 0.

Код

```
List.InsertRange({2, 3, 4}, 0, {1, {1.1, 1.2}})
```

Результат

```
{
    1, {
        1.1,
        1.2
    },
    2,
    3,
    4
```

```
}
```

Пример №-2

Описание

Вставка списка ({3, 4}) в целевой список ({1, 2, 5}) по индексу 2.

Код

```
List.InsertRange({1, 2, 5}, 2, {3, 4})
```

Результат

```
{
    1,
    2,
    3,
    4,
    5
}
```

List.Intersect

List.Intersect(lists as list, optional equationCriteria as any) as list

Описание

Возвращает пересечение значений списка, содержащихся в списке входных данных `lists`. Может быть указан необязательный параметр `equationCriteria`.

Пример №-1

Описание

Определение пересечения двух списков {1..5}, {2..6}, {3..7}.

Код

```
List.Intersect({{1..5}, {2..6}, {3..7}})
```

Результат

```
{3, 4, 5}
```

List.IsDistinct

List.IsDistinct(list as list, optional equationCriteria as any) as logical

Описание

Возвращает логическое значение, указывающее, имеются ли повторяющиеся значения в списке `list`; `true`, если значения в списке уникальны; `false`, если есть повторяющиеся значения.

Пример №-1

Описание

Определяет, уникальны ли значения списка {1, 2, 3, 3} (т. е. отсутствуют ли в нем повторяющиеся значения).

Код

```
List.IsDistinct({1, 2, 3, 3})
```

Результат

```
false
```

Пример №-2

Описание

Определяет, уникальны ли значения списка {1, 2, 3} (т. е. отсутствуют ли в нем повторяющиеся значения).

Код

```
List.IsDistinct({1, 2, 3})
```

Результат

```
true
```

List.IsEmpty

List.IsEmpty(list as list) as logical

Описание

Возвращает значение `true`, если список `list` не содержит значений (длина равна 0). Если список содержит значения (длина > 0), возвращает `false`.

Пример №-1

Описание

Выясняет, пуст ли список {}.

Код

```
List.IsEmpty({})
```

Результат

```
true
```

Пример №-2

Описание

Выясняет, пуст ли список {1, 2}.

Код

```
List.IsEmpty({1, 2})
```

Результат

```
false
```

List.Last

List.Last(list as list, optional defaultValue as any) as any

Описание

Возвращает последний элемент в списке `list` или необязательное значение по умолчанию `defaultValue`, если список пуст. Если список пуст, а значение по умолчанию не указано, функция возвращает значение `null`.

Пример №-1

Описание

Найти последнее значение в списке {} или получить -1, если он пуст.

Код

```
List.Last({}, -1)
```

Результат

```
-1
```

Пример №-2

Описание

Найти последнее значение в списке {1, 2, 3}.

Код

```
List.Last({1, 2, 3})
```

Результат

```
3
```

List.LastN

List.LastN(list as list, optional countOrCondition as any) as any

Описание

Возвращает последний элемент списка `list`. Если список пуст, возникает исключение. Эта функция принимает необязательный параметр `countOrCondition` для поддержки сбора нескольких элементов или фильтрации элементов. `countOrCondition` можно указать тремя способами:

- Если указано число, возвращается количество элементов до указанного.
- Если указано условие, возвращаются все элементы, которые изначально соответствуют условию, начиная с конца списка. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.
- Если этот параметр имеет значение NULL, то возвращается последний элемент в списке.

Пример №-1

Описание

Найти последнее значение в списке {3, 4, 5, -1, 7, 8, 2}.

Код

```
List.LastN({3, 4, 5, -1, 7, 8, 2},1)
```

Результат

```
{2}
```

Пример №-2

Описание

Найти последние значения в списке {3, 4, 5, -1, 7, 8, 2}, превышающие 0.

Код

```
List.LastN({3, 4, 5, -1, 7, 8, 2}, each _ > 0)
```

Результат

```
{7, 8, 2}
```

List.MatchesAll

List.MatchesAll(list as list, condition as function) as logical

Описание

Возвращает `true`, если функции условия `condition` удовлетворяют все значения в списке `list`, в противном случае возвращает значение `false`.

Пример №-1

Описание

Определить, все ли значения в списке {11, 12, 13} больше 10.

Код

```
List.MatchesAll({11, 12, 13},each _  > 10)
```

Результат

```
true
```

Пример №-2

Описание

Определить, все ли значения в списке {1, 2, 3} больше 10.

Код

```
List.MatchesAll({1, 2, 3},each _  > 10)
```

Результат

```
false
```

List.MatchesAny

List.MatchesAny(list as list, condition as function) as logical

Описание

Возвращает `true`, если функции условия `condition` соответствует любое значение в списке `list`, в противном случае возвращает значение `false`.

Пример №-1

Описание

Определить, есть ли в списке {1, 2, 3} хотя бы одно значение больше 10.

Код

```
List.MatchesAny({1, 2, 3},each _  > 10)
```

Результат

```
false
```

Пример №-2

Описание

Определить, есть ли в списке {9, 10, 11} хотя бы одно значение больше 10.

Код

```
List.MatchesAny({9, 10, 11},each _  > 10)
```

Результат

```
true
```

List.Max

List.Max(list as list, optional default as any, optional comparisonCriteria as any, optional includeNulls as logical) as any

Описание

Возвращает максимальный элемент в списке `list` или необязательное значение по умолчанию `default`, если список пуст. Может быть указано необязательное значение comparisonCriteria, `comparisonCriteria`, определяющее, как сравнивать элементы списка. Если этот параметр имеет значение NULL, используется функция сравнения по умолчанию.

Пример №-1

Описание

Найти максимальное значение в списке {1, 4, 7, 3, -2, 5}.

Код

```
List.Max({1, 4, 7, 3, -2, 5},1)
```

Результат

```
7
```

Пример №-2

Описание

Найти максимум в списке {} или получить -1, если список пуст.

Код

```
List.Max({}, -1)
```

Результат

```
-1
```

List.MaxN

List.MaxN(list as list, countOrCondition as any, optional comparisonCriteria as any, optional includeNulls as logical) as list

Описание

Возвращает максимальные значения в списке `list`. После того как строки отсортированы, результаты могут быть отфильтрованы с помощью необязательных параметров, если они заданы. Необязательный параметр `countOrCondition` указывает количество возвращаемых значений или условие фильтрования. Необязательный параметр `comparisonCriteria` указывает, как сравнивать значения в списке.

- `list`: список значений.
- `countOrCondition`: если указано число, возвращается список элементов до `countOrCondition` по возрастанию. Если указано условие, возвращается список элементов, которые изначально соответствуют условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.
- `comparisonCriteria`: *[Необязательно]* Необязательное значение `comparisonCriteria` определяет, как сравнивать элементы в списке. Если этот параметр имеет значение NULL, используется функция сравнения по умолчанию.

List.Median

List.Median(list as list, optional comparisonCriteria as any) as any

Описание

Возвращает медиану списка `list`. Эта функция вызывает исключение, если список пуст. Если в списке четное число элементов, функция выберет меньший из двух элементов-медиан.

Пример №-1

Описание

Найти медиану списка `{5, 3, 1, 7, 9}`.

Код

```
List.Median({5, 3, 1, 7, 9})
```

Результат

```
5
```

List.Min

List.Min(list as list, optional default as any, optional comparisonCriteria as any, optional includeNulls as logical) as any

Описание

Возвращает минимальный элемент в списке `list` или необязательное значение по умолчанию `default`, если список пуст. Может быть указано необязательное значение comparisonCriteria, `comparisonCriteria`, определяющее, как сравнивать элементы списка. Если этот параметр имеет значение NULL, используется функция сравнения по умолчанию.

Пример №-1

Описание

Найти минимум в списке {} или получить -1, если список пуст.

Код

```
List.Min({}, -1)
```

Результат

```
-1
```

Пример №-2

Описание

Найти минимальное значение в списке {1, 4, 7, 3, -2, 5}.

Код

```
List.Min({1, 4, 7, 3, -2, 5})
```

Результат

```
-2
```

List.MinN

List.MinN(list as list, countOrCondition as any, optional comparisonCriteria as any, optional includeNulls as logical) as list

Описание

Возвращает минимальные значения в списке `list`. Параметр `countOrCondition` задает количество возвращаемых значений или условие фильтрования. Необязательный параметр `comparisonCriteria` указывает, как сравнивать значения в списке.

- `list`: список значений.
- `countOrCondition`: если указано число, возвращается список элементов до `countOrCondition` по возрастанию. Если указано условие, возвращается список элементов, которые изначально соответствуют условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются. Если этот параметр имеет значение NULL, то возвращается одно наименьшее значение в списке.
- `comparisonCriteria`: *[Необязательно]* Необязательное значение `comparisonCriteria` определяет, как сравнивать элементы в списке. Если этот параметр имеет значение NULL, используется функция сравнения по умолчанию.

Пример №-1

Описание

Найти 5 наименьших значений в списке `{3, 4, 5, -1, 7, 8, 2}`.

Код

```
List.MinN({3, 4, 5, -1, 7, 8, 2}, 5)
```

Результат

```
{-1, 2, 3, 4, 5}
```

List.Mode

List.Mode(list as list, optional equationCriteria as any) as any

Описание

Возвращает элемент, который появляется чаще всего в списке `list`. Если список пуст, возникает исключение. Если несколько элементов присутствуют с одинаковой максимальной частотой, выбирается последний из них. Можно указать необязательное значение `comparisonCriteria`, `equationCriteria`, для управления проверкой на равенство.

Пример №-1

Описание

Найти элемент, который наиболее часто появляется в списке `{"A", 1, 2, 3, 3, 4, 5}`.

Код

```
List.Mode({"A", 1, 2, 3, 3, 4, 5})
```

Результат

```
3
```

Пример №-2

Описание

Найти элемент, который наиболее часто появляется в списке `{"A", 1, 2, 3, 3, 4, 5, 5}`.

Код

```
List.Mode({"A", 1, 2, 3, 3, 4, 5, 5})
```

Результат

```
5
```

List.Modes

List.Modes(list as list, optional equationCriteria as any) as list

Описание

Возвращает элемент, который появляется чаще всего в списке `list`. Если список пуст, возникает исключение. Если несколько элементов присутствуют с одинаковой максимальной частотой, выбирается последний из них. Можно указать необязательное значение `comparisonCriteria`, `equationCriteria`, для управления проверкой на равенство.

Пример №-1

Описание

Найти элементы, которые появляются чаще всего в списке `{"A", 1, 2, 3, 3, 4, 5, 5}`.

Код

```
List.Modes({"A", 1, 2, 3, 3, 4, 5, 5})
```

Результат

```
{3, 5}
```

List.NonNullCount

List.NonNullCount(list as list) as number

Описание

Возвращает число элементов, отличных от NULL, в списке `list`.

List.Numbers

List.Numbers(start as number, count as number, optional increment as number) as list

Описание

Возвращает список чисел для заданного исходного значения, количества и необязательного значения приращения. Значение приращения по умолчанию - 1.

- `start`: исходное значение в списке.
- `count`: количество значений, которое требуется создать.
- `increment`: *[Необязательно]* Значение, на которое выполняется увеличение. Если не указано иное, значение увеличивается с шагом 1.

Пример №-1

Описание

Создать список из 10 последовательных чисел, начиная с 1.

Код

```
List.Numbers(1, 10)
```

Результат

```
{
    1,
    2,
    3,
    4,
    5,
    6,
    7,
    8,
    9,
    10
}
```

Пример №-2

Описание

Создать список чисел из 10 чисел, начиная с 1, с шагом 2 для каждого последующего числа.

Код

```
List.Numbers(1, 10, 2)
```

Результат

```
{
    1,
    3,
    5,
    7,
```

```
    9,
    11,
    13,
    15,
    17,
    19
}
```

List.PositionOf

List.PositionOf(list as list, value as any, optional occurrence as number, optional equationCriteria as any) as any

Описание

Возвращает смещение, на котором значение `value` появляется в списке `list`. Возвращает -1, если значение не обнаруживается. Можно указать необязательный параметр вхождения `occurrence`.

- `occurrence`: Максимальное количество возвращаемых вхождений.

Пример №-1

Описание

Найти позицию в списке {1, 2, 3}, в которой появляется значение 3.

Код

```
List.PositionOf({1, 2, 3}, 3)
```

Результат

```
2
```

List.PositionOfAny

List.PositionOfAny(list as list, values as list, optional occurrence as number, optional equationCriteria as any) as any

Описание

Возвращает смещение в списке `list` первого вхождения значения в список `values`. Возвращает значение -1, если вхождение не найдено. Можно указать необязательный параметр вхождения `occurrence`.

- `occurrence`: Максимальное количество вхождений, которые могут быть возвращены.

Пример №-1

Описание

Найти первую позицию в списке {1, 2, 3}, в которой появляется значение 2 или 3.

Код

```
List.PositionOfAny({1, 2, 3}, {2, 3})
```

Результат

```
1
```

List.Positions

List.Positions(list as list) as list

Описание

Возвращает список смещений для списка ввода `list`. Если для изменения списка используется List.Transform, список позиций может служить для предоставления преобразованию доступа к позиции.

Пример №-1

Описание

Найти смещения значений в списке {1, 2, 3, 4, null, 5}.

Код

```
List.Positions({1, 2, 3, 4, null, 5})
```

Результат

```
{0, 1, 2, 3, 4, 5}
```

List.Product

List.Product(numbersList as list, optional precision as number) as number

Описание

Возвращает произведение чисел в списке `numbersList`, отличных от NULL. Возвращает значение NULL, если в списке нет значений, отличных от NULL.

Пример №-1

Описание

Найти произведение чисел в списке `{1, 2, 3, 3, 4, 5, 5}`.

Код

```
List.Product({1, 2, 3, 3, 4, 5, 5})
```

Результат

```
1800
```

List.Random

List.Random(count as number, optional seed as number) as list

Описание

Возвращает список случайных чисел от 0 до 1 для заданного количества создаваемых значений и необязательного начального значения.

- `count`: число создаваемых случайных значений.
- `seed`: *[необязательно]* числовое значение, используемое для инициализации генератора псевдослучайных чисел. Если не указано, при каждом вызове функции создается уникальный список случайных чисел. Если указано численное начальное значение, каждый вызов функции будет создавать один и тот же список случайных чисел для этого значения.

Пример №-1

Описание

Создать список из 3 случайных чисел.

Код

```
List.Random(3)
```

Результат

```
{0.992332, 0.132334, 0.023592}
```

Пример №-2

Описание

Создать список из 3 случайных чисел с указанным начальным значением.

Код

```
List.Random(3, 2)
```

Результат

```
{0.883002, 0.245344, 0.723212}
```

List.Range

List.Range(list as list, offset as number, optional count as number) as list

Описание

Возвращает подмножество списка, начиная со смещения `list`. Необязательный параметр `offset` устанавливает максимальное число элементов в подмножестве.

Пример №-1

Описание

Найти подмножество с количеством элементов 2, начиная со смещения 6, в списке чисел от 1 до 10.

Код

```
List.Range({1..10}, 6, 2)
```

Результат

```
{7, 8}
```

Пример №-2

Описание

Найти подмножество, начиная со смещения 6, в списке чисел от 1 до 10.

Код

```
List.Range({1..10}, 6)
```

Результат

```
{7, 8, 9, 10}
```

List.RemoveFirstN

List.RemoveFirstN(list as list, optional countOrCondition as any) as list

Описание

Возвращает список, который удаляет первый элемент списка `list`. Если список `list` пуст, то возвращается пустой список. Эта функция принимает необязательный параметр `countOrCondition` для поддержки удаления нескольких значений, как указано ниже.

- Если указано число, удалено будет количество элементов до указанного значения.
- Если указано условие, возвращаемый список начинается с первого элемента в `list`, удовлетворяющего условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.
- Если параметр имеет значение NULL, наблюдается поведение по умолчанию.

Пример №-1

Описание

Создать список из {1, 2, 3, 4, 5} без первых 3 цифр.

Код

```
List.RemoveFirstN({1, 2, 3, 4, 5}, 3)
```

Результат

```
{4, 5}
```

Пример №-2

Описание

Создать список из {5, 4, 2, 6, 1}, который начинается с числа меньше 3.

Код

```
List.RemoveFirstN({5, 4, 2, 6, 1}, each _ > 3)
```

Результат

```
{2, 6, 1}
```

List.RemoveItems

List.RemoveItems(list1 as list, list2 as list) as list

Описание

Удаляет все вхождения указанных значений из `list2` в `list1`. Если значения в `list2` не существуют в `list1`, то возвращается исходный список.

Пример №-1

Описание

Удалить элементы списка {2, 4, 6} из списка {1, 2, 3, 4, 2, 5, 5}.

Код

```
List.RemoveItems({1, 2, 3, 4, 2, 5, 5}, {2, 4, 6})
```

Результат

```
{1, 3, 5, 5}
```

List.RemoveLastN

List.RemoveLastN(list as list, optional countOrCondition as any) as list

Описание

Возвращает список, который удаляет последние `countOrCondition` элементов в конце списка `list`. Если в `list` меньше `countOrCondition` элементов, будет возвращен пустой список.

- Если указано число, удалено будет количество элементов до указанного значения.
- Если указано условие, возвращаемый список заканчивается первым элементом из конца в `list`, удовлетворяющего условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.
- Если этот параметр имеет значение NULL, будет удален только один элемент.

Пример №-1

Описание

Создать список из {5, 4, 2, 6, 4}, который оканчивается на число меньше 3.

Код

```
List.RemoveLastN({5, 4, 2, 6, 4}, each _ > 3)
```

Результат

```
{5, 4, 2}
```

Пример №-2

Описание

Создать список из {1, 2, 3, 4, 5} без последних 3 цифр.

Код

```
List.RemoveLastN({1, 2, 3, 4, 5}, 3)
```

Результат

```
{1, 2}
```

List.RemoveMatchingItems

List.RemoveMatchingItems(list1 as list, list2 as list, optional equationCriteria as any) as list

Описание

Удаляет все вхождения значений, заданных в списке `list2`, из списка `list1`. Если значения в `list2` не существуют в `list1`, то возвращается исходный список. Необязательное значение критерия уравнения `equationCriteria` можно указать для управления проверкой на равенство.

Пример №-1

Описание

Создать список из {1, 2, 3, 4, 5, 5} без {1, 5}.

Код

```
List.RemoveMatchingItems({1, 2, 3, 4, 5, 5}, {1, 5})
```

Результат

```
{2, 3, 4}
```

List.RemoveNulls

List.RemoveNulls(list as list) as list

Описание

Удаляет все вхождения значений NULL из списка `list`. Если в списке нет значений NULL, возвращается исходный список.

Пример №-1

Описание

Удалить значения NULL из списка {1, 2, 3, null, 4, 5, null, 6}.

Код

```
List.RemoveNulls({1, 2, 3, null, 4, 5, null, 6})
```

Результат

```
{1, 2, 3, 4, 5, 6}
```

List.RemoveRange

List.RemoveRange(list as list, index as number, optional count as number) as list

Описание

Удаляет `count` значений из списка `list`, начиная с указанной позиции `index`.

Пример №-1

Описание

Удалить 3 значения из списка {1, 2, 3, 4, -6, -2, -1, 5}, начиная с индекса 4.

Код

```
List.RemoveRange({1, 2, 3, 4, -6, -2, -1, 5}, 4, 3)
```

Результат

```
{1, 2, 3, 4, 5}
```

List.Repeat

List.Repeat(list as list, count as number) as list

Описание

Возвращает список, `count` раз повторяющий исходный список `list`.

Пример №-1

Описание

Создать список, в котором {1, 2} повторяется 3 раза.

Код

```
List.Repeat({1, 2}, 3)
```

Результат

```
{1, 2, 1, 2, 1, 2}
```

List.ReplaceMatchingItems

List.ReplaceMatchingItems(list as list, replacements as list, optional equationCriteria as any) as list

Описание

Выполняет заданные замены в списке `list`. В операции замены `replacements` применяется список из пар значений, старого и нового. Необязательное значение критерия уравнения `equationCriteria` можно указать для управления проверкой на равенство.

Пример №-1

Описание

Создать список из {1, 2, 3, 4, 5}, заменив значение 5 на -5, а значение 1 на -1.

Код

```
List.ReplaceMatchingItems({1, 2, 3, 4, 5}, {{5, -5}, {1, -1}})
```

Результат

```
{-1, 2, 3, 4, -5}
```

List.ReplaceRange

List.ReplaceRange(list as list, index as number, count as number, replaceWith as list) as list

Описание

Заменяет `count` значений в списке `list` списком `replaceWith`, начиная с указанной позиции `index`.

Пример №-1

Описание

Заменить {7, 8, 9} в списке {1, 2, 7, 8, 9, 5} списком {3, 4}.

Код

```
List.ReplaceRange({1, 2, 7, 8, 9, 5}, 2, 3, {3, 4})
```

Результат

```
{1, 2, 3, 4, 5}
```

List.ReplaceValue

List.ReplaceValue(list as list, oldValue as any, newValue as any, replacer as function) as list

Описание

Ищет в списке значений `list` значение `oldValue` и заменяет каждое его вхождение значением `newValue`.

Пример №-1

Описание

Заменить все значения "a" в списке {"a", "B", "a", "a"} значением "A".

Код

```
List.ReplaceValue({"a", "B", "a", "a"}, "a", "A", Replacer.ReplaceText)
```

Результат

```
{"A", "B", "A", "A"}
```

List.Reverse

List.Reverse(list as list) as list

Описание

Возвращает список со значениями в списке `list` в обратном порядке.

Пример №-1

Описание

Создать список из {1..10} в обратном порядке.

Код

```
List.Reverse({1..10})
```

Результат

```
{10, 9, 8, 7, 6, 5, 4, 3, 2, 1}
```

List.Select

List.Select(list as list, selection as function) as list

Описание

Возвращает список значений из списка `list`, которые удовлетворяют условию выбора `selection`.

Пример №-1

Описание

Найти в списке {1, -3, 4, 9, -2} значения больше 0.

Код

```
List.Select({1, -3, 4, 9, -2}, each _ > 0)
```

Результат

```
{1, 4, 9}
```

List.Single

List.Single(list as list) as any

Описание

Если в списке `list` только один элемент, возвращает этот элемент. Если в списке больше одного элемента или он пуст, функция вызывает исключение.

Пример №-1

Описание

Найти единственное значение в списке {1, 2, 3}.

Код

```
List.Single({1, 2, 3})
```

Результат

```
[Expression.Error] There were too many elements in the enumeration to complete the operation.
```

Пример №-2

Описание

Найти единственное значение в списке {1}.

Код

```
List.Single({1})
```

Результат

```
1
```

List.SingleOrDefault

List.SingleOrDefault(list as list, optional default as any) as any

Описание

Если в списке `list` только один элемент, возвращает этот элемент. Если этот список пуст, то функция возвращает значение NULL, если не указан необязательный параметр `default`. Если в списке более одного элемента, функция возвращает ошибку.

Пример №-1

Описание

Найти единственное значение в списке {}. Если список пуст, возвратить -1.

Код

```
List.SingleOrDefault({}, -1)
```

Результат

```
-1
```

Пример №-2

Описание

Найти единственное значение в списке {}.

Код

```
List.SingleOrDefault({})
```

Результат

```
null
```

Пример №-3

Описание

Найти единственное значение в списке {1}.

Код

```
List.SingleOrDefault({1})
```

Результат

```
1
```

List.Skip

List.Skip(list as list, optional countOrCondition as any) as list

Описание

Возвращает список, который пропускает первый элемент списка `list`. Если список `list` пуст, то возвращается пустой список. Эта функция принимает необязательный параметр `countOrCondition` для поддержки пропуска нескольких значений, как указано ниже.

- Если указано число, пропускается количество элементов до указанного.
- Если указано условие, возвращаемый список начинается с первого элемента в `list`, удовлетворяющего условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.
- Если параметр имеет значение NULL, наблюдается поведение по умолчанию.

Пример №-1

Описание

Создать список из {1, 2, 3, 4, 5} без первых 3 цифр.

Код

```
List.Skip({1, 2, 3, 4, 5}, 3)
```

Результат

```
{4, 5}
```

Пример №-2

Описание

Создать список из {5, 4, 2, 6, 1}, который начинается с числа меньше 3.

Код

```
List.Skip({5, 4, 2, 6, 1}, each _ > 3)
```

Результат

```
{2, 6, 1}
```

List.Sort

List.Sort(list as list, optional comparisonCriteria as any) as list

Описание

Сортирует список данных `list` по необязательным указанным критериям. Необязательный параметр `comparisonCriteria` можно указать как критерий сравнения. Он может принимать следующие значения:

- Для управления порядком критерий сравнения может быть значением перечислителя Order. (`Order.Descending`, `Order.Ascending`).
- Для вычисления ключа, который будет использоваться для сортировки, можно использовать функцию с 1 аргументом.
- Чтобы выбрать и ключ, и порядок сортировки, критерий сравнения может быть задан списком, содержащим ключ и порядок (`{each 1 / _, Order.Descending}`).
- Чтобы полностью управлять сравнением, можно использовать функцию с 2 аргументами, которая возвращает -1, 0 или 1 в зависимости от связи между левым и правым аргументами. Value.Compare - метод, с помощью которого можно делегировать эту логику.

Пример №-1

Описание

Сортировать список {2, 3, 1}.

Код

```
List.Sort({2, 3, 1})
```

Результат

```
{1, 2, 3}
```

Пример №-2

Описание

Сортировать список {2, 3, 1} в порядке убывания с помощью метода Value.Compare.

Код

```
List.Sort({2, 3, 1}, (x, y) => Value.Compare(1/x, 1/y))
```

Результат

```
{3, 2, 1}
```

Пример №-3

Описание

Сортировать список {2, 3, 1} в порядке убывания.

Код

```
List.Sort({2, 3, 1}, Order.Descending)
```

Результат

```
{3, 2, 1}
```

List.StandardDeviation

List.StandardDeviation(numbersList as list) as number

Описание

Возвращает основанную на выборке оценку стандартного отклонения значений в списке `numbersList`. Если `numbersList` — список чисел, возвращается число. Исключение возникает при пустом списке или списке элементов, не имеющих тип `number`.

Пример №-1

Описание

Найти стандартное отклонение чисел от 1 до 5.

Код

```
List.StandardDeviation({1..5})
```

Результат

```
1.5811388300841898
```

List.Sum

List.Sum(list as list, optional precision as number) as any

Описание

Возвращает сумму всех значений в списке `list` , отличных от NULL. Возвращает значение NULL, если в списке нет значений, отличных от NULL.

Пример №-1

Описание

Найти сумму чисел в списке `{1, 2, 3}`.

Код

```
List.Sum({1, 2, 3})
```

Результат

```
6
```

List.Times

List.Times(start as time, count as number, step as duration) as list

Описание

Возвращает список значений `time` с размером `count`, начиная с `start`. Данное значение приращения `step` является значением `duration`, которое добавляется к каждому значению.

Пример №-1

Описание

Создать список из 4 значений, начиная с полудня (#time(12, 0, 0)), последовательно увеличивая время на один час (#duration(0, 1, 0, 0)).

Код

```
List.Times(#time(12, 0, 0), 4, #duration(0, 1, 0, 0))
```

Результат

```
{
    #time(12, 0, 0),
    #time(13, 0, 0),
    #time(14, 0, 0),
    #time(15, 0, 0)
}
```

List.Transform

List.Transform(list as list, transform as function) as list

Описание

Возвращает новый список значений, применяя функцию преобразования `transform` к списку `list`.

Пример №-1

Описание

Добавить 1 к каждому значению в списке {1, 2}.

Код

```
List.Transform({1, 2}, each _ + 1)
```

Результат

```
{2, 3}
```

List.TransformMany

List.TransformMany(list as list, collectionTransform as function, resultTransform as function) as list

Описание

Возвращает список элементов, отображенных из входного списка. Функция collectionTransform применяется к каждому элементу, а функция resultTransform вызывается для построения результирующего списка. collectionSelector имеет подпись (x as Any) => … Здесь x - элемент в списке. resultTransform отображает форму результата и имеет подпись (x as Any, y as Any) =>… Здесь x - элемент в списке, а y - элемент, полученный путем применения collectionTransform к данному элементу.

List.Union

List.Union(lists as list, optional equationCriteria as any) as list

Описание

Принимает список списков `lists`, объединяет элементы отдельных списков и возвращает их в выходном списке. Результирующий возвращаемый список содержит все элементы всех входных списков. Эта операция поддерживает традиционную семантику мультимножества, поэтому

повторяющиеся значения сопоставляются как часть объединения. Необязательное значение критерия уравнения `equationCriteria` можно указать для управления проверкой на равенство.

Пример №-1

Описание

Создать объединение списков {1..5}, {2..6}, {3..7}.

Код

```
List.Union({ {1..5}, {2..6}, {3..7} })
```

Результат

```
{1, 2, 3, 4, 5, 6, 7}
```

List.Zip

List.Zip(lists as list) as list

Описание

Принимает список списков, `lists` и возвращает список списков, объединяя элементы на одной позиции.

Пример №-1

Описание

Пакует два простых списка разной длины {1, 2} и {3}.

Код

```
List.Zip({{1, 2}, {3}})
```

Результат

```
{
    { 1, 3 },
    { 2, null }
}
```

Пример №-2

Описание

Пакует два простых списка {1, 2} и {3, 4}.

Код

```
List.Zip({{1, 2}, {3, 4}})
```

Результат

```
{
    { 1, 3 },
    { 2, 4 }
}
```

Logical

Logical.From

Logical.From(value as any) as logical

Описание

Возвращает значение `logical` из указанного `value`. Если данное `value` имеет значение `null`, `Logical.From` возвращает `null`. Если данное `value` - `logical`, возвращается `value`. Значения следующих типов можно преобразовать в значение `logical`:

- `text`: значение `logical` из текстового значения, `"true"` или `"false"`. Дополнительные сведения см. в разделе `Logical.FromText`.
- `number`: `false`, если `value` равно `0`, `true` - в противном случае.

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Преобразовать `2` в значение `logical`.

Код

```
Logical.From(2)
```

Результат

```
true
```

Logical.FromText

Logical.FromText(text as text) as logical

Описание

Создает логическое значение из текстового значения `text`, которое может быть равно true или false. Если `text` содержит другую строку, возникает исключение. В текстовом значении `text` не учитывается регистр.

Пример №-1

Описание

Создание логического значения из текстовой строки true.

Код

```
Logical.FromText("true")
```

Результат

```
true
```

Пример №-2

Описание

Создание логического значения из текстовой строки "a".

Код

```
Logical.FromText("a")
```

Результат

```
[Expression.Error] Could not convert to a logical.
```

Logical.ToText

Logical.ToText(logicalValue as logical) as text

Описание

Создает текстовое значение из логического значения `logicalValue`, `true` или `false`. Если `logicalValue` не является логическим значением, возникает исключение.

Пример №-1

Описание

Создать текстовое значение из логического `true`.

Код

```
Logical.ToText(true)
```

Результат

```
"true"
```

Marketplace

Marketplace.Subscriptions

Marketplace.Subscriptions() as table

Описание

Возвращает все каналы, на которые имеется подписка из Azure Marketplace.

MySQL

MySQL.Database

MySQL.Database(server as text, database as text, optional options as record) as table

Описание

Возвращает список таблиц, представлений и хранимых скалярных функций SQL, доступных в базе данных MySQL на сервере `server` в экземпляре базы данных `database`. Дополнительно к имени сервера через двоеточие может быть указан порт. Необязательный параметр записи `options` может быть указан для управления следующими параметрами:

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `TreatTinyAsBoolean` : Логическое значение (True или False), которое указывает, задавать ли для стобцов tinyint на сервере логические значения. Значение по умолчанию — True.
- `OldGuids` : Логическое значение (True или False), которое указывает, будут ли столбцы char(36) (если False) или столбцы binary(16) (если True), учитываться как GUID. Значение по умолчанию — False.
- `ReturnSingleDatabase` : Логическое значение (True или False), которое указывает, следует ли возвращать все таблицы в базах данных (False) или возвращать таблицы и представления указанной базы данных (True). Значение по умолчанию — False.
- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

Number

Number.Abs

Number.Abs(number as number) as number

Описание

Возвращает абсолютное значение "`number`". Если "`number`" имеет значение NULL, то `Number.Abs` возвращает NULL.

- `number`: число (`number`), для которого вычисляется абсолютное значение.

Пример №-1

Описание

Абсолютное значение -3.

Код

```
Number.Abs(-3)
```

Результат

```
3
```

Number.Acos

Number.Acos(number as number) as number

Описание

Возвращает арккосинус `number`.

Number.Asin

Number.Asin(number as number) as number

Описание

Возвращает арксинус `number`.

Number.Atan

Number.Atan(number as number) as number

Описание

Возвращает арктангенс `number`.

Number.Atan2

Number.Atan2(y as number, x as number) as number

Описание

Возвращает арктангенс частного от деления двух чисел, `y` и `x`. Деление будет построено как `y/x`.

Number.BitwiseAnd

Number.BitwiseAnd(number1 as number, number2 as number) as number

Описание

Возвращает результат выполнения побитовой операции AND со значениями `number1` и `number2`.

Number.BitwiseNot

Number.BitwiseNot(number as any) as any

Описание

Возвращает результат выполнения побитовой операции NOT над `number`.

Number.BitwiseOr

Number.BitwiseOr(number1 as number, number2 as number) as number

Описание

Возвращает результат выполнения побитовой операции OR со значениями `number1` и `number2`.

Number.BitwiseShiftLeft

Number.BitwiseShiftLeft(number1 as number, number2 as number) as number

Описание

Возвращает результат выполнения побитового сдвига влево над `number1` на указанное число битов `number2`.

Number.BitwiseShiftRight

Number.BitwiseShiftRight(number1 as number, number2 as number) as number

Описание

Возвращает результат выполнения побитового сдвига вправо над `number1` на указанное число битов `number2`.

Number.BitwiseXor

Number.BitwiseXor(number1 as number, number2 as number) as number

Описание

Возвращает результат выполнения побитовой операции XOR (исключающее ИЛИ) со значениями `number1` и `number2`.

Number.Combinations

Number.Combinations(setSize as number, combinationSize as number) as number

Описание

Возвращает определенное количество уникальных сочетаний из элементов списка `setSize` с указанным размером сочетания `combinationSize`.

- `setSize`: Количество элементов в списке.
- `combinationSize`: Количество элементов в каждом сочетании.

Пример №-1

Описание

Найти количество сочетаний из 5 элементов по 3.

Код

```
Number.Combinations(5, 3)
```

Результат

```
10
```

Number.Cos

Number.Cos(number as number) as number

Описание

Возвращает косинус `number`.

Пример №-1

Описание

Найти косинус угла 0.

Код

```
Number.Cos(0)
```

Результат

```
1
```

Number.Cosh

Number.Cosh(number as number) as number

Описание

Возвращает гиперболический косинус `number`.

Number.Exp

Number.Exp(number as number) as number

Описание

Возвращает результат возведения e в степень "`number`" (экспонента).

- `number`: число (`number`), для которого вычисляется экспонента. Если "`number`" имеет значение NULL, то `Number.Exp` возвращает NULL.

Пример №-1

Описание

Возвести e в степень 3.

Код

```
Number.Exp(3)
```

Результат

```
20.085536923187668
```

Number.Factorial

Number.Factorial(number as number) as number

Описание

Возвращает факториал числа `number`.

Пример №-1

Описание

Найти факториал 10.

Код

```
Number.Factorial(10)
```

Результат

```
3628800
```

Number.From

Number.From(value as any, optional culture as text) as number

Описание

Возвращает значение `number` из указанного `value`. Если данное `value` имеет значение `null`, `Number.From` возвращает `null`. Если данное `value` - `number`, возвращается `value`. Значения следующих типов можно преобразовать в значение `number`:

- `text`: значение `number` из текстового представления. Обрабатываются стандартные текстовые форматы ("15", "3,423.10", "5.0E-10"). Дополнительные сведения см. в разделе `Number.FromText` .
- `logical`: 1 для `true`, 0 для `false`.
- `datetime`: число с плавающей запятой двойной точности, содержащее эквивалент даты OLE-автоматизации.
- `datetimezone`: число с плавающей запятой двойной точности, содержащее эквивалент даты OLE-автоматизации для локальных даты и времени `value`.
- `date`: число с плавающей запятой двойной точности, содержащее эквивалент даты OLE-автоматизации.
- `time`: выраженный в нецелых сутках.
- `duration`: выраженный в целых и нецелых сутках.

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Получает значение `number` для `"12.3 %"`.

Код

```
Number.From("12.3%")
```

Результат

```
0.123
```

Пример №-2

Описание

Получить значение `number` для `"4"`.

Код

```
Number.From("4")
```

Результат

```
4
```

Пример №-3

Описание

Получить значение `number` *для* `#datetime(2020, 3, 20, 6, 0, 0)`.

Код

```
Number.From(#datetime(2020, 3, 20, 6, 0, 0))
```

Результат

```
43910.25
```

Number.FromText

Number.FromText(text as text, optional culture as text) as number

Описание

Возвращает значение `number` из заданного текстового значения `text`.

- `text`: Текстовое представление числовых значений. Представление должно быть в общем формате числа - "15", "3,423.10", "5.0E-10".

Пример №-1

Описание

Возвратить числовое значение `"4"`.

Код

```
Number.FromText("4")
```

Результат

```
4
```

Пример №-2

Описание

Возвратить числовое значение `"5.0e-10"`.

Код

```
Number.FromText("5.0e-10")
```

Результат

```
5E-10
```

Number.IntegerDivide

Number.IntegerDivide(number1 as number, number2 as number, optional precision as number) as number

Описание

Возвращает целочисленную часть результата от деления числа "`number1`" на число "`number2`". Если "`number1`" или "`number2`" равны NULL, `Number.IntegerDivide` возвращает NULL.

- `number1`: делимое.
- `number2`: делитель.

Пример №-1

Описание

Разделить 8,3 на 3.

Код

```
Number.IntegerDivide(8.3, 3)
```

Результат

```
2
```

Пример №-2

Описание

Разделить 6 на 4.

Код

```
Number.IntegerDivide(6, 4)
```

Результат

```
1
```

Number.IsEven

Number.IsEven(number as number) as logical

Описание

Указывает, четно ли значение `number`, возвращая `true`, если оно четно, и `false` - в противном случае.

Пример №-1

Описание

Проверить, четно ли число 82.

Код

```
Number.IsEven(82)
```

Результат

```
true
```

Пример №-2

Описание

Проверить, четно ли число 625.

Код

```
Number.IsEven(625)
```

Результат

```
false
```

Number.IsNaN

Number.IsNaN(number as number) as logical

Описание

Указывает, является ли значение NaN (нечисловым). Возвращает `true`, если значение `number` эквивалентно `Number.IsNaN`, и `false` - в противном случае.

Пример №-1

Описание

Проверьте, будет ли получено NaN при делении 1 на 0.

Код

```
Number.IsNaN(1/0)
```

Результат

```
false
```

Пример №-2

Описание

Проверьте, будет ли получено NaN при делении 0 на 0.

Код

```
Number.IsNaN(0/0)
```

Результат

```
true
```

Number.IsOdd

Number.IsOdd(number as number) as logical

Описание

Указывает, нечетно ли число. Возвращает `true`, если `number` - нечетное число, `false` - в противном случае.

Пример №-1

Описание

Проверить, нечетно ли число 82.

Код

```
Number.IsOdd(82)
```

Результат

```
false
```

Пример №-2

Описание

Проверить, нечетно ли число 625.

Код

```
Number.IsOdd(625)
```

Результат

```
true
```

Number.Ln

Number.Ln(number as number) as number

Описание

Возвращает натуральный логарифм числа `number`. Если `number` имеет значение NULL, то `Number.Ln` возвращает NULL.

Пример №-1

Описание

Получить натуральный логарифм числа 15.

Код

```
Number.Ln(15)
```

Результат

```
2.70805020110221
```

Number.Log

Number.Log(number as number, optional base as number) as number

Описание

Возвращает логарифм числа `number` по основанию `base`. Если `base` не указано, применяется значение по умолчанию — "Number.E". Если `number` имеет значение NULL, `Number.Log` возвращает NULL.

Пример №-1

Описание

Получение десятичного логарифма для 2.

Код

```
Number.Log(2, 10)
```

Результат

```
0.3010299956639812
```

Пример №-2

Описание

Получение натурального логарифма для 2.

Код

```
Number.Log(2)
```

Результат

```
0.69314718055994529
```

Number.Log10

Number.Log10(number as number) as number

Описание

Возвращает десятичный логарифм числа `number`. Если `number` имеет значение NULL, `Number.Log10` возвращает NULL.

Пример №-1

Описание

Получение десятичного логарифма для 2.

Код

```
Number.Log10(2)
```

Результат

```
0.3010299956639812
```

Number.Mod

Number.Mod(number as number, divisor as number, optional precision as number) as number

Описание

Возвращает остаток от целочисленного деления "`number`" на "`divisor`". Если "`number`" или "`divisor`" равны NULL, `Number.Mod` возвращает NULL.

- `number`: делимое.
- `divisor`: делитель.

Пример №-1

Описание

Найти остаток от деления 5 на 3.

Код

```
Number.Mod(5, 3)
```

Результат

```
2
```

Number.Permutations

Number.Permutations(setSize as number, permutationSize as number) as number

Описание

Возвращает число перестановок, которое можно создать из нескольких элементов `setSize`, с указанным размером перестановки `permutationSize`.

Пример №-1

Описание

Найти число перестановок для 5 элементов по 3.

Код

```
Number.Permutations(5, 3)
```

Результат

```
60
```

Number.Power

Number.Power(number as number, power as number) as number

Описание

Возвращает результат возведения "`number`" в степень "`power`". Если "`number`" или "`power`" равны NULL, `Number.Power` возвращает NULL.

- `number`: основание.
- `power`: показатель степени.

Пример №-1

Описание

Найти значение 5 в степени 3 (5 в кубе).

Код

```
Number.Power(5, 3)
```

Результат

```
125
```

Number.Random

Number.Random() as number

Описание

Возвращает случайное число от 0 до 1.

Пример №-1

Описание

Возвращает случайное число.

Код

```
Number.Random()
```

Результат

```
0.919303
```

Number.RandomBetween

Number.RandomBetween(bottom as number, top as number) as number

Описание

Возвращает случайное число, расположенное между `bottom` и `top`.

Пример №-1

Описание

Получить случайное число между 1 и 5.

Код

```
Number.RandomBetween(1, 5)
```

Результат

```
2.546797
```

Number.Round

Number.Round(number as number, optional digits as number, optional roundingMode as number) as number

Описание

Возвращает результат округления "`number`" до ближайшего числа. Если "`number`" имеет значение NULL, то `Number.Round` возвращает NULL. "`number`" округляется до ближайшего целого числа, если не указан необязательный параметр "`digits`". Если задан параметр "`digits`", "`number`" округляется до `digits` десятичных разрядов. Необязательный параметр "`roundingMode`" определяет направление округления, когда есть несколько равнозначных чисел, до которых выполняется округление (возможные значения см. в `RoundingMode.Type`).

Пример №-1

Описание

Округлить 1,2345 до 3 десятичных разрядов (с округлением в меньшую сторону).

Код

```
Number.Round(1.2345, 3, RoundingMode.Down)
```

Результат

```
1.234
```

Пример №-2

Описание

Округлить 1,234 до ближайшего целого числа.

Код

```
Number.Round(1.234)
```

Результат

```
1
```

Пример №-3

Описание

Округлить 1,2345 до 2 десятичных разрядов.

Код

```
Number.Round(1.2345, 2)
```

Результат

```
1.23
```

Пример №-4

Описание

Округлить 1,2345 до 3 десятичных разрядов (с округлением в большую сторону).

Код

```
Number.Round(1.2345, 3, RoundingMode.Up)
```

Результат

```
1.235
```

Пример №-5

Описание

Округлить 1,56 до ближайшего целого числа.

Код

```
Number.Round(1.56)
```

Результат

```
2
```

Number.RoundAwayFromZero

Number.RoundAwayFromZero(number as number, optional digits as number) as number

Описание

Возвращает результат округления `number` в зависимости от знака числа. Эта функция будет округлять положительные числа в большую сторону, а отрицательные - в меньшую. Если задано `digits`, то `number` округляется до заданного как `digits` числа десятичных цифр.

Пример №-1

Описание

Округлить число 1,2 в сторону от нуля.

Код

```
Number.RoundAwayFromZero(1.2)
```

Результат

```
2
```

Пример №-2

Описание

Округлить число –1,234 от нуля до двух знаков после запятой.

Код

```
Number.RoundAwayFromZero(-1.234, 2)
```

Результат

```
-1.24
```

Пример №-3

Описание

Округлить число -1,2 в сторону от нуля.

Код

```
Number.RoundAwayFromZero(-1.2)
```

Результат

```
-2
```

Number.RoundDown

Number.RoundDown(number as number, optional digits as number) as number

Описание

Возвращает результат округления `number` в меньшую сторону к предыдущему наибольшему целому числу. Если `number` имеет значение NULL, то `Number.RoundDown` возвращает NULL. Если задано `digits`, то `number` округляется до заданного как `digits` числа десятичных цифр.

Пример №-1

Описание

Округлить 1,999 в меньшую сторону до целого числа.

Код

```
Number.RoundDown(1.999)
```

Результат

```
1
```

Пример №-2

Описание

Округлить 1,234 в меньшую сторону до целого числа.

Код

```
Number.RoundDown(1.234)
```

Результат

```
1
```

Пример №-3

Описание

Округлить 1,999 до двух знаков.

Код

```
Number.RoundDown(1.999, 2)
```

Результат

```
1.99
```

Number.RoundTowardZero

Number.RoundTowardZero(number as number, optional digits as number) as number

Описание

Возвращает результат округления `number` в зависимости от знака числа. Эта функция будет округлять положительные числа в меньшую сторону, а отрицательные - в большую. Если задано `digits`, то `number` округляется до заданного как `digits` числа десятичных цифр.

Number.RoundUp

Number.RoundUp(number as number, optional digits as number) as number

Описание

Возвращает результат округления `number` в меньшую сторону к предыдущему наибольшему целому числу. Если `number` имеет значение NULL, то `Number.RoundDown` возвращает NULL. Если задано `digits`, то `number` округляется до заданного как `digits` числа десятичных цифр.

Пример №-1

Описание

Округлить 1,234 в большую сторону до целого числа.

Код

```
Number.RoundUp(1.234)
```

Результат

```
2
```

Пример №-2

Описание

Округлить 1,999 в большую сторону до целого числа.

Код

```
Number.RoundUp(1.999)
```

Результат

```
2
```

Пример №-3

Описание

Округлить число 1,234 до двух знаков после запятой.

Код

```
Number.RoundUp(1.234, 2)
```

Результат

```
1.24
```

Number.Sign

Number.Sign(number as number) as number

Описание

Возвращает 1, если `number` является положительным числом, значение -1, если отрицательным, и 0, если оно равно нулю. Если `number` имеет значение NULL, то `Number.Sign` возвращает NULL.

Пример №-1

Описание

Определить знак числа -182.

Код

```
Number.Sign(-182)
```

Результат

```
-1
```

Пример №-2

Описание

Определить знак числа 182.

Код

```
Number.Sign(182)
```

Результат

```
1
```

Пример №-3

Описание

Определить знак числа 0.

Код

```
Number.Sign(0)
```

Результат

```
0
```

Number.Sin

Number.Sin(number as number) as number

Описание

Возвращает синус `number`.

Пример №-1

Описание

Найти синус угла 0.

Код

```
Number.Sin(0)
```

Результат

```
0
```

Number.Sinh

Number.Sinh(number as number) as number

Описание

Возвращает гиперболический синус `number`.

Number.Sqrt

Number.Sqrt(number as number) as number

Описание

Возвращает квадратный корень числа `number`. Если `number` имеет значение NULL, то `Number.Sqrt` возвращает NULL. Если это отрицательное значение, то возвращается значение `Number.NaN` (нечисловое).

Пример №-1

Описание

Найти квадратный корень числа 625.

Код

```
Number.Sqrt(625)
```

Результат

```
25
```

Пример №-2

Описание

Найти квадратный корень числа 85.

Код

```
Number.Sqrt(85)
```

Результат

```
9.2195444572928871
```

Number.Tan

Number.Tan(number as number) as number

Описание

Возвращает тангенс `number`.

Пример №-1

Описание

Найти тангенс угла 1.

Код

```
Number.Tan(1)
```

Результат

```
1.5574077246549023
```

Number.Tanh

Number.Tanh(number as number) as number

Описание

Возвращает гиперболичный тангенс `number`.

Number.ToText

Number.ToText(number as number, optional format as text, optional culture as text) as text

Описание

Преобразует числовое значение "`number`" в текстовое в соответствии с форматом, указанным в "`format`". В качестве формата используется одиночный символ, к которому может добавляться число, определяющее точность. Следующие параметры форматирования могут быть использованы для "`format`".

- "D" или "d". (Десятичное.) Результат представляется в виде целого числа. Описатель точности определяет количество цифр в выходных данных.
- "E" или "e". (Экспоненциальное [инженерное].) Экспоненциальная нотация. Описатель точности определяет максимальное количество десятичных разрядов (по умолчанию 6).
- "F" или "f". (С фиксированной запятой.) Целая и десятичная дробная части.
- "G" или "g". (Общее.) Наиболее компактная форма из формы с фиксированной запятой и экспоненциальной.
- "N" или "n". (Число.) Целая и десятичная дробная части с разделителями разрядов и десятичным разделителем.
- "P" или "p". (Проценты.) Число, умноженное на 100 и отображаемое со знаком процента.
- "R" или "r". (Для обратного преобразования.) Текстовое значение, которое может быть преобразовано обратно в идентичное число. Описатель точности не учитывается.
- "X" или "x". (Шестнадцатеричное.) Шестнадцатеричное текстовое значение.

Пример №-1

Описание

Представление числа в виде текста в экспоненциальном формате.

Код

```
Number.ToText(4, "e")
```

Результат

```
"4.000000e+000"
```

Пример №-2

Описание

Форматирование числа в виде текста в десятичном формате с ограниченной точностью.

Код

```
Number.ToText(-0.1234, "P1")
```

Результат

```
"-12.3 %"
```

Пример №-3

Описание

Представление числа в виде текста без указания формата.

Код

```
Number.ToText(4)
```

Результат

```
"4"
```

OData

OData.Feed

OData.Feed(serviceUri as text, optional headers as record, optional options as any) as any

Описание

Возвращает таблицу веб-каналов OData, предоставляемую службой OData, по универсальному коду ресурса (URI) "`serviceUri`" (заголовки: `headers`). Можно использовать логическое значение, указывающее, должны ли использоваться одновременные подключения, или дополнительный параметр записи (`options`), управляющий следующими параметрами:

- `Query`: добавление параметров запроса в URL-адрес программными средствами без маскировки escape-символами.
- `Headers`: указание этого значения как записи приведет к добавлению дополнительных заголовков в HTTP-запрос.
- `ExcludedFromCacheKey`: указание этого значения как списка приведет к исключению указанных ключей заголовков HTTP из расчета данных кэширования.
- `ApiKeyName`: если на целевом сайте упоминается ключ API, этот параметр можно использовать для указания названия (но не значения) параметра ключа, который должен использоваться в URL-адресе. Фактическое значение ключа указывается в учетных данных.
- `Timeout`: указание этого значения как длительности приведет к изменению времени ожидания для HTTP-запроса. Значение по умолчанию — 600 секунд.
- `EnableBatch`: логическое значение (True или False), которое разрешает или запрещает создание запроса OData $batch, если значение MaxUriLength превышено (по умолчанию — False).
- `MaxUriLength`: число, указывающее предельную длину разрешенного универсального кода ресурса, который отправляется в службу OData. Если оно превышено и значение EnableBatch равно True, то запрос будет направлен в конечную точку OData $batch, в противном случае произойдет сбой (значение по умолчанию — 2048).
- `Concurrent`: логическое выражение (True или False). Если задано значение True, запросы к службе осуществляются параллельно. Если задано значение False, запросы осуществляются последовательно. Если значение не указано, оно определяется аннотацией службы AsynchronousRequestsSupported. Если в службе не определена поддержка AsynchronousRequestsSupported, запросы будут выполняться последовательно.
- `ODataVersion`: число (3 или 4), указывающее версию протокола OData, которую следует использовать для этой службы OData. Если ничего не указано, запрашиваются все поддерживаемые версии. Версия службы определяется заголовком версии OData-Version, возвращаемым службой.
- `FunctionOverloads`: логическое значение (True или False): если задано значение True, перегрузки импорта функции будут перечислены в отдельных записях в навигаторе; если задано значение False, перегрузки импорта функции будут перечислены в виде одной функции Union в навигаторе. Значение по умолчанию для версии 3 — False, для версии 4 — True.
- `MoreColumns`: логическое значение (True или False): если задано значение True, добавляется столбец "More Columns" для каждого веб-канала сущности, содержащего открытые и полиморфные типы. Столбец будет содержать поля, не объявленные в базовом типе. Если задано значение False, это поле будет отсутствовать. Значение по умолчанию — False.

- `IncludeAnnotations`: список названий терминов или шаблонов с пространствами имен с разделителями-запятыми, в который включается подстановочный знак "*". По умолчанию ни одна из аннотаций не включена.

Odbc

Odbc.DataSource

Odbc.DataSource(connectionString as any, optional options as record) as table

Описание

Возвращает список таблиц и представлений SQL из источника данных ODBC, заданного в строке подключения `connectionString`. `connectionString` может быть текстом или записью пар "свойство-значение". Значения свойств могут быть текстом или числом. Дополнительный параметр записи `options` указывает дополнительные свойства. Запись может содержать следующие поля.

- `CreateNavigationProperties`: логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `HierarchicalNavigation`: логическое значение (True или False), которое указывает, следует ли отображать таблицы сгруппированными по именам схем и каталогов, если это поддерживается. Значение по умолчанию — False.

Odbc.Query

Odbc.Query(connectionString as any, query as text) as table

Описание

Возвращает результат запуска `query` со строкой подключения `connectionString`, используя ODBC. `connectionString` может быть текстом или записью пар значений свойств. Значения свойств могут быть текстом или числом.

OleDb

OleDb.DataSource

OleDb.DataSource(connectionString as any, optional options as record) as table

Описание

Возвращает таблицу с таблицами и представлениями SQL из источника данных OLE DB, указанными в строке подключения `connectionString`. `connectionString` может быть текстом или записью из пар "значение-свойство". Значения свойств могут быть текстом или числом. Для указания дополнительных свойств можно предоставить необязательный параметр записи, `options`. Запись может содержать следующие поля:

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.

- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `HierarchicalNavigation` : Логическое значение (true или false), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — true.

Пример указания параметра записи: [параметр1 = значение1, параметр2 = значение2...] или [Query = "select ..."].

OleDb.Query

OleDb.Query(connectionString as any, query as text) as table

Описание

Возвращает результат запуска `query` со строкой подключения `connectionString`, используя OLE DB. `connectionString` может быть текстом или записью пар значений свойств. Значения свойств могут быть текстом или числом.

Oracle

Oracle.Database

Oracle.Database(server as text, optional options as record) as table

Описание

Возвращает список таблиц и представлений SQL из базы данных Oracle на сервере `server`. Дополнительно к имени сервера через двоеточие может быть указан порт. Необязательный параметр записи `options` может быть указан для управления следующими параметрами.

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

Percentage

Percentage.From

Percentage.From(value as any, optional culture as text) as number

Описание

Возвращает значение `percentage` от заданного `value`. Если значение заданного `value` равно `null`, `Percentage.From` возвращает `null`. Если значение заданного `value` — это `text` с символом процента в конце, то будет возвращено преобразованное десятичное число. В противном случае используйте `Number.From` для преобразования в `number`.

Пример №-1

Описание

Получапет значение `percentage` для `"12.3 %"`.

Код

```
Percentage.From("12.3%")
```

Результат

```
0.123
```

PostgreSQL

PostgreSQL.Database

PostgreSQL.Database(server as text, database as text, optional options as record) as table

Описание

Возвращает список таблиц и представлений SQL, доступных в базе данных PostgreSQL на сервере `server` в экземпляре базы данных `database`. Дополнительно к имени сервера через двоеточие может быть указан порт. Необязательный параметр записи `options` может быть указан для управления следующими параметрами.

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.

- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

RData

RData.FromBinary

RData.FromBinary(stream as binary) as any

Описание

Возвращает запись кадров данных из файла RData.

Record

Record.AddField

Record.AddField(record as record, fieldName as text, value as any, optional delayed as logical) as record

Описание

Добавляет поле к записи `record` для заданного имени поля `fieldName` и значения `value`.

Пример №-1

Описание

Добавление поля Address к записи.

Код

```
Record.AddField([CustomerID = 1, Name = "Bob", Phone = "123-4567"], "Address", "123 Main St.")
```

Результат

```
[CustomerID=1, Name= "Bob", Phone="123-4567", Address="123 Main St."]
```

Record.Combine

Record.Combine(records as list) as record

Описание

Объединяет записи в данном `records`. Если `records` содержит значения, отличные от записи, возвращается ошибка.

Пример №-1

Описание

Создание комбинированной записи из записей.

Код

```
Record.Combine({ [CustomerID =1, Name ="Bob"] , [Phone =  "123-4567"]})
```

Результат

```
[CustomerID=1, Name="Bob", Phone="123-4567"]
```

Record.Field

Record.Field(record as record, field as text) as any

Описание

Возвращает значение указанного `field` в `record`. Если поле не найдено, возникает исключение.

Пример №-1

Описание

Нахождение значения поля CustomerID в записи.

Код

```
Record.Field([CustomerID = 1, Name = "Bob", Phone = "123-4567"], "CustomerID")
```

Результат

```
1
```

Record.FieldCount

Record.FieldCount(record as record) as number

Описание

Возвращает число полей в записи `record`.

Пример №-1

Описание

Нахождение числа полей в записи.

Код

```
Record.FieldCount([CustomerID = 1, Name = "Bob"])
```

Результат

```
2
```

Record.FieldNames

Record.FieldNames(record as record) as list

Описание

Возвращает имена полей в записи `record` в виде текста.

Пример №-1

Описание

Нахождение имен полей в записи.

Код

```
Record.FieldNames([OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0])
```

Результат

```
{"OrderID","CustomerID", "Item", "Price"}
```

Record.FieldOrDefault

Record.FieldOrDefault(record as record, field as text, optional defaultValue as any) as any

Описание

Возвращает значение указанного поля `field` в записи `record`. Если поле не найдено, то возвращается необязательное значение `defaultValue`.

Пример №-1

Описание

Нахождение значения поля Phone в записи или возврат значения Null, если оно не существует.

Код

```
Record.FieldOrDefault([CustomerID =1, Name="Bob"], "Phone")
```

Результат

```
null
```

Пример №-2

Описание

Нахождение значения поля Phone в записи или возврат значения по умолчанию, если оно не существует.

Код

```
Record.FieldOrDefault([CustomerID =1, Name="Bob"], "Phone", "123-4567")
```

Результат

```
"123-4567"
```

Record.FieldValues

Record.FieldValues(record as record) as list

Описание

Возвращает список значений полей в записи `record`.

Пример №-1

Описание

Нахождение значений полей в записи.

Код

```
Record.FieldValues([CustomerID = 1, Name = "Bob", Phone = "123-4567"])
```

Результат

```
{1, "Bob", "123-4567"}
```

Record.FromList

Record.FromList(list as list, fields as any) as record

Описание

Возвращает запись для данного списка `list` значений полей и набора полей. `fields` можно задать либо списком текстовых значений, либо типом "запись". Если поля неуникальны, выдается ошибка.

Пример №-1

Описание

Создать запись из списка значений полей и записи.

Код

```
Record.FromList({1, "Bob", "123-4567"}, type [CustomerID = number, Name = text, Phone = number])
```

Результат

```
[CustomerID = 1, Name = "Bob", Phone = "123-4567"]
```

Пример №-2

Описание

Создать запись из списка значений полей и списка имен полей.

Код

```
Record.FromList({1, "Bob", "123-4567"}, {"CustomerID", "Name", "Phone"})
```

Результат

```
[CustomerID = 1, Name = "Bob", Phone = "123-4567"]
```

Record.FromTable

Record.FromTable(table as table) as record

Описание

Возвращает запись из таблицы записей `table`, которая содержит имена полей и имена значений `{[Name = name, Value = value]}`. Если имена полей неуникальны, вызывает исключение.

Пример №-1

Описание

Создание записи из таблицы формы Table.FromRecords({[Name = "CustomerID", Value = 1], [Name = "Name", Value = "Bob"], [Name = "Phone", Value = "123-4567"]}).

Код

```
Record.FromTable(Table.FromRecords({[Name = "CustomerID", Value = 1], [Name = "Name", Value = "Bob"], [Name = "Phone", Value = "123-4567"]}))
```

Результат

```
[CustomerID = 1, Name = "Bob", Phone = "123-4567"]
```

Record.HasFields

Record.HasFields(record as record, fields as any) as logical

Описание

Указывает, содержит ли запись `record` поля, заданные в `fields`, возвращая логическое значение (true или false). Значения нескольких полей можно указать с помощью списка.

Пример №-1

Описание

Проверка, содержится ли в записи поле CustomerID.

Код

```
Record.HasFields([CustomerID = 1, Name = "Bob", Phone = "123-4567"],"CustomerID")
```

Результат

```
true
```

Пример №-2

Описание

Проверка, содержатся ли в записи поля CustomerID и Address.

Код

```
Record.HasFields([CustomerID = 1, Name = "Bob", Phone = "123-4567"],{"CustomerID",
"Address"})
```

Результат

```
false
```

Record.RemoveFields

Record.RemoveFields(record as record, fields as any, optional missingField as number) as record

Описание

Возвращает запись, которая удаляет все поля, указанные в списке `fields`, из входных данных `record`. Если указанное поле не существует, возникает исключение.

Пример №-1

Описание

Удаление поля Price и Item из записи.

Код

```
Record.RemoveFields([CustomerID=1, Item = "Fishing rod", Price=18.00], {"Price",
"Item"})
```

Результат

```
[CustomerID=1]
```

Пример №-2

Описание

Удаление поля Price из записи.

Код

```
Record.RemoveFields([CustomerID=1, Item = "Fishing rod", Price=18.00], "Price")
```

Результат

```
[CustomerID=1, Item="Fishing rod"]
```

Record.RenameFields

Record.RenameFields(record as record, renames as list, optional missingField as number) as record

Описание

Возвращает запись после присвоения полям во входных данных `record` новых имен, указанных в списке `renames`. Для нескольких переименований можно использовать вложенный список ({ {старое1, новое1}, {старое2, новое2} }.

Пример №-1

Описание

Переименование поля UnitPrice в Price в записи.

Код

```
Record.RenameFields([OrderID = 1, CustomerID = 1, Item = "Fishing rod", UnitPrice =
100.0], {"UnitPrice","Price"})
```

Результат

```
[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0]
```

Пример №-2

Описание

Переименование поля UnitPrice в Price, а поле OrderNum в OrderID в записи.

Код

```
Record.RenameFields([OrderNum = 1, CustomerID = 1, Item = "Fishing rod", UnitPrice =
100.0], {{"UnitPrice", "Price"}, {"OrderNum", "OrderID"}})
```

Результат

```
[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0]
```

Record.ReorderFields

Record.ReorderFields(record as record, fieldOrder as list, optional missingField as number) as record

Описание

Возвращает запись после изменения порядка полей в `record` согласно порядку, указанному в списке `fieldOrder`. Значения полей сохраняются, а поля, не указанные в `fieldOrder`, остаются в исходном положении.

Пример №-1

Описание

Изменение порядка некоторых полей в записи.

Код

```
Record.ReorderFields([CustomerID= 1, OrderID = 1, Item = "Fishing rod", Price = 100.0],
{"OrderID", "CustomerID"})
```

Результат

```
[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0]
```

Record.SelectFields

Record.SelectFields(record as record, fields as any, optional missingField as number) as record

Описание

Возвращает запись, которая включает только поля, указанные в списке `fields`, из входных данных `record`.

Пример №-1

Описание

Выбор полей Item и Price в записи.

Код

```
Record.SelectFields( [OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0]
, {"Item", "Price"})
```

Результат

```
[Item = "Fishing rod", Price = 100]
```

Record.ToList

Record.ToList(record as record) as list

Описание

Возвращает список значений, содержащих значения полей из входных данных `record`.

Пример №-1

Описание

Извлечь значения полей из записи.

Код

```
Record.ToList([A = 1, B = 2, C = 3])
```

Результат

```
{1, 2, 3}
```

Record.ToTable

Record.ToTable(record as record) as table

Описание

Возвращает таблицу, содержащую столбцы `Name` и `Value` со строкой для каждого поля в `record`.

Пример №-1

Описание

Возврат таблицы из записи.

Код

```
Record.ToTable([OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0])
```

Результат

```
Table.FromRecords({[Name="OrderID", Value=1],[Name="CustomerID", Value=1],[Name="Item",
Value="Fishing rod"],[Name="Price", Value=100]})
```

Record.TransformFields

Record.TransformFields(record as record, transformOperations as list, optional missingField as number) as record

Описание

Возвращает запись после применения преобразований, указанных в списке `transformOperations`, в `record`. Одновременно могут быть преобразованы одно или несколько полей.

В случае одного преобразуемого поля предполагается, что `transformOperations` - список 2 с элементами. Первый элемент в `transformOperations` указывает имя поля, а второй элемент в `transformOperations` указывает функцию, используемую для преобразования. Например, `{"Quantity", Number.FromText}`
В случае преобразования нескольких полей предполагается, что `transformOperations` - список списков, где каждый внутренний список представляет собой пару из имени поля и операции преобразования. Например, `{{"Quantity",Number.FromText},{"UnitPrice", Number.FromText}}`

Пример №-1

Описание

Преобразовать поле "Price" в число.

Код

```
Record.TransformFields([OrderID = 1, CustomerID= 1, Item = "Fishing rod", Price =
"100.0"], {"Price", Number.FromText})
```

Результат

```
[OrderID = 1, CustomerID= 1, Item = "Fishing rod", Price = 100]
```

Пример №-2

Описание

Преобразовать поля "OrderID" и "Price" в числа.

Код

```
Record.TransformFields(
    [OrderID ="1", CustomerID= 1, Item = "Fishing rod", Price = "100.0"],
    {{"OrderID", Number.FromText}, {"Price",Number.FromText}})
```

Результат

```
[OrderID = 1, CustomerID= 1, Item = "Fishing rod", Price = 100]
```

Replacer

Replacer.ReplaceText

Replacer.ReplaceText(text as text, old as text, new as text) as text

Описание

Заменяет текст `old` в исходном `text` текстом `new`. Эту функцию замены можно использовать в `List.ReplaceValue` и `Table.ReplaceValue`.

Пример №-1

Описание

Заменить текст "hE" на "He"в строке "hEllo world".

Код

```
Replacer.ReplaceText("hEllo world","hE","He")
```

Результат

```
"Hello world"
```

Replacer.ReplaceValue

Replacer.ReplaceValue(value as any, old as any, new as any) as any

Описание

Заменяет значение `old` в исходном `value` значением `new`. Эту функцию замены можно использовать в `List.ReplaceValue` и `Table.ReplaceValue`.

Пример №-1

Описание

Заменить значение 11 значением 10.

Код

```
Replacer.ReplaceValue(11, 11, 10)
```

Результат

```
10
```

Resource

Resource.Access

Resource.Access(resource as any, optional nativeQuery as text) as any

Описание

Resource.Access

RowExpression

RowExpression.Column

RowExpression.Column(columnName as text) as record

Описание

Возвращает AST, представляющий доступ к столбцу `columnName` строки в выражении строки.

Пример №-1

Описание

Создает AST, представляющий доступ к столбцу CustomerName.

Код

```
RowExpression.Column("CustomerName")
```

Результат

```
[
    Kind = "FieldAccess",
    Expression = RowExpression.Row,
    MemberName = "CustomerName"
]
```

RowExpression.From

RowExpression.From(function as function) as record

Описание

Возвращает AST для тела `function`, нормализованного в *выражение строки*:

- Функция должна быть лямбдой с 1 аргументом.
- Все ссылки на параметры функции заменяются `RowExpression.Row`.
- Все ссылки на столбцы заменяются `RowExpression.Column(`*`columnName`*`)`.
- Упрощенный AST будет содержать только узлы следующих видов:
    - `Constant`
    - `Invocation`

- Unary
- Binary
- If
- FieldAccess

Происходит ошибка, если AST выражения строки невозможно вернуть для тела `function`.

Пример №-1

Описание

Возвращает AST для тела функции `each [CustomerID] = "ALFKI"`

Код

```
RowExpression.From(each [CustomerName] = "ALFKI")
```

Результат

```
[
    Kind = "Binary",
    Operator = "Equals",
    Left = RowExpression.Column("CustomerName"),
    Right =
    [
        Kind = "Constant",
        Value = "ALFKI"
    ]
]
```

Salesforce

Salesforce.Data

Salesforce.Data(optional loginUrl as any, optional options as record) as table

Описание

Возвращает объекты из указанной в учетных данных учетной записи Salesforce. Учетная запись будет подключена с помощью заданного окружения `loginUrl`. Если оно не указано, учетная запись подключится к рабочей среде (https://login.salesforce.com). Чтобы задать дополнительные свойства, можно указать необязательный параметр записи `options`. Запись может содержать следующие поля:

- `CreateNavigationProperties` : Логическое значение (true или false), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — false.
- `ApiVersion` : Версия API Salesforce для этого запроса. По умолчанию используется версия API 29.0.

Salesforce.Reports

Salesforce.Reports(optional loginUrl as text, optional options as record) as table

Описание

Возвращает отчеты из указанной в учетных данных учетной записи Salesforce. Учетная запись будет подключена с помощью заданного окружения `loginUrl`. Если оно не указано, учетная запись подключится к рабочей среде (https://login.salesforce.com). Чтобы задать дополнительные свойства, можно указать необязательный параметр записи `options`. Запись может содержать следующие поля:

- `ApiVersion` : Версия API Salesforce для этого запроса. По умолчанию используется версия API 29.0.

SapBusinessObjects

SapBusinessObjects.Universes

SapBusinessObjects.Universes(url as text, optional options as record) as table

Описание

Возвращает сущности в службе SapBusinessObjects BI `url`. Необязательный параметр записи `options` может содержать дополнительные свойства. Запись может содержать следующие поля.

- `Culture`: укажите язык и региональные параметры для данных. Также называется предпочитаемой локалью для просмотра.

SapHana

SapHana.Database

SapHana.Database(server as text, optional options as record) as table

Описание

Возвращает таблицу многомерных пакетов из базы данных HANA SAP `server`. Может быть указан необязательный параметр записи `options` для управления следующими параметрами:

- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.

SharePoint

SharePoint.Contents

SharePoint.Contents(url as text, optional options as record) as table

Описание

Возвращает таблицу, содержащую по одной строке для каждой папки и документа, найденных на указанном сайте SharePoint (`url`). Каждая строка представляет свойства файла или папки и ссылку на содержимое. Можно указать `options` для управления следующими параметрами:

- `ApiVersion` : Число (14 или 15) или значение "Auto" (Автоматически), указывающее версию API SharePoint для использования на этом сайте. Если значение не задано, используется версия API 14. Если задано значение Auto, версия сервера будет по возможности определяться

автоматически. В случае невозможности будет использоваться значение по умолчанию — 14. Для неанглоязычных сайтов SharePoint требуется по меньшей мере версия 15.

SharePoint.Files

SharePoint.Files(url as text, optional options as record) as table

Описание

Возвращает таблицу, содержащую по одной строке для каждого документа, найденного на указанном сайте SharePoint (`url`) и во вложенных папках. Каждая строка представляет свойства файла или папки и ссылку на содержимое. Можно указать `options` для управления следующими параметрами:

- `ApiVersion` : Число (14 или 15) или значение "Auto" (Автоматически), указывающее версию API SharePoint для использования на этом сайте. Если значение не задано, используется версия API 14. Если задано значение Auto, версия сервера будет по возможности определяться автоматически. В случае невозможности будет использоваться значение по умолчанию — 14. Для неанглоязычных сайтов SharePoint требуется по меньшей мере версия 15.

SharePoint.Tables

SharePoint.Tables(url as text, optional options as record) as table

Описание

Возвращает таблицу, содержащую по одной строке для каждого элемента списка, найденного в указанном списке SharePoint (`url`). Каждая строка содержит свойства списка. Можно указать `options` для управления следующими параметрами:

- `ApiVersion` : Число (14 или 15) или значение "Auto" (Автоматически), указывающее версию API SharePoint для использования на этом сайте. Если значение не задано, используется версия API 14. Если задано значение Auto, версия сервера будет по возможности определяться автоматически. В случае невозможности будет использоваться значение по умолчанию — 14. Для неанглоязычных сайтов SharePoint требуется по меньшей мере версия 15.

Single

Single.From

Single.From(value as any, optional culture as text) as number

Описание

Возвращает значение Single `number` из данного значения `value`. Если данное значение `value` является значением `null`, `Single.From` возвращает `null`. Если данное значение `value` является значением `number` в диапазоне Single, возвращается `value`, иначе возвращается ошибка. Если данное значение `value` является значением любого другого типа, см. `Number.FromText` для преобразования его в значение `number`, после чего применяется предыдущее утверждение о преобразовании значения `number` в значение Single `number`.

Пример №-1

Описание

Получить значение Single `number` *для* `"1.5"`*.*

Код

```
Single.From("1.5")
```

Результат

```
1.5
```

Soda

Soda.Feed

Soda.Feed(url as text) as table

Описание

Возвращает таблицу с содержимым по указанному URL-адресу `url`, отформатированному согласно SODA 2.0 API. URL-адрес должен указывать на действительный SODA-совместимый источник, который заканчивается на расширение CSV.

Splitter

Splitter.SplitByNothing

Splitter.SplitByNothing() as function

Описание

Возвращает функцию, которая не разбивает текст, возвращая свой аргумент как единый список элементов.

Splitter.SplitTextByAnyDelimiter

Splitter.SplitTextByAnyDelimiter(delimiters as list, optional quoteStyle as number, optional startAtEnd as logical) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список по всем указанным разделителям.

Splitter.SplitTextByDelimiter

Splitter.SplitTextByDelimiter(delimiter as text, optional quoteStyle as number) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список в соответствии с указанным разделителем.

Splitter.SplitTextByEachDelimiter

Splitter.SplitTextByEachDelimiter(delimiters as list, optional quoteStyle as number, optional startAtEnd as logical) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список по всем заданным разделителям в последовательности.

Splitter.SplitTextByLengths

Splitter.SplitTextByLengths(lengths as list, optional startAtEnd as logical) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список по каждой указанной длине.

Splitter.SplitTextByPositions

Splitter.SplitTextByPositions(positions as list, optional startAtEnd as logical) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список по всем указанным позициям.

Splitter.SplitTextByRanges

Splitter.SplitTextByRanges(ranges as list, optional startAtEnd as logical) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список по заданным значениям смещения и длины.

Splitter.SplitTextByRepeatedLengths

Splitter.SplitTextByRepeatedLengths(length as number, optional startAtEnd as logical) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список в цикле, после каждого отрезка указанной длины.

Splitter.SplitTextByWhitespace

Splitter.SplitTextByWhitespace(optional quoteStyle as number) as function

Описание

Возвращает функцию, которая разбивает текст на текстовый список по пробелам.

Sql

Sql.Database

Sql.Database(server as text, database as text, optional options as record) as table

Описание

Возвращает список таблиц, представлений и хранимых функций SQL из базы данных SQL Server `database` на сервере `server`. Дополнительно к имени сервера через двоеточие или запятую может быть указан порт. Можно указать необязательный параметр записи `options` для управления следующими параметрами.

- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `MaxDegreeOfParallelism` : Число, которое задает значение предложения запроса "maxdop" в созданном запросе SQL.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.
- `MultiSubnetFailover` : Логическое значение (True или False), которое задает значение свойства "MultiSubnetFailover" в строке подключения. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

Sql.Databases

Sql.Databases(server as text, optional options as record) as table

Описание

Возвращает таблицу баз данных на указанном сервере SQL Server, `server`. Можно указать необязательный параметр записи, `options`, для управления следующими параметрами.

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `MaxDegreeOfParallelism` : Число, которое задает значение предложения запроса "maxdop" в созданном запросе SQL.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.

- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.
- `MultiSubnetFailover` : Логическое значение (True или False), которое задает значение свойства "MultiSubnetFailover" в строке подключения. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...].
Выбор запуска запроса SQL на сервере не поддерживается. Для запуска запроса SQL следует использовать `Sql.Database`.

SqlExpression

SqlExpression.SchemaFrom

SqlExpression.SchemaFrom(schema as any) as any

Описание

SqlExpression.SchemaFrom

SqlExpression.ToExpression

SqlExpression.ToExpression(sql as text, environment as record) as text

Описание

SqlExpression.ToExpression

Sybase

Sybase.Database

Sybase.Database(server as text, database as text, optional options as record) as table

Описание

Возвращает список таблиц, доступных в базе данных Sybase на сервере `server` в экземпляре базы данных `database`. Дополнительно к имени сервера через двоеточие может быть указан порт. Необязательный параметр записи `options` может быть указан для управления следующими параметрами.

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.

- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

Table

Table.AddColumn

Table.AddColumn(table as table, newColumnName as text, columnGenerator as function, optional columnType as type) as table

Описание

Добавляет столбец с именем `newColumnName` в таблицу `table`. Значения для этого столбца вычисляются с помощью заданной функции выбора `columnGenerator`, при этом каждая строка берется в качестве входных данных.

Пример №-1

Описание

Добавление столбца с именем TotalPrice в таблицу, где каждое значение является суммой значений из столбцов [Price] и [Shipping].

Код

```
Table.AddColumn(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod",
Price = 100.0, Shipping = 10.00],  [OrderID = 2, CustomerID = 1, Item = "1 lb. worms",
Price = 5.0, Shipping = 15.00],  [OrderID = 3, CustomerID = 2, Item = "Fishing net",
Price = 25.0, Shipping = 10.00]}), "TotalPrice", each [Price] + [Shipping])
```

Результат

```
Table.FromRecords({[OrderID=1,CustomerID=1,Item="Fishing rod",
Price=100,Shipping=10,TotalPrice=110],[OrderID=2,CustomerID=1,Item="1 lb. worms",
Price=5,Shipping=15,TotalPrice=20],[OrderID=3,CustomerID=2,Item="Fishing net",
Price=25,Shipping=10,TotalPrice=35]})
```

Table.AddIndexColumn

Table.AddIndexColumn(table as table, newColumnName as text, optional initialValue as number, optional increment as number) as table

Описание

Добавляет столбец с именем `newColumnName` в `table` с явно указанными значениями позиции. Необязательное значение `initialValue`, начальное значение индекса. Необязательное значение `increment` определяет, на какую величину увеличивается каждое значение индекса.

Пример №-1

Описание

Добавление столбца индекса с именем Index в таблицу.

Код

```
Table.AddIndexColumn(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name =
"Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
"Index")
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-4567",
Index=0],[CustomerID=2,Name="Jim",Phone="987-6543",
Index=1],[CustomerID=3,Name="Paul",Phone="543-7890",
Index=2],[CustomerID=4,Name="Ringo",Phone="232-1550", Index=3]})
```

Пример №-2

Описание

Добавление столбца индекса с именем index с начальным значением 10 и шагом 5 в таблицу.

Код

```
Table.AddIndexColumn(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name =
"Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
"Index", 10, 5)
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-4567",
Index=10],[CustomerID=2,Name="Jim",Phone="987-
6543",Index=15],[CustomerID=3,Name="Paul",Phone="543-
7890",Index=20],[CustomerID=4,Name="Ringo",Phone="232-1550",Index=25]})
```

Table.AddJoinColumn

Table.AddJoinColumn(table1 as table, key1 as any, table2 as any, key2 as any, newColumnName as text) as table

Описание

Соединяет строки таблицы `table1` со строками таблицы `table2` с учетом тождественности значений ключевых столбцов, выбранных в `key1` (для `table1`) и в `key2` (для `table2`). Результаты записываются в столбец с именем `newColumnName`. Эта функция работает подобно Table.Join с JoinKind из LeftOuter, за тем исключением, что результаты соединения представлены во вложенном виде, а не преобразованными в плоскую структуру.

Пример №-1

Описание

Добавить столбец соединения для ({[saleID = 1, item = "Shirt"], [saleID = 2, item = "Hat"]}) с именем "цена-наличие" из таблицы ({[saleID = 1, price = 20], [saleID = 2, price = 10]}), соединенный по [saleID].

Код

```
Table.AddJoinColumn(Table.FromRecords({[saleID = 1, item = "Shirt"], [saleID = 2, item = "Hat"]}), "saleID", () => Table.FromRecords({[saleID = 1, price = 20, stock = 1234], [saleID = 2, price = 10, stock = 5643]}), "saleID", "price")
```

Результат

```
Table.FromRecords({ [
        saleID = 1,
        item = "Shirt",
        price =Table.FromRecords({ [
                saleID = 1,
                price = 20,
                stock = 1234
            ]
        }, {
            "saleID",
            "price",
            "stock"
        })
    ], [
        saleID = 2,
        item = "Hat",
        price =Table.FromRecords({ [
                saleID = 2,
                price = 10,
                stock = 5643
            ]
        }, {
            "saleID",
            "price",
            "stock"
        })
    ]
}, {
    "saleID",
    "item",
    "price"
})
```

Table.AddKey

Table.AddKey(table as table, columns as list, isPrimary as logical) as table

Описание

Добавить ключ в `table`, где `columns` - подмножество имен столбцов таблицы `table`, которое определяет ключ, а `isPrimary` указывает, является ли ключ первичным.

Пример №-1

Описание

Добавить в {[Id = 1, Name = "Hello There"], [Id = 2, Name = "Good Bye"]} ключ, состоящий из {"Id"}, и сделать его первичным.

Код

```
let
    tableType = type table [Id = Int32.Type, Name = text],
    table = Table.FromRecords({[Id = 1, Name = "Hello There"], [Id = 2, Name = "Good
Bye"]}),
    resultTable = Table.AddKey(table, {"Id"}, true)
in
    resultTable
```

Результат

```
Table.FromRecords({[Id = 1, Name = "Hello There"], [Id = 2, Name = "Good Bye"]}, {
    "Id",
    "Name"
})
```

Table.AggregateTableColumn

Table.AggregateTableColumn(table as table, column as text, aggregations as list) as table

Описание

Агрегирует таблицы в `table`[`column`] в несколько столбцов, содержащих статистические значения для таблиц. `aggregations` используется для указания столбцов, содержащих таблицы для статистической обработки, статистические функции, которые применяются к таблицам для создания их значений, и имена столбцов групповых операций, которые требуется создать.

Пример №-1

Описание

Агрегировать столбцы таблицы в `[t]` в таблице `{[t = {[a=1, b=2, c=3], [a=2,b=4,c=6]}, b = 2]}` в сумму `[t.a]`, минимум и максимум `[t.b]` и количество значений в `[t.a]`.

Код

```
Table.AggregateTableColumn(Table.FromRecords({[t = Table.FromRecords({[a=1, b=2, c=3],
[a=2,b=4,c=6]}), b = 2]}, type table [t = table [a=number, b=number, c=number], b =
number]), "t", {{"a", List.Sum, "sum of t.a"}, {"b", List.Min, "min of t.b"}, {"b",
List.Max, "max of t.b"}, {"a", List.Count, "count of t.a"}})
```

Результат

```
Table.FromRecords({[#"sum of t.a" = 3, #"min of t.b" = 2, #"max of t.b" = 4, #"count of
t.a" = 2, b = 2]})
```

Table.AlternateRows

Table.AlternateRows(table as table, offset as number, skip as number, take as number) as table

Описание

Сохраняет исходное смещение, затем попеременно принимает и пропускает следующие строки.

- `table`: входная таблица.
- `offset`: число строк, которые должны быть сохранены перед началом итераций.
- `skip`: число строк, удаляемых в каждой итерации.
- `take`: число строк, сохраняемых в каждой итерации.

Пример №-1

Описание

Получение таблицы из таблицы, в которой, начиная с первой строки, пропускается одно значение, а затем сохраняется одно значение.

Код

```
Table.AlternateRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name =
"Paul", Phone = "543-7890"]}), 1, 1, 1)
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-
4567"],[CustomerID=3,Name="Paul",Phone="543-7890"]})
```

Table.Buffer

Table.Buffer(table as table) as table

Описание

Помещает таблицу в буфер памяти, изолируя ее из внешних изменений во время оценки.

Table.Column

Table.Column(table as table, column as text) as list

Описание

Возвращает столбец данных, указанный с помощью `column`, из таблицы `table` в виде списка.

Пример №-1

Описание

Возврат значений из столбца [Name] таблицы.

Код

```
Table.Column(Table.FromRecords({  [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), "Name")
```

Результат

```
{"Bob", "Jim", "Paul", "Ringo"}
```

Table.ColumnCount

Table.ColumnCount(table as table) as number

Описание

Возвращает количество столбцов в таблице `table`.

Пример №-1

Описание

Определение числа столбцов в таблице.

Код

```
Table.ColumnCount(Table.FromRecords({[CustomerID =1, Name ="Bob", Phone = "123-4567"],[CustomerID =2, Name ="Jim", Phone = "987-6543"],[CustomerID =3, Name ="Paul", Phone = "543-7890"]}))
```

Результат

```
3
```

Table.ColumnNames

Table.ColumnNames(table as table) as list

Описание

Возвращает имена столбцов в таблице `table` в виде текстового списка.

Пример №-1

Описание

Нахождение имен столбцов таблицы.

Код

```
Table.ColumnNames(Table.FromRecords({  [CustomerID = 1, Name = "Bob", Phone = "123-4567"],  [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,  [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}))
```

Результат

```
{"CustomerID", "Name", "Phone"}
```

Table.ColumnsOfType

Table.ColumnsOfType(table as table, listOfTypes as list) as list

Описание

Возвращает список с именами столбцов таблицы `table`, соответствующих указанным в `listOfTypes` типам.

Пример №-1

Описание

Возврат имен столбцов типа Number.Type из таблицы.

Код

```
Table.ColumnsOfType(Table.FromRecords({[a=1,b="hello"]}, type table[a=Number.Type,
b=Text.Type]), {type number})
```

Результат

```
{"a"}
```

Table.Combine

Table.Combine(tables as list) as table

Описание

Возвращает таблицу, полученную в результате объединения списка таблиц `tables`. Таблицы должны иметь одинаковую структуру типов строк.

Пример №-1

Описание

Объединение трех таблиц.

Код

```
Table.Combine({Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"]}),
Table.FromRecords({[CustomerID = 2, Name = "Jim", Phone = "987-6543"]
}),Table.FromRecords({[CustomerID = 3, Name = "Paul", Phone = "543-7890"]})})
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-
4567"],[CustomerID=2,Name="Jim",Phone="987-6543"],[CustomerID=3,Name="Paul",Phone="543-
7890"]})
```

Table.CombineColumns

Table.CombineColumns(table as table, sourceColumns as list, combiner as function, column as text) as table

Описание

Объединяет указанные столбцы в новый столбец с помощью заданной функции объединения.

Table.Contains

Table.Contains(table as table, row as record, optional equationCriteria as any) as logical

Описание

Указывает, появляется ли указанная запись `row` в виде строки в `table`. Для управления сравнением строк таблицы может быть указан необязательный параметр `equationCriteria`.

Пример №-1

Описание

Определение, содержит ли таблица строку.

Код

```
Table.Contains(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
[Name="Bob"])
```

Результат

```
true
```

Пример №-2

Описание

Определение, содержит ли таблица строку, путем сравнения только столбца [Name].

Код

```
Table.Contains(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
[CustomerID=4, Name="Bob"], "Name")
```

Результат

```
true
```

Пример №-3

Описание

Определение, содержит ли таблица строку.

Код

```
Table.Contains(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
[Name="Ted"])
```

Результат

```
false
```

Table.ContainsAll

Table.ContainsAll(table as table, rows as list, optional equationCriteria as any) as logical

Описание

Указывает, появляются ли все записи, указанные в списке записей `rows`, в виде строк в `table`. Для управления сравнением строк таблицы может быть указан необязательный параметр `equationCriteria`.

Пример №-1

Описание

Определение, содержит ли таблица все строки, путем сравнения только столбца [CustomerID].

Код

```
Table.ContainsAll(  Table.FromRecords( {  [CustomerID = 1, Name = "Bob", Phone = "123-
4567"],  [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name =
"Paul", Phone = "543-7890"] ,  [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
{[CustomerID=1, Name="Bill"],[CustomerID=2, Name="Fred"]},    "CustomerID")
```

Результат

```
true
```

Пример №-2

Описание

Определение, содержит ли таблица все строки.

Код

```
Table.ContainsAll(  Table.FromRecords( {  [CustomerID = 1, Name = "Bob", Phone = "123-4567"],  [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,  [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), {[CustomerID=1, Name="Bill"],[CustomerID=2, Name="Fred"]})
```

Результат

```
false
```

Table.ContainsAny

Table.ContainsAny(table as table, rows as list, optional equationCriteria as any) as logical

Описание

Указывает, появляется ли любая запись, указанная в списке записей `rows`, в виде строки в `table`. Для управления сравнением строк таблицы может быть указан необязательный параметр `equationCriteria`.

Пример №-1

Описание

Определить, содержит ли таблица `({[a = 1, b = 2], [a = 3, b = 4]})` строку `[a = 1, b = 2]` или `[a = 3, b = 5]`.

Код

```
Table.ContainsAny(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {[a = 1, b = 2], [a = 3, b = 5]})
```

Результат

```
true
```

Пример №-2

Описание

Определить, содержит ли таблица `(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}))` строку `[a = 1, b = 3]` или `[a = 3, b = 5]`, сравнивая только столбец [a].

Код

```
Table.ContainsAny(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {[a = 1, b = 3], [a = 3, b = 5]}, "a")
```

Результат

```
true
```

Пример №-3

Описание

Определить, содержит ли таблица `({[a = 1, b = 2], [a = 3, b = 4]})` *строку* `[a = 1, b = 3]` *или* `[a = 3, b = 5]`.

Код

```
Table.ContainsAny(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {[a = 1, b = 3], [a = 3, b = 5]})
```

Результат

```
false
```

Table.DemoteHeaders

Table.DemoteHeaders(table as table) as table

Описание

Переносит заголовки столбцов (т. е. имена столбцов) в первую строку значений. Имена столбцов по умолчанию - "Column1", "Column2" и т. д.

Пример №-1

Описание

Понижение уровня первой строки значений в таблице.

Код

```
Table.DemoteHeaders(Table.FromRecords({[CustomerID=1, Name="Bob", Phone="123-4567"],[CustomerID=2, Name="Jim", Phone="987-6543"]}))
```

Результат

```
Table.FromRecords({[Column1="CustomerID", Column2="Name", Column3="Phone"],[Column1=1, Column2="Bob", Column3="123-4567"],[Column1=2, Column2="Jim", Column3="987-6543"]})
```

Table.Distinct

Table.Distinct(table as table, optional equationCriteria as any) as table

Описание

Удаляет повторяющиеся строки из таблицы `table`. Необязательный параметр `equationCriteria` определяет, какие столбцы таблицы проверяются на наличие повторения. Если `equationCriteria` не задан, проверяются все столбцы.

Пример №-1

Описание

Удаление повторяющихся строк из таблицы.

Код

```
Table.Distinct(Table.FromRecords({[a = "A", b = "a"], [a = "B", b = "b"], [a = "A", b = "a"]}))
```

Результат

```
Table.FromRecords({[a = "A", b = "a"],
    [a = "B", b = "b"]}, {
    "a",
    "b"
})
```

Пример №-2

Описание

Удалить повторяющиеся строки из столбца [b] в таблице `({[a = "A", b = "a"], [a = "B", b = "a"], [a = "A", b = "b"]})`.

Код

```
Table.Distinct(Table.FromRecords({[a = "A", b = "a"], [a = "B", b = "a"], [a = "A", b = "b"]}), "b")
```

Результат

```
Table.FromRecords({[a = "A", b = "a"],
    [a = "A", b = "b"]}, {
    "a",
    "b"
})
```

Table.DuplicateColumn

Table.DuplicateColumn(table as table, columnName as text, newColumnName as text, optional columnType as type) as table

Описание

Повторять столбец с именем `columnName` в таблице `table`. Значения и тип данных для столбца `newColumnName` копируются из столбца `columnName`.

Пример №-1

Описание

Повторять столбец "a" в столбец с именем "copied column" в таблице `({[a = 1, b = 2], [a = 3, b = 4]})`.

Код

```
Table.DuplicateColumn(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), "a", "copied column")
```

Результат

```
Table.FromRecords({ [
        a = 1,
        b = 2,
        #"copied column" = 1
    ], [
        a = 3,
        b = 4,
        #"copied column" = 3
    ]
}, {
    "a",
    "b",
    "copied column"
})
```

Table.ExpandListColumn

Table.ExpandListColumn(table as table, column as text) as table

Описание

Получив `table`, где `column` - список значений, разделить список построчно для каждого значения. Значения в других столбцах повторяются в каждой вновь созданной строке.

Пример №-1

Описание

Разделение столбца списка [Name] в таблице.

Код

```
Table.ExpandListColumn(Table.FromRecords({[Name= {"Bob", "Jim", "Paul"}, Discount = .15]}), "Name")
```

Результат

```
Table.FromRecords({[Name="Bob", Discount=0.15],[Name="Jim", Discount=0.15],[Name="Paul", Discount=0.15]})
```

Table.ExpandRecordColumn

Table.ExpandRecordColumn(table as table, column as text, fieldNames as list, optional newColumnNames as list) as table

Описание

Получив `column` записей на входе `table`, создать таблицу со столбцом для каждого поля в записи. Можно также задать необязательный параметр `newColumnNames`, чтобы обеспечить уникальность имен столбцов в новой таблице.

- `table`: исходная таблица со столбцом записи, который требуется развернуть.
- `column`: столбец, который необходимо развернуть.
- `fieldNames`: список полей, которые требуется развернуть в столбцы таблицы.
- `newColumnNames`: список имен для новых столбцов. Новые имена столбцов не могут повторять какие-либо столбцы в новой таблице.

Пример №-1

Описание

Развернуть столбец [a] в таблице `({[a = [aa = 1, bb = 2, cc = 3], b = 2]})` ***в 3 столбца - "aa", "bb" и "cc".***

Код

```
Table.ExpandRecordColumn(Table.FromRecords({[a = [aa = 1, bb = 2, cc = 3], b = 2]}),
"a", {"aa", "bb", "cc"})
```

Результат

```
Table.FromRecords({[aa = 1,
       bb = 2,
       cc = 3,
       b = 2]}, {
    "aa",
    "bb",
    "cc",
    "b"
})
```

Table.ExpandTableColumn

Table.ExpandTableColumn(table as table, column as text, columnNames as list, optional newColumnNames as list) as table

Описание

Развертывает таблицы в `table`[`column`] в несколько строк и столбцов. `columnNames` используется для выбора столбцов для развертывания из внутренней таблицы. Укажите `newColumnNames`, чтобы избежать конфликтов между существующими и новыми столбцами.

Пример №-1

Описание

Развернуть столбцы таблицы в `[a]` в таблице `({[t = {[a=1, b=2, c=3], [a=2,b=4,c=6]}, b = 2]})` в 3 столбца - `[t.a]`, `[t.b]` и `[t.c]`.

Код

```
Table.ExpandTableColumn(Table.FromRecords({[t = Table.FromRecords({[a=1, b=2, c=
3],[a=2,b=4,c=6]}), b = 2]}), "t", {"a","b","c"}, {"t.a","t.b","t.c"})
```

Результат

```
Table.FromRecords({[t.a = 1, t.b = 2, t.c = 3, b = 2],
[t.a = 2, t.b = 4, t.c = 6, b = 2]}, {
    "t.a",
    "t.b",
    "t.c",
    "b"
})
```

Table.FillDown

Table.FillDown(table as table, columns as list) as table

Описание

Возвращает таблицу из указанного `table`, где значение предыдущей ячейки распространяется на ячейки со значением NULL ниже в указанном `columns`.

Пример №-1

Описание

Получение таблицы со значениями NULL в столбце [Place], заполненном значениями, расположенными над ними в таблице.

Код

```
Table.FillDown(Table.FromRecords({[Place=1, Name="Bob"],     [Place=null, Name="John"],
[Place=2,    Name="Brad"],    [Place=3, Name="Mark"], [Place=null, Name="Tom"],
[Place=null, Name="Adam"]}), {"Place"})
```

Результат

```
Table.FromRecords({[Place=1,Name="Bob"],[Place=1,Name="John"],[Place=2,Name="Brad"],[Pl
ace=3,Name="Mark"],[Place=3,Name="Tom"],[Place=3,Name="Adam"]})
```

Table.FillUp

Table.FillUp(table as table, columns as list) as table

Описание

Возвращает таблицу из указанного `table`, где значение следующей ячейки распространяется на ячейки со значением NULL выше в указанном `columns`.

Пример №-1

Описание

Получение таблицы со значениями NULL в столбце [Column2], заполненном значениями, расположенными под ними в таблице.

Код

```
Table.FillUp(Table.FromRecords({[Column1 = 1, Column2 = 2], [Column1  = 3, Column2 = null], [Column1  = 5, Column2 = 3]}), {"Column2"})
```

Результат

```
Table.FromRecords({[Column1=1,Column2=2],[Column1=3,Column2=3],[Column1=5,Column2=3]})
```

Table.FilterWithDataTable

Table.FilterWithDataTable(table as table, dataTableIdentifier as text) as any

Описание

Table.FilterWithDataTable

Table.FindText

Table.FindText(table as table, text as text) as table

Описание

Возвращает строки из таблицы `table` с текстом `text`. Если текст не найден, то возвращается пустая таблица.

Пример №-1

Описание

Нахождение в таблице строк, содержащих Bob.

Код

```
Table.FindText(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), "Bob")
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-4567"]})
```

Table.First

Table.First(table as table, optional default as any) as any

Описание

Возвращает первую строку из `table` или необязательное значение по умолчанию `default`, если таблица пуста.

Пример №-1

Описание

Найти первую строку таблицы `({})` или получить [a = 0, b = 0], если она пуста.

Код

```
Table.First(Table.FromRecords({}), [a = 0, b = 0])
```

Результат

```
[a = 0, b = 0]
```

Пример №-2

Описание

Нахождение первой строки таблицы.

Код

```
Table.First(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name = "Paul",
Phone = "543-7890"]}))
```

Результат

```
[CustomerID = 1, Name = "Bob", Phone = "123-4567"]
```

Table.FirstN

Table.FirstN(table as table, countOrCondition as any) as table

Описание

Возвращает первые строки таблицы `table` в зависимости от значения `countOrCondition`:

- Если `countOrCondition` является числом, то указывает, сколько строк возвращается (начиная с верхней).
- Если `countOrCondition` является условием, то строки возвращаются до тех пор, пока не будет достигнута строка, не удовлетворяющая условию.

Пример №-1

Описание

Нахождение первых двух строк таблицы.

Код

```
Table.FirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name = "Paul",
Phone = "543-7890"]}), 2)
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-
4567"],[CustomerID=2,Name="Jim",Phone="987-6543"]})
```

Пример №-2

Описание

Нахождение первых строк, в которых [a] > 0, в таблице.

Код

```
Table.FirstN(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4], [a = -5, b = -6]}),
each [a] > 0)
```

Результат

```
Table.FromRecords({ [a = 1, b = 2],
    [a = 3, b = 4] })
```

Table.FirstValue

Table.FirstValue(table as table, optional default as any) as any

Описание

Возвращает первый столбец первой строки таблицы `table` или указанное значение по умолчанию.

Table.FromColumns

Table.FromColumns(lists as list, optional columns as any) as table

Описание

Создает таблицу типа `columns` из списка `lists`, содержащего вложенные списки с именами столбцов и значениями. Если некоторые столбцы имеют больше значений, чем другие, отсутствующие значения заполняются значением по умолчанию NULL, если столбец допускает значения NULL.

Пример №-1

Описание

Создать таблицу из заданного списка столбцов и списка имен столбцов.

Код

```
Table.FromColumns({ {1, "Bob", "123-4567"} , {2, "Jim", "987-6543"}, {3, "Paul", "543-7890"}}, {"CustomerID", "Name", "Phone"})
```

Результат

```
Table.FromRecords({[CustomerID=1,Name=2,Phone=3],[CustomerID="Bob",Name="Jim",Phone="Paul"],[CustomerID="123-4567",Name="987-6543",Phone="543-7890"]})
```

Пример №-2

Описание

Получение таблицы из списка имен клиентов в списке. Каждое значение элемента списка клиентов становится значением строки, а каждый список становится столбцом.

Код

```
Table.FromColumns({ {1, "Bob", "123-4567"} , {2, "Jim", "987-6543"}, {3, "Paul", "543-7890"} })
```

Результат

```
Table.FromRecords({[Column1=1,Column2=2,Column3=3],[Column1="Bob",Column2="Jim",Column3="Paul"],[Column1="123-4567",Column2="987-6543",Column3="543-7890"]})
```

Пример №-3

Описание

Создание таблицы с другим числом столбцов на строку. Значением отсутствующей строки является Null.

Код

```
Table.FromColumns({ {1, 2, 3}, {4, 5}, {6, 7, 8, 9} }, {"column1", "column2", "column3"})
```

Результат

```
Table.FromRecords({[column1=1,column2=4,column3=6],[column1=2,column2=5,column3=7],[column1=3,column2=null,column3=8],[column1=null,column2=null,column3=9]})
```

Table.FromList

Table.FromList(list as list, optional splitter as function, optional columns as any, optional default as any, optional extraValues as number) as table

Описание

Преобразует список `list` в таблицу путем применения необязательной функции разбиения `splitter` к каждому элементу в списке. По умолчанию считается, что список является списком текстовых значений, разделенных запятыми. Необязательный параметр `columns` может быть числом столбцов, списком столбцов или TableType. Также можно указать необязательные параметры `default` и `extraValues`.

Пример №-1

Описание

Создание таблицы из списка с применением разделителя Record.FieldValues со столбцами CustomerID и Name в результирующей таблице.

Код

```
Table.FromList({[CustomerID=1,Name="Bob"],[CustomerID=2,Name="Jim"]} ,
Record.FieldValues, {"CustomerID", "Name"})
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob"],[CustomerID=2,Name="Jim"]})
```

Пример №-2

Описание

Создание таблицы из списка со столбцом, имеющим имя Letters, с применением разделителя по умолчанию.

Код

```
Table.FromList({"a", "b", "c", "d"}, null, {"Letters"})
```

Результат

```
Table.FromRecords({ [
        Letters = "a"
    ], [
        Letters = "b"
    ], [
        Letters = "c"
    ], [
        Letters = "d"
    ]
}, {
    "Letters"
})
```

Table.FromPartitions

Table.FromPartitions(partitionColumn as text, partitions as list, optional partitionColumnType as type) as table

Описание

Возвращает таблицу, являющуюся результатом комбинирования набора разделенных таблиц `partitions`. `partitionColumn` представляет собой имя добавляемого столбца. По умолчанию используется тип столбца `any`, однако он может быть указан в `partitionColumnType`.

Пример №-1

Описание

Найти тип элемента в списке `{number}`.

Код

```
Table.FromPartitions(
        "Year",
        {
            {
                1994,
                Table.FromPartitions(
                    "Month",
                    {
                        {
                            "Jan",
                            Table.FromPartitions(
                                "Day",
                                {
                                    {
                                        1,
                                        #table({"Foo"},{{"Bar"}})
                                    },
                                    {
                                        2,
                                        #table({"Foo"},{{"Bar"}})
                                    }
                                }
                            )
                        },
                        {
                            "Feb",
                            Table.FromPartitions(
                                "Day",
                                {
                                    {
                                        3,
                                        #table({"Foo"},{{"Bar"}})
                                    },
                                    {
                                        4,
                                        #table({"Foo"},{{"Bar"}})
                                    }
                                }
                            )
                        }
                    }
                )
            }
        })
```

Результат

```
Table.FromRecords({ [
        Foo = "Bar",
        Day = 1,
        Month = "Jan",
        Year = 1994
    ], [
        Foo = "Bar",
        Day = 2,
        Month = "Jan",
        Year = 1994
    ], [
        Foo = "Bar",
        Day = 3,
        Month = "Feb",
        Year = 1994
    ], [
        Foo = "Bar",
        Day = 4,
        Month = "Feb",
        Year = 1994
    ]
}, {
    "Foo",
    "Day",
    "Month",
    "Year"
})
```

Table.FromRecords

Table.FromRecords(records as list, optional columns as any) as table

Описание

Преобразует список записей `records` в таблицу.

Пример №-1

Описание

Создать таблицу из записей с помощью имен полей записей в качестве имен столбцов.

Код

```
Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"]})
```

Результат

```
Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"]})
```

Пример №-2

Описание

Создать таблицу из записей с введенными столбцами и выбрать числовые столбцы.

Код

```
Table.ColumnsOfType(Table.FromRecords({[CustomerID=1, Name="Bob"]}, type
table[CustomerID=Number.Type, Name=Text.Type]), {type number})
```

Результат

```
{"CustomerID"}
```

Table.FromRows

Table.FromRows(rows as list, optional columns as any) as table

Описание

Создает таблицу из списка "`rows`", где каждый элемент списка — внутренний список, содержащий значения столбцов для одной строки. Дополнительно для "`columns`" можно указать необязательный список имен столбцов, тип таблицы или число столбцов.

Пример №-1

Описание

Получение таблицы со столбцом [CustomerID] со значениями {1, 2}, столбцом [Name] со значениями {"Bob", "Jim"} и столбцом [Phone] со значениями {"123-4567", "987-6543"}, где [CustomerID] имеет числовой тип, а [Name] и [Phone] имеют текстовый тип.

Код

```
Table.FromRows({{1, "Bob", "123-4567"}, {2, "Jim", "987-6543"}}, type table [CustomerID
= number, Name = text, Phone = text])
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-
4567"],[CustomerID=2,Name="Jim",Phone="987-6543"]})
```

Пример №-2

Описание

Получение таблицы со столбцом [CustomerID] со значениями {1, 2}, столбцом [Name] со значениями {"Bob", "Jim"} и столбцом [Phone] со значениями {"123-4567", "987-6543"}.

Код

```
Table.FromRows({ {1, "Bob", "123-4567"},{2, "Jim", "987-6543"}},{"CustomerID", "Name",
"Phone"})
```

Результат

```
Table.FromRecords({[CustomerID=1, Name="Bob", Phone="123-4567"],[CustomerID=2,
Name="Jim", Phone="987-6543"]})
```

Table.FromValue

Table.FromValue(value as any, optional options as record) as table

Описание

Создает таблицу со столбцом, содержащим предоставленное значение или список значений `value`. Можно указать необязательный параметр записи `options`, чтобы контролировать следующее:

- `DefaultColumnName` : имя столбца, используемое при создании таблицы из списка или скалярного значения.

Пример №-1

Описание

Создание таблицы из списка.

Код

```
Table.FromValue({1, "Bob", "123-4567"})
```

Результат

```
Table.FromRecords({[ Value = 1 ], [Value = "Bob"], [Value="123-4567"]})
```

Пример №-2

Описание

Создать из значения 1 таблицу с настраиваемым именем столбца.

Код

```
Table.FromValue(1, [DefaultColumnName = "MyValue"])
```

Результат

```
Table.FromRecords({ [ MyValue = 1 ] })
```

Пример №-3

Описание

Создать таблицу из значения 1.

Код

```
Table.FromValue(1)
```

Результат

```
Table.FromRecords({ [ Value = 1 ] })
```

Table.Group

Table.Group(table as table, key as any, aggregatedColumns as list, optional groupKind as number, optional comparer as function) as table

Описание

Группирует строки в `table` по значениям в указанном столбце `key` для каждой строки. Для каждой группы создается запись, содержащая ключевые столбцы (и их значения) вместе со всеми агрегированными столбцами, заданными с помощью `aggregatedColumns`. Обратите внимание, что если функции сравнения соответствует несколько ключей, могут возвращаться разные ключи. Эта функция не гарантирует фиксированный порядок строк при возврате. Можно также указать `groupKind` и `comparer`.

Пример №-1

Описание

Группирование таблицы, путем добавления столбца групповой операции [total] с суммой цен (each List.Sum([price])).

Код

```
Table.Group(Table.FromRecords({[CustomerID= 1, price = 20], [CustomerID= 2, price =
10], [CustomerID= 2, price = 20], [CustomerID= 1, price = 10], [CustomerID= 3, price =
20], [CustomerID= 3, price = 5]}), "CustomerID", {"total",each List.Sum([price])})
```

Результат

```
Table.FromRecords({ [CustomerID= 1, total = 30], [CustomerID= 2, total = 30],
[CustomerID= 3, total = 25]},  {"CustomerID", "total"})
```

Table.HasColumns

Table.HasColumns(table as table, columns as any) as logical

Описание

указывает, содержит ли таблица `table` указанные столбцы `columns`. Возвращает `true`, если таблица содержит эти столбцы, `false` - в противном случае.

Пример №-1

Описание

Определение, содержит ли таблица столбец [Name].

Код

```
Table.HasColumns(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),"Name")
```

Результат

```
true
```

Пример №-2

Описание

Определение, содержит ли таблица столбцы [Name] и [PhoneNumber].

Код

```
Table.HasColumns(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),{"Name",
"PhoneNumber"})
```

Результат

```
false
```

Table.InsertRows

Table.InsertRows(table as table, offset as number, rows as list) as table

Описание

Возвращает таблицу со списком строк `rows`, вставленных в `table` в заданной позиции `offset`. Каждый столбец строки для вставки должен согласовываться по типу со столбцами таблицы.

Пример №-1

Описание

Вставка двух строк в таблицу в позиции 1.

Код

```
Table.InsertRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"]}), 1, {[CustomerID = 2, Name = "Jim", Phone = "987-6543"],[CustomerID = 3, Name
= "Paul", Phone = "543-7890"] })
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-
4567"],[CustomerID=2,Name="Jim",Phone="987-6543"],
[CustomerID=3,Name="Paul",Phone="543-7890"]})
```

Пример №-2

Описание

Вставка строки в таблицу в позиции 1.

Код

```
Table.InsertRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"]}), 1, {[CustomerID = 3, Name =
"Paul", Phone = "543-7890"]})
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-
4567"],[CustomerID=3,Name="Paul",Phone="543-7890"],[CustomerID=2,Name="Jim",Phone="987-
6543"]})
```

Table.IsDistinct

Table.IsDistinct(table as table, optional comparisonCriteria as any) as logical

Описание

Указывает, содержит ли "`table`" только уникальные строки (без повторений). Возвращает значение `true`, если строки уникальны, и `false` в противном случае. Необязательный параметр "`comparisonCriteria`" определяет, какие столбцы таблицы проверяются на наличие повторов. Если "`comparisonCriteria`" не задан, проверяются все столбцы.

Пример №-1

Описание

Определение, уникальны ли все строки таблицы в столбце.

Код

```
Table.IsDistinct(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-
4567"],  [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name =
"Paul", Phone = "543-7890"] ,  [CustomerID = 5, Name = "Bob", Phone = "232-1550"]}),
"Name")
```

Результат

```
false
```

Пример №-2

Описание

Определение, состоит ли таблица из уникальных строк.

Код

```
Table.IsDistinct(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-
4567"],  [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name =
"Paul", Phone = "543-7890"] , [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}))
```

Результат

```
true
```

Table.IsEmpty

Table.IsEmpty(table as table) as logical

Описание

Указывает, содержит ли "`table`" хотя бы одну строку. Возвращает значение `true`, если строк нет (т. е. таблица пуста), и `false` в противном случае.

Пример №-1

Описание

Определить, пуста ли таблица `({})`.

Код

```
Table.IsEmpty(Table.FromRecords({}))
```

Результат

```
true
```

Пример №-2

Описание

Определение, пуста ли таблица.

Код

```
Table.IsEmpty(Table.FromRecords({[CustomerID =1, Name ="Bob", Phone = "123-
4567"],[CustomerID =2, Name ="Jim", Phone = "987-6543"],[CustomerID =3, Name ="Paul",
Phone = "543-7890"]}))
```

Результат

```
false
```

Table.Join

Table.Join(table1 as table, key1 as any, table2 as table, key2 as any, optional joinKind as number, optional joinAlgorithm as number, optional keyEqualityComparers as list) as table

Описание

Соединяет строки таблицы `table1` со строками таблицы `table2` с учетом тождественности значений ключевых столбцов, выбранных в `key1` (для `table1`) и в `key2` (для `table2`).

По умолчанию выполняется внутреннее соединение, но можно включить необязательный параметр `joinKind` для указания типа соединения. Возможны следующие значения.

- `JoinKind.Inner`
- `JoinKind.LeftOuter`
- `JoinKind.RightOuter`
- `JoinKind.FullOuter`
- `JoinKind.LeftAnti`
- `JoinKind.RightAnti`

Необязательный набор `keyEqualityComparers` может быть включен для указания способа сравнения ключевых столбцов.

Пример №-1

Описание

Внутреннее соединение двух таблиц по полю [CustomerID]

Код

```
Table.Join(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"],  [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
"CustomerID", Table.FromRecords({ [OrderID = 1, CustomerID = 1, Item = "Fishing rod",
Price = 100.0], [OrderID = 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0],
[OrderID = 3, CustomerID = 2, Item = "Fishing net", Price = 25.0], [OrderID = 4,
CustomerID = 3, Item = "Fish tazer", Price = 200.0], [OrderID = 5, CustomerID = 3, Item
= "Bandaids", Price = 2.0], [OrderID = 6, CustomerID = 1, Item = "Tackle box", Price =
20.0], [OrderID = 7, CustomerID = 5, Item = "Bait", Price = 3.25], [OrderID = 8,
CustomerID = 5, Item = "Fishing Rod", Price = 100.0], [OrderID = 9, CustomerID = 6,
Item = "Bait", Price = 3.25]}), "CustomerID")
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-4567",OrderID=1,Item="Fishing
rod",Price=100],[CustomerID=1,Name="Bob",Phone="123-4567",OrderID=2,Item="1 lb.
worms",Price=5],[CustomerID=2,Name="Jim",Phone="987-6543",OrderID=3,Item="Fishing
net",Price=25],[CustomerID=3,Name="Paul",Phone="543-7890",OrderID=4,Item="Fish
tazer",Price=200],[CustomerID=3,Name="Paul",Phone="543-
7890",OrderID=5,Item="Bandaids",Price=2],[CustomerID=1,Name="Bob",Phone="123-
4567",OrderID=6,Item="Tackle box",Price=20]})
```

Table.Keys

Table.Keys(table as table) as list

Описание

Table.Keys

Table.Last

Table.Last(table as table, optional default as any) as any

Описание

Возвращает последнюю строку из `table` или необязательное значение по умолчанию `default`, если таблица пуста.

Пример №-1

Описание

Найти последнюю строку таблицы `({})` или возвратить [a = 0, b = 0], если она пуста.

Код

```
Table.Last(Table.FromRecords({}), [a = 0, b = 0])
```

Результат

```
[a = 0, b = 0]
```

Пример №-2

Описание

Нахождение последней строки таблицы.

Код

```
Table.Last(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"]}))
```

Результат

```
[CustomerID = 3, Name = "Paul", Phone = "543-7890"]
```

Table.LastN

Table.LastN(table as table, countOrCondition as any) as table

Описание

Возвращает последние строки таблицы `table` в зависимости от значения `countOrCondition`:

- Если `countOrCondition` является числом, то будет возвращено соответствующее количество строк, начиная с позиции (последняя - `countOrCondition`).
- Если `countOrCondition` является условием, то строки возвращаются в порядке возрастания до тех пор, пока не будет встречена строка, не соответствующая условию.

Пример №-1

Описание

Нахождение последних строк, в которых [a] > 0, в таблице.

Код

```
Table.LastN(Table.FromRecords({[a = -1, b = -2], [a = 3, b = 4], [a = 5, b = 6]}), each
_ [a] > 0)
```

Результат

```
Table.FromRecords({[a = 3, b = 4],
    [a = 5, b = 6]})
```

Пример №-2

Описание

Нахождение последних двух строк таблицы.

Код

```
Table.LastN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name = "Paul",
Phone = "543-7890"]}), 2)
```

Результат

```
Table.FromRecords({[CustomerID=2,Name="Jim",Phone="987-
6543"],[CustomerID=3,Name="Paul",Phone="543-7890"]})
```

Table.MatchesAllRows

Table.MatchesAllRows(table as table, condition as function) as logical

Описание

Указывает, все ли строки в `table` соответствуют заданному `condition`. Возвращает `true`, если все строки соответствуют условию, `false` - в противном случае.

Пример №-1

Описание

Определить, все ли значения строк равны [a = 1, b = 2], в таблице `({[a = 1, b = 2], [a = 3, b = 4]})`.

Код

```
Table.MatchesAllRows(Table.FromRecords({[a = 1, b = 2], [a = -3, b = 4]}), each _ = [a = 1, b = 2])
```

Результат

```
false
```

Пример №-2

Описание

Определение, все ли значения строк в столбце [a] в таблице являются четными.

Код

```
Table.MatchesAllRows(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8]}), each Number.Mod([a], 2) = 0 )
```

Результат

```
true
```

Table.MatchesAnyRows

Table.MatchesAnyRows(table as table, condition as function) as logical

Описание

Указывает, есть ли в `table` строка, соответствующая заданному `condition`. Возвращает `true`, если хотя бы одна строка соответствует условию, `false` - в противном случае.

Пример №-1

Описание

Определить, является ли хотя бы одно из значений строк в столбце [a] в таблице `({[a = 2, b = 4], [a = 6, b = 8]})` *четным.*

Код

```
Table.MatchesAnyRows(Table.FromRecords({[a = 1, b = 4], [a = 3, b = 8]}), each Number.Mod([a], 2) = 0 )
```

Результат

```
false
```

Пример №-2

Описание

Определить, имеется ли хотя бы одно значение строк [a = 1, b = 2] в таблице `({[a = 1, b = 2], [a = 3, b = 4]})`.

Код

```
Table.MatchesAnyRows(Table.FromRecords({[a = 1, b = 2], [a = -3, b = 4]}), each _ = [a = 1, b = 2])
```

Результат

```
true
```

Table.Max

Table.Max(table as table, comparisonCriteria as any, optional default as any) as any

Описание

Возвращает наибольшую строку в `table`, исходя из критериев `comparisonCriteria`. Если таблица пуста, то возвращается значение необязательного параметра `default`.

Пример №-1

Описание

Найти строку с наибольшим значением в столбце [a] в таблице `({[a = 2, b = 4], [a = 6, b = 8]})`.

Код

```
Table.Max(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8]}), "a")
```

Результат

```
[a = 6, b = 8]
```

Пример №-2

Описание

Найти строку с наибольшим значением в столбце [a] в таблице `({})`. *Получить -1, если таблица пуста.*

Код

```
Table.Max(#table({"a"},{}), "a", -1)
```

Результат

```
-1
```

Table.MaxN

Table.MaxN(table as table, comparisonCriteria as any, countOrCondition as any) as table

Описание

Возвращает наибольшую строку или строки в `table`, исходя из критериев `comparisonCriteria`. Для дальнейшей фильтрации строк после сортировки следует указать параметр `countOrCondition`. Обратите внимание, что алгоритм сортировки не гарантирует фиксированного порядка сортировки в результате. Параметр `countOrCondition` может принимать несколько форм:

- Если указано число, возвращается список до `countOrCondition` элементов по возрастанию.
- Если указано условие, возвращается список элементов, которые изначально соответствуют условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.

Пример №-1

Описание

Нахождение строки с наибольшим значением в столбце [a] с условием [b] > 0 в таблице. Помните, что строки сортируются до применения фильтра.

Код

```
Table.MaxN(Table.FromRecords({[a = 2, b = 4], [a = 8, b = 0], [a = 6, b = 2]}), "a",
each [b] > 0)
```

Результат

```
Table.FromRecords({})
```

Пример №-2

Описание

Нахождение строки с наибольшим значением в столбце [a] с условием [a] > 0 в таблице. Помните, что строки сортируются до применения фильтра.

Код

```
Table.MaxN(Table.FromRecords({[a = 2, b = 4], [a = 0, b = 0], [a = 6, b = 2]}), "a",
each [a] > 0)
```

Результат

```
Table.FromRecords({[a = 6, b = 2],
    [a = 2, b = 4]})
```

Table.Min

Table.Min(table as table, comparisonCriteria as any, optional default as any) as any

Описание

Возвращает наименьшую строку в `table`, исходя из критериев `comparisonCriteria`. Если таблица пуста, то возвращается значение необязательного параметра `default`.

Пример №-1

Описание

Нахождение строки с наименьшим значением в столбце [a] в таблице. Возврат -1, если таблица пуста.

Код

```
Table.Min(#table({"a"},{}), "a", -1)
```

Результат

```
-1
```

Пример №-2

Описание

Нахождение строки с наименьшим значением в столбце [a] в таблице.

Код

```
Table.Min(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8]}), "a")
```

Результат

```
[a = 2, b = 4]
```

Table.MinN

Table.MinN(table as table, comparisonCriteria as any, countOrCondition as any) as table

Описание

Возвращает наименьшие строки в `table`, исходя из критериев `comparisonCriteria`. Для дальнейшей фильтрации строк после сортировки следует указать параметр `countOrCondition`. Обратите внимание, что алгоритм сортировки не гарантирует фиксированного порядка сортировки в результате. Параметр `countOrCondition` может принимать несколько форм:

- Если указано число, возвращается список до `countOrCondition` элементов по возрастанию.
- Если указано условие, возвращается список элементов, которые изначально соответствуют условию. Как только обнаруживается элемент, не соответствующий условию, последующие элементы не рассматриваются.

Пример №-1

Описание

Нахождение строки с наименьшим значением в столбце [a] с условием [b] < 0 в таблице. Помните, что строки сортируются до применения фильтра.

Код

```
Table.MinN(Table.FromRecords({[a = 2, b = 4], [a = 8, b = 0], [a = 6, b = 2]}), "a",
each [b] < 0)
```

Результат

```
Table.FromRecords({})
```

Пример №-2

Описание

Нахождение строки с наименьшим значением в столбце [a] с условием [a] < 3 в таблице. Помните, что строки сортируются до применения фильтра.

Код

```
Table.MinN(Table.FromRecords({[a = 2, b = 4], [a = 0, b = 0], [a = 6, b = 4]}), "a",
each [a] < 3)
```

Результат

```
Table.FromRecords({[a = 0, b = 0],
    [a = 2, b = 4]})
```

Table.NestedJoin

Table.NestedJoin(table1 as table, key1 as any, table2 as any, key2 as any, newColumnName as text, optional joinKind as number, optional keyEqualityComparers as list) as table

Описание

Соединяет строки таблицы `table1` со строками таблицы `table2` с учетом тождественности значений ключевых столбцов, выбранных в `key1` (для `table1`) и в `key2` (для `table2`). Результаты записываются в столбец с именем `newColumnName`.

Необязательное значение `joinKind` указывает вид выполняемого соединения. По умолчанию выполняется левое внешнее соединение, если `joinKind` не указано.

Необязательный набор `keyEqualityComparers` может быть включен для указания способа сравнения ключевых столбцов.

Table.Partition

Table.Partition(table as table, column as text, groups as number, hash as function) as list

Описание

Секционирует `table` в список из `groups` таблиц на основании значения `column` и функции `hash`. Функция `hash` применяется к значению строки `column` для получения хэш-значения строки. Остаток от целочисленного деления хэш-значения `groups` определяет, в какую из возвращенных таблиц будет помещена строка.

- `table`: таблица для секционирования.
- `column`: столбец для хэширования, позволяющий определить, в какой из возвращенных таблиц находится строка.
- `groups`: количество таблиц, на которое будет разделена исходная таблица.
- `hash`: функция, применяемая для получения хэш-значения.

Пример №-1

Описание

Секционировать таблицу `({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]})` в две таблицы по столбцу [a], используя значения столбцов в качестве хэш-функции.

Код

```
Table.Partition(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), "a", 2, each _)
```

Результат

```
{ Table.FromRecords({[a = 2, b = 4], [a = 2, b = 4]}, {
    "a",
    "b"
}),
    Table.FromRecords({[a = 1, b = 4], [a = 1, b = 4]}, {
    "a",
    "b"
}) }
```

Table.PartitionValues

Table.PartitionValues(table as table) as table

Описание

Возвращает сведения о том, как секционирована таблица. Возвращается таблица, в которой каждый столбец является столбцом секций исходной таблицы, а каждая строка соответствует секции исходной таблицы.

Table.Pivot

Table.Pivot(table as table, pivotValues as list, attributeColumn as text, valueColumn as text, optional aggregationFunction as function) as table

Описание

Принимая пару столбцов, представляющих пары "атрибут-значение", переставляет данные из столбца атрибутов в заголовки столбцов.

Пример №-1

Описание

Получить значения "a", "b" и "c" в столбце атрибутов таблицы `({ [ key = "x", attribute = "a", value = 1 ], [ key = "x", attribute = "c", value = 3 ], [ key = "y", attribute = "a", value = 2 ], [ key = "y", attribute = "b", value = 4 ] })` *и свести их в их собственный столбец.*

Код

```
Table.Pivot(Table.FromRecords({ [ key = "x", attribute = "a", value = 1 ], [ key = "x", attribute = "c", value = 3 ], [ key = "y", attribute = "a", value = 2 ], [ key = "y", attribute = "b", value = 4 ] }), { "a", "b", "c" }, "attribute", "value")
```

Результат

```
Table.FromRecords({[ key = "x", a = 1, b = null, c = 3 ], [ key = "y", a = 2, b = 4, c = null ]})
```

Пример №-2

Описание

Получить значения "a", "b" и "c" в столбце атрибутов таблицы `({ [ key = "x", attribute = "a", value = 1 ], [ key = "x", attribute = "c", value = 3 ], [ key = "x", attribute = "c", value = 5 ], [ key = "y", attribute = "a", value = 2 ], [ key = "y", attribute = "b", value = 4 ] })` *и свести их в их собственный столбец. Атрибут "c" для ключа "x" имеет несколько связанных с ним значений, поэтому используйте функцию List.Max для разрешения конфликта.*

Код

```
Table.Pivot(Table.FromRecords({ [ key = "x", attribute = "a", value = 1 ], [ key = "x", attribute = "c", value = 3 ], [ key = "x", attribute = "c", value = 5 ], [ key = "y", attribute = "a", value = 2 ], [ key = "y", attribute = "b", value = 4 ] }), { "a", "b", "c" }, "attribute", "value", List.Max)
```

Результат

```
Table.FromRecords({[ key = "x", a = 1, b = null, c = 5 ], [ key = "y", a = 2, b = 4, c = null ]})
```

Table.PositionOf

Table.PositionOf(table as table, row as record, optional occurrence as any, optional equationCriteria as any) as any

Описание

Возвращает позицию строки первого вхождения `table` в указанной `row`. Возвращает значение -1, если вхождение не найдено.

- `table`: входная таблица.
- `row`: строка в таблице, позиция которой должна быть найдена.
- `occurrence`: *[необязательно]* указывает, какие вхождения строки возвращать.
- `equationCriteria`: *[необязательно]* управляет сравнением строк таблицы.

Пример №-1

Описание

Найти позицию второго вхождения [a = 2, b = 4] в таблице `({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]})`.

Код

```
Table.PositionOf(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), [a = 2, b = 4], 1)
```

Результат

```
2
```

Пример №-2

Описание

Найти позиции всех вхождений [a = 2, b = 4] в таблице `({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]})`.

Код

```
Table.PositionOf(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), [a = 2, b = 4], Occurrence.All)
```

Результат

```
{0, 2}
```

Пример №-3

Описание

Найти позицию первого вхождения [a = 2, b = 4] в таблице `({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]})`.

Код

```
Table.PositionOf(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), [a = 2, b = 4])
```

Результат

```
0
```

Table.PositionOfAny

Table.PositionOfAny(table as table, rows as list, optional occurrence as number, optional equationCriteria as any) as any

Описание

Возвращает позиции строк из `table` от первого вхождения в списке `rows`. Возвращает значение -1, если вхождение не найдено.

- `table`: входная таблица.
- `rows`: список строк в таблице для поиска позиций.
- `occurrence`: *[необязательно]* указывает, какие вхождения строки возвращать.
- `equationCriteria`: *[необязательно]* управляет сравнением строк таблицы.

Пример №-1

Описание

Найти позиции всех вхождений [a = 2, b = 4] или [a = 6, b = 8] в таблице `({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]}`.

Код

```
Table.PositionOfAny(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]}), {[a = 2, b = 4], [a = 6, b = 8]}, Occurrence.All)
```

Результат

```
{0, 1, 2}
```

Пример №-2

Описание

Найти позицию первого вхождения [a = 2, b = 4] или [a = 6, b = 8] в таблице `({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]})`.

Код

```
Table.PositionOfAny(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), {[a = 2, b = 4], [a = 6, b = 8]})
```

Результат

0

Table.PrefixColumns

Table.PrefixColumns(table as table, prefix as text) as table

Описание

Возвращает таблицу, где все имена столбцов из предоставленной "`table`" имеют в качестве префикса заданный текст (`prefix`), а также точку в формате `prefix.ColumnName`.

Пример №-1

Описание

Задание префикса MyTable для столбцов в таблице.

Код

```
Table.PrefixColumns(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"]}), "MyTable")
```

Результат

```
Table.FromRecords({[MyTable.CustomerID=1,MyTable.Name="Bob",
MyTable.Phone="123-4567"]})
```

Table.Profile

Table.Profile(table as table) as table

Описание

Возвращает профиль столбцов в `table`.

Для каждого столбца возвращаются следующие сведения (когда это возможно):

- минимум
- максимум
- среднее
- стандартное отклонение
- счетчик
- количество значений NULL
- число различных объектов

Table.PromoteHeaders

Table.PromoteHeaders(table as table, optional options as record) as table

Описание

Назначает первую строку значений в качестве новых заголовков столбцов (т. е. имен столбцов). По умолчанию заголовками назначаются только текстовые или числовые значения. Допустимые параметры:

`PromoteAllScalars`: если задано значение `true`, все скалярные значения в первой строке назначаются заголовками с использованием языка и региональных параметров (`Culture`), если они указаны (или текущего языкового стандарта документа). Для значений, которые невозможно преобразовать в текст, будет использоваться имя столбца по умолчанию.
`Culture`: имя языка и региональных параметров для данных.

Пример №-1

Описание

Повышение уровня первой строки значений в таблице.

Код

```
Table.PromoteHeaders(Table.FromRecords({[Column1 = "CustomerID", Column2 = "Name",
Column3 = #date(1980,1,1)], [Column1 = 1, Column2 = "Bob", Column3 =
#date(1980,1,1)]}))
```

Результат

```
Table.FromRecords({[CustomerID = 1,Name = "Bob", Column3 = #date(1980,1,1)]})
```

Пример №-2

Описание

Назначение всех скалярных значений в первой строке таблицы заголовками.

Код

```
Table.PromoteHeaders(Table.FromRecords({[Rank = 1, Name = "Name", Date =
#date(1980,1,1)],[Rank = 1, Name = "Bob", Date = #date(1980,1,1)]}), [PromoteAllScalars
= true, Culture = "en-US"])
```

Результат

```
Table.FromRecords({[1 = 1, Name = "Bob", #"1/1/1980" = #date(1980, 1, 1)]})
```

Table.Range

Table.Range(table as table, offset as number, optional count as number) as table

Описание

Возвращает строки из `table`, начиная с указанного `offset`. Необязательный параметр `count` указывает, сколько строк необходимо получить. По умолчанию возвращаются все строки после смещения.

Пример №-1

Описание

Получение всех строк в таблице, начиная со смещения 1.

Код

```
Table.Range(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 1)
```

Результат

```
Table.FromRecords({[CustomerID=2,Name="Jim",Phone="987-
6543"],[CustomerID=3,Name="Paul",Phone="543-7890"], [CustomerID = 4, Name = "Ringo",
Phone = "232-1550"]})
```

Пример №-2

Описание

Получение одной строки в таблице, начиная со смещения 1.

Код

```
Table.Range(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 1, 1)
```

Результат

```
Table.FromRecords({[CustomerID=2,Name="Jim",Phone="987-6543"]})
```

Table.RemoveColumns

Table.RemoveColumns(table as table, columns as any, optional missingField as number) as table

Описание

Удаляет указанный `columns` из предоставленной `table`. Если столбец не существует, возникает исключение, если необязательным параметром `missingField` не задана альтернатива (например, `MissingField.UseNull` или `MissingField.Ignore`).

Пример №-1

Описание

Удаление столбца [Address] из таблицы. Создает ошибку, если столбец не существует.

Код

```
Table.RemoveColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-
4567"]}), "Address")
```

Результат

```
[Expression.Error] The field 'Address' of the record was not found.
```

Пример №-2

Описание

Удаление столбца [Phone] из таблицы.

Код

```
Table.RemoveColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-
4567"]}), "Phone")
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob"]})
```

Table.RemoveFirstN

Table.RemoveFirstN(table as table, countOrCondition as any) as table

Описание

Возвращает таблицу, не содержащую первые `countOrCondition` строк из начала таблицы `table`. Количество удаленных строк зависит от необязательного параметра `countOrCondition`.

- Если `countOrCondition` не указано, удаляется только первая строка.
- Если `countOrCondition` - число, удаляется соответствующее количество строк (начиная с начала).
- Если `countOrCondition` - условие, будут удалены строки, соответствующие условию, до первой строки, не соответствующей условию.

Пример №-1

Описание

Удаление из таблицы первых строк, в которых [CustomerID] <= 2.

Код

```
Table.RemoveFirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"],  [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name =
"Paul", Phone = "543-7890"] ,  [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
each [CustomerID] <= 2)
```

Результат

```
Table.FromRecords({[CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID =
4, Name = "Ringo", Phone = "232-1550"]})
```

Пример №-2

Описание

Удаление первых двух строк таблицы.

Код

```
Table.RemoveFirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name =
"Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 2)
```

Результат

```
Table.FromRecords({[CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID =
4, Name = "Ringo", Phone = "232-1550"]})
```

Пример №-3

Описание

Удаление первой строки таблицы.

Код

```
Table.RemoveFirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name =
"Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 1)
```

Результат

```
Table.FromRecords({[CustomerID = 2, Name = "Jim", Phone = "987-6543"],  [CustomerID =
3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-
1550"]})
```

Table.RemoveLastN

Table.RemoveLastN(table as table, optional countOrCondition as any) as table

Описание

Возвращает таблицу, не содержащую последние `countOrCondition` строк из таблицы `table`. Количество удаленных строк зависит от необязательного параметра `countOrCondition`.

- Если `countOrCondition` не указано, удаляется только последняя строка.
- Если `countOrCondition` - число, удаляется соответствующее количество строк (начиная с конца).
- Если `countOrCondition` - условие, будут удалены строки, соответствующие условию, до первой строки, не соответствующей условию.

Пример №-1

Описание

Удаление последней строки таблицы.

Код

```
Table.RemoveLastN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"],[CustomerID = 2, Name = "Jim", Phone = "987-6543"],[CustomerID = 3, Name =
"Paul", Phone = "543-7890"],[CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 1)
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-
4567"],[CustomerID=2,Name="Jim",Phone="987-6543"],[CustomerID=3,Name="Paul",Phone="543-
7890"]})
```

Пример №-2

Описание

Удаление из таблицы последних строк, в которых [CustomerID] > 2.

Код

```
Table.RemoveLastN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"],[CustomerID = 2, Name = "Jim", Phone = "987-6543"],[CustomerID = 3, Name =
"Paul", Phone = "543-7890"],[CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}),
each [CustomerID] >= 2)
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob",Phone="123-4567"]})
```

Table.RemoveMatchingRows

Table.RemoveMatchingRows(table as table, rows as list, optional equationCriteria as any) as table

Описание

Удаляет все вхождения указанных `rows` из `table`. Для управления сравнением строк таблицы может быть указан необязательный параметр `equationCriteria`.

Пример №-1

Описание

Удалить все строки, где [a = 1], из таблицы `({[a = 1, b = 2], [a = 3, b = 4], [a = 1, b = 6]})`.

Код

```
Table.RemoveMatchingRows(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4], [a = 1, b = 6]}), {[a = 1]}, "a")
```

Результат

```
Table.FromRecords({[a = 3, b = 4]}, {
    "a",
    "b"
})
```

Table.RemoveRows

Table.RemoveRows(table as table, offset as number, optional count as number) as table

Описание

Удаляет `count` строк от начала `table`, начиная с указанной `offset`. Если параметр `count` не предоставлен, используется значение по умолчанию 1.

Пример №-1

Описание

Удаление двух строк из таблицы, начиная с позиции 1.

Код

```
Table.RemoveRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 1, 2)
```

Результат

```
Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]})
```

Пример №-2

Описание

Удаление из таблицы строки в позиции 1.

Код

```
Table.RemoveRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 1)
```

Результат

```
Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]})
```

Пример №-3

Описание

Удаление первой строки из таблицы.

Код

```
Table.RemoveRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul",
Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 0)
```

Результат

```
Table.FromRecords ({[CustomerID = 2, Name = "Jim", Phone = "987-6543"],  [CustomerID =
3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-
1550"]})
```

Table.RemoveRowsWithErrors

Table.RemoveRowsWithErrors(table as table, optional columns as list) as table

Описание

Возвращает таблицу со строками, удаленными из входной таблицы и содержащими ошибку по крайней мере в одной из ячеек. Если указан список столбцов, на наличие ошибок проверяются только ячейки в указанных столбцах.

Пример №-1

Описание

Удалить ошибочное значение из первой строки.

Код

```
Table.RemoveRowsWithErrors(Table.FromRecords({[Column1=...],[Column1=2], [Column1=3]}))
```

Результат

```
Table.FromRecords({[Column1=2], [Column1=3]})
```

Table.RenameColumns

Table.RenameColumns(table as table, renames as list, optional missingField as number) as table

Описание

Выполняет заданное переименовывание столбцов в таблице `table`. В операции замены `renames` применяется список пар значений - старое имя столбца и новое имя столбца. Если столбец не существует, возникает исключение, если необязательным параметром `missingField` не задана альтернатива (например, `MissingField.UseNull` или `MissingField.Ignore`).

Пример №-1

Описание

Замена имени столбца NewCol на NewColumn в таблице и пропуск замены, если столбца не существует.

Код

```
Table.RenameColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-
4567"]}), {"NewCol", "NewColumn"}, MissingField.Ignore)
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob", Phone="123-4567"]})
```

Пример №-2

Описание

Замена имени столбца CustomerNum на CustomerID, а имя PhoneNum на Phone в таблице.

Код

```
Table.RenameColumns(Table.FromRecords({[CustomerNum=1, Name="Bob", PhoneNum = "123-
4567"]}), {{"CustomerNum", "CustomerID"}, {"PhoneNum", "Phone"}})
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob", Phone="123-4567"]})
```

Пример №-3

Описание

Замена имени столбца CustomerNum на CustomerID в таблице.

Код

```
Table.RenameColumns(Table.FromRecords({[CustomerNum=1, Name="Bob", Phone = "123-
4567"]}), {"CustomerNum", "CustomerID"})
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob", Phone="123-4567"]})
```

Table.ReorderColumns

Table.ReorderColumns(table as table, columnOrder as list, optional missingField as number) as table

Описание

Возвращает таблицу из входных данных `table` со столбцами в порядке, указанном в `columnOrder`. Порядок столбцов, не включенных в список, не будет изменен. Для несуществующего столбца возникает исключение, если необязательный параметр `missingField` не указывает дополнительное действие (например, `MissingField.UseNull` или `MissingField.Ignore`).

Пример №-1

Описание

Смена местами столбцов [Phone] и [Name] в таблице.

Код

```
Table.ReorderColumns(Table.FromRecords({[CustomerID=1, Phone = "123-4567", Name
="Bob"]}), {"Name","Phone"})
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob", Phone="123-4567"]})
```

Пример №-2

Описание

Смена местами столбцов [Phone] и [Address] или использование MissingField.Ignore в таблице. Таблица не будет изменена, поскольку столбца [Address] не существует.

Код

```
Table.ReorderColumns(Table.FromRecords({[CustomerID=1, Name = "Bob", Phone = "123-
4567"]}), {"Phone", "Address"}, MissingField.Ignore)
```

Результат

```
Table.FromRecords({[CustomerID=1,Name="Bob", Phone="123-4567"]})
```

Table.Repeat

Table.Repeat(table as table, count as number) as table

Описание

Возвращает таблицу со строками из входа `table`, повторенными указанное число раз, `count`.

Пример №-1

Описание

Повтор строк в таблице дважды.

Код

```
Table.Repeat(Table.FromRecords({[a = 1, b = "hello"], [a = 3, b = "world"]}), 2)
```

Результат

```
Table.FromRecords({ [
        a = 1,
        b = "hello"
    ], [
        a = 3,
        b = "world"
    ], [
        a = 1,
        b = "hello"
    ], [
        a = 3,
        b = "world"
    ]
}, {
    "a",
    "b"
})
```

Table.ReplaceErrorValues

Table.ReplaceErrorValues(table as table, errorReplacement as list) as table

Описание

Заменяет значения ошибок в указанных столбцах `table` новыми значениями в списке `errorReplacement`. Формат списка имеет вид {{столбец1, значение1}, …}. Для каждого столбца может быть только одно замещающее значение; если указать столбец несколько раз, возникнет ошибка.

Пример №-1

Описание

Замена в таблице значения ошибки в столбце A текстом hello, а в столбце B — текстом world.

Код

```
Table.ReplaceErrorValues(Table.FromRows({{..., ...},{1,2}}, {"A","B"}), {{"A",
"hello"}, {"B", "world"}})
```

Результат

```
Table.FromRecords({[A = "hello", B = "world"],
    [A = 1, B = 2]}, {
    "A",
    "B"
})
```

Пример №-2

Описание

Замена в таблице значения ошибки текстом world.

Код

```
Table.ReplaceErrorValues(Table.FromRows({{1,"hello"},{3,...}}, {"A","B"}), {"B",
"world"})
```

Результат

```
Table.FromRecords({[A = 1, B = "hello"],
    [A = 3, B = "world"]}, {
    "A",
    "B"
})
```

Table.ReplaceKeys

Table.ReplaceKeys(table as table, keys as list) as table

Описание

Table.ReplaceKeys

Table.ReplaceMatchingRows

Table.ReplaceMatchingRows(table as table, replacements as list, optional equationCriteria as any) as table

Описание

Заменяет все указанные строки в `table` предоставленными строками. Строки, которые необходимо заменить, и сами замены определяются в `replacements` с использованием формата {старый, новый}. Для управления сравнением строк таблицы может быть указан необязательный параметр `equationCriteria`.

Пример №-1

Описание

Замена в таблице строк [a = 1, b = 2] и [a = 2, b = 3] на [a = -1, b = -2], [a = -2, b = -3].

Код

```
Table.ReplaceMatchingRows(Table.FromRecords({[a = 1, b =2], [a = 2, b = 3], [a = 3, b =
4], [a = 1, b = 2]}),{ {[a = 1, b = 2], [a = -1, b = -2]}, {[a = 2, b = 3], [a = -2, b
= -3]} })
```

Результат

```
Table.FromRecords({ [
        a = -1,
        b = -2
    ], [
```

```
        a = -2,
        b = -3
    ], [
        a = 3,
        b = 4
    ], [
        a = -1,
        b = -2
    ] }, {
    "a",
    "b"
})
```

Table.ReplaceRelationshipIdentity

Table.ReplaceRelationshipIdentity(value as any, identity as text) as any

Описание

Table.ReplaceRelationshipIdentity

Table.ReplaceRows

Table.ReplaceRows(table as table, offset as number, count as number, rows as list) as table

Описание

Заменяет указанное количество строк `count` во входных данных `table` заданными в `rows`, начиная с `offset`. Параметр `rows` представляет собой список записей.

- `table`: таблица, в которой выполняется замена.
- `offset`: количество строк, которые следует пропустить, прежде чем выполнять замену.
- `count`: количество строк для замены.
- `rows`: список записей строк, вставляемых в `table` в расположении, заданном `offset`.

Пример №-1

Описание

Замена 3 строк, начиная с позиции 1.

Код

```
Table.ReplaceRows(Table.FromRecords({[Column1=1], [Column1=2], [Column1=3],
[Column1=4], [Column1=5]}), 1, 3, {[Column1=6], [Column1=7]})
```

Результат

```
Table.FromRecords({[Column1=1],[Column1=6],[Column1=7],[Column1=5]})
```

Table.ReplaceValue

Table.ReplaceValue(table as table, oldValue as any, newValue as any, replacer as function, columnsToSearch as list) as table

Описание

Заменяет `oldValue` на `newValue` в указанных столбцах таблицы `table`.

Пример №-1

Описание

Замена в таблице текста ur текстом or.

Код

```
Table.ReplaceValue(Table.FromRecords({[a = 1, b = "hello"], [a = 3, b = "wurld"]}),
"ur", "or", Replacer.ReplaceText, {"b"})
```

Результат

```
Table.FromRecords({ [a = 1, b = "hello"],
    [a = 3, b = "world"]}, {
    "a",
    "b"
})
```

Пример №-2

Описание

Замена в таблице текста goodbye текстом world.

Код

```
Table.ReplaceValue(Table.FromRecords({[a = 1, b = "hello"], [a = 3, b = "goodbye"]}),
"goodbye", "world", Replacer.ReplaceText, {"b"})
```

Результат

```
Table.FromRecords({[a = 1, b = "hello"],
    [a = 3, b = "world"]}, {
    "a",
    "b"
})
```

Table.ReverseRows

Table.ReverseRows(table as table) as table

Описание

Возвращает таблицу со строками из входных данных `table` в обратном порядке.

Пример №-1

Описание

Изменение порядка строк в таблице на противоположный.

Код

```
Table.ReverseRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name =
"Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}))
```

Результат

```
Table.FromRecords({[CustomerID=4,Name="Ringo",Phone="232-
1550"],[CustomerID=3,Name="Paul",Phone="543-7890"],[CustomerID=2,Name="Jim",Phone="987-
6543"],[CustomerID=1,Name="Bob",Phone="123-4567"]})
```

Table.RowCount

Table.RowCount(table as table) as number

Описание

Возвращает количество строк в `table`.

Пример №-1

Описание

Нахождение числа строк в таблице.

Код

```
Table.RowCount(Table.FromRecords({[CustomerID =1, Name ="Bob", Phone = "123-
4567"],[CustomerID =2, Name ="Jim", Phone = "987-6543"],[CustomerID =3, Name ="Paul",
Phone = "543-7890"]}))
```

Результат

```
3
```

Table.Schema

Table.Schema(table as table) as table

Описание

Возвращает таблицу с описанием столбцов `table`.

Каждая строка в таблице описывает свойства столбца `table`:

Имя столбца	Описание
`Name`	Имя столбца.
`Position`	Отсчитываемое от 0 положение столбца в `table`.

`TypeName`	Имя типа столбца.
`Kind`	Вид типа столбца.
`IsNullable`	Определяет, может ли столбец содержать значения `null`.
`NumericPrecisionBase`	Основание системы счисления (например, основание 2, основание 10) полей `NumericPrecision` и `NumericScale`.
`NumericPrecision`	Точность числового столбца по основанию, указанному в `NumericPrecisionBase`. Это максимальное число цифр, которое может быть представлено значением данного типа (включая цифры дробной части).
`NumericScale`	Масштаб числового столбца по основанию, указанному в `NumericPrecisionBase`. Это число цифр в дробной части значения данного типа. Значение `0` указывает на фиксированный масштаб без цифр дробной части. Значение `null` указывает, что масштаб неизвестен (из-за того, что он является плавающим или не задан).
`DateTimePrecision`	Максимальное число цифр дробной части, поддерживаемое в той части значения даты и времени, которая отвечает за секунды.
`MaxLength`	Максимальное число знаков, разрешенное в столбце `text`, или максимальное число байтов, разрешенное в столбце `binary`.
`IsVariableLength`	Указывает, может ли столбец иметь переменную длину (вплоть до значения `MaxLength`) или его размер фиксирован.
`NativeTypeName`	Имя типа столбца в собственной системе типов источника (например, `nvarchar` для SQL Server).
`NativeDefaultExpression`	Выражение по умолчанию для значения этого столбца в собственном языке выражений источника (например, `42` или `newid()` для SQL Server).
`Description`	Описание столбца.

Table.SelectColumns

Table.SelectColumns(table as table, columns as any, optional missingField as number) as table

Описание

Возвращает `table` только с указанными `columns`.

- `table`: предоставленная таблица.
- `columns`: список столбцов из таблицы `table`, которые должны быть возвращены. Столбцы в возвращаемой таблице содержатся в порядке, указанном параметром `columns`.
- `missingField`: *(необязательно)* что делать, если столбец не существует. Пример: `MissingField.UseNull` или `MissingField.Ignore`.

Пример №-1

Описание

Если включенный столбец не существует, параметр `MissingField.UseNull` создаст столбец с значениями NULL.

Код

```
Table.SelectColumns(Table.FromRecords({[CustomerID=1, Name = "Bob", Phone = "123-4567"
]}), {"CustomerID", "NewColumn"}, MissingField.UseNull)
```

Результат

```
Table.FromRecords({[CustomerID=1, NewColumn=null]})
```

Пример №-2

Описание

Включить только столбец [Name].

Код

```
Table.SelectColumns(Table.FromRecords({
    [CustomerID = 1, Name = "Bob", Phone = "123-4567"],
    [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,
    [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,
    [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]
}), "Name")
```

Результат

```
Table.FromRecords({
    [Name = "Bob"],
    [Name = "Jim"] ,
    [Name = "Paul"] ,
    [Name = "Ringo"]
})
```

Пример №-3

Описание

Включить только столбцы [CustomerID] и [Name].

Код

```
Table.SelectColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-
4567"]}), {"CustomerID", "Name"})
```

Результат

```
Table.FromRecords({[CustomerID=1, Name="Bob"]})
```

Пример №-4

Описание

Если включенный столбец не существует, результат по умолчанию будет ошибкой.

Код

```
Table.SelectColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-
4567"]}), "NewColumn")
```

Результат

```
[Expression.Error] The field 'NewColumn' of the record wasn't found.
```

Table.SelectRows

Table.SelectRows(table as table, condition as function) as table

Описание

Возвращает таблицу строк из `table`, согласующуюся с выбором `condition`.

Пример №-1

Описание

Выбрать строки в таблице со значениями в столбце [CustomerID] больше 2.

Код

```
Table.SelectRows(Table.FromRecords({
    [CustomerID = 1, Name = "Bob", Phone = "123-4567"],
    [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,
    [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,
    [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]
}), each [CustomerID] > 2)
```

Результат

```
Table.FromRecords({
    [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,
    [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]
})
```

Пример №-2

Описание

Выбрать строки таблицы, в которых имена не содержат "B".

Код

```
Table.SelectRows(Table.FromRecords({
    [CustomerID = 1, Name = "Bob", Phone = "123-4567"],
    [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,
    [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,
    [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]
}), each not Text.Contains([Name], "B"))
```

Результат

```
Table.FromRecords({
    [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,
    [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,
    [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]
})
```

Table.SelectRowsWithErrors

Table.SelectRowsWithErrors(table as table, optional columns as list) as table

Описание

Возвращает таблицу, содержащую только те строки входной таблицы, которые содержат ошибку по крайней мере в одной из ячеек. Если указан список столбцов, на наличие ошибок проверяются только ячейки в указанных столбцах.

Пример №-1

Описание

Выбрать имена клиентов, в строках которых есть ошибки.

Код

```
Table.SelectRowsWithErrors(Table.FromRecords({
    [CustomerID =..., Name = "Bob", Phone = "123-4567"],
    [CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,
    [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,
    [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]
}))[Name]
```

Результат

```
{"Bob"}
```

Table.SingleRow

Table.SingleRow(table as table) as record

Описание

Возвращает единственную строку в одной строке `table`. Если `table` имеет более одной строки, возникает исключение.

Пример №-1

Описание

Возврат одной строки из таблицы.

Код

```
Table.SingleRow(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"]}))
```

Результат

```
[CustomerID = 1, Name = "Bob", Phone = "123-4567"]
```

Table.Skip

Table.Skip(table as table, countOrCondition as any) as table

Описание

Возвращает таблицу, не содержащую первые `countOrCondition` строк из начала таблицы `table`. Количество пропущенных строк зависит от необязательного параметра `countOrCondition`.

- Если `countOrCondition` не указано, пропускается только первая строка.
- Если `countOrCondition` - число, пропускается соответствующее количество строк (начиная с начала).
- Если `countOrCondition` - условие, будут пропускаться строки, соответствующие условию, до первой строки, не соответствующей условию.

Пример №-1

Описание

Пропуск первой строки таблицы.

Код

```
Table.Skip(Table.FromRecords({  [CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"] ,  [CustomerID = 3, Name = "Paul",
Phone = "543-7890"] ,  [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 1)
```

Результат

```
Table.FromRecords({[CustomerID=2,Name="Jim",Phone="987-
6543"],[CustomerID=3,Name="Paul",Phone="543-
7890"],[CustomerID=4,Name="Ringo",Phone="232-1550"]})
```

Пример №-2

Описание

Пропуск первых строк таблицы, в которых [Price] > 25.

Код

```
Table.Skip(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price
= 100.0], [OrderID = 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID =
3, CustomerID = 2, Item = "Fishing net", Price = 25.0], [OrderID = 4, CustomerID = 3,
```

```
Item = "Fish tazer", Price = 200.0], [OrderID = 5, CustomerID = 3, Item = "Bandaids",
Price = 2.0], [OrderID = 6, CustomerID = 1, Item = "Tackle box", Price = 20.0],
[OrderID = 7, CustomerID = 5, Item = "Bait", Price = 3.25], [OrderID = 8, CustomerID =
5, Item = "Fishing Rod", Price = 100.0], [OrderID = 9, CustomerID = 6, Item = "Bait",
Price = 3.25]}), each [Price] > 25)
```

Результат

```
Table.FromRecords({[OrderID=2,CustomerID=1,Item="1 lb.
worms",Price=5],[OrderID=3,CustomerID=2,Item="Fishing
net",Price=25],[OrderID=4,CustomerID=3,Item="Fish
tazer",Price=200],[OrderID=5,CustomerID=3,Item="Bandaids",Price=2],[OrderID=6,CustomerI
D=1,Item="Tackle
box",Price=20],[OrderID=7,CustomerID=5,Item="Bait",Price=3.25],[OrderID=8,CustomerID=5,
Item="Fishing Rod",Price=100],[OrderID=9,CustomerID=6,Item="Bait",Price=3.25]})
```

Пример №-3

Описание

Пропуск первых двух строк таблицы.

Код

```
Table.Skip(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-
4567"],[CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name =
"Paul", Phone = "543-7890"], [CustomerID = 4, Name = "Ringo", Phone = "232-1550"]}), 2)
```

Результат

```
Table.FromRecords({[CustomerID=3,Name="Paul",Phone="543-
7890"],[CustomerID=4,Name="Ringo",Phone="232-1550"]})
```

Table.Sort

Table.Sort(table as table, comparisonCriteria as any) as table

Описание

Сортирует `table`, используя список из одного или нескольких имен столбцов и необязательного параметра `comparisonCriteria` в форме { { col1, comparisonCriteria }, {col2} }.

Пример №-1

Описание

Сортировка таблицы по столбцу CustomerID, затем по столбцу OrderID, при этом по CustomerID — в порядке возрастания.

Код

```
Table.Sort(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price
= 100.0], [OrderID = 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID =
3, CustomerID = 2, Item = "Fishing net", Price = 25.0], [OrderID = 4, CustomerID = 3,
```

```
Item = "Fish tazer", Price = 200.0], [OrderID = 5, CustomerID = 3, Item = "Bandaids",
Price = 2.0], [OrderID = 6, CustomerID = 1, Item = "Tackle box", Price = 20.0],
[OrderID = 7, CustomerID = 5, Item = "Bait", Price = 3.25], [OrderID = 8, CustomerID =
5, Item = "Fishing Rod", Price = 100.0], [OrderID = 9, CustomerID = 6, Item = "Bait",
Price = 3.25]}), {{"CustomerID", Order.Ascending}, "OrderID"})
```

Результат

```
Table.FromRecords({[OrderID=1,CustomerID=1,Item="Fishing
rod",Price=100],[OrderID=2,CustomerID=1,Item="1 lb.
worms",Price=5],[OrderID=6,CustomerID=1,Item="Tackle
box",Price=20],[OrderID=3,CustomerID=2,Item="Fishing
net",Price=25],[OrderID=4,CustomerID=3,Item="Fish
tazer",Price=200],[OrderID=5,CustomerID=3,Item="Bandaids",Price=2],[OrderID=7,CustomerI
D=5,Item="Bait",Price=3.25],[OrderID=8,CustomerID=5,Item="Fishing
Rod",Price=100],[OrderID=9,CustomerID=6,Item="Bait",Price=3.25]})
```

Пример №-2

Описание

Сортировка таблицы по столбцу OrderID.

Код

```
Table.Sort(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price
= 100.0], [OrderID = 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID =
3, CustomerID = 2, Item = "Fishing net", Price = 25.0], [OrderID = 4, CustomerID = 3,
Item = "Fish tazer", Price = 200.0], [OrderID = 5, CustomerID = 3, Item = "Bandaids",
Price = 2.0], [OrderID = 6, CustomerID = 1, Item = "Tackle box", Price = 20.0],
[OrderID = 7, CustomerID = 5, Item = "Bait", Price = 3.25], [OrderID = 8, CustomerID =
5, Item = "Fishing Rod", Price = 100.0], [OrderID = 9, CustomerID = 6, Item = "Bait",
Price = 3.25]}), {"OrderID"})
```

Результат

```
Table.FromRecords({[OrderID=1,CustomerID=1,Item="Fishing
rod",Price=100],[OrderID=2,CustomerID=1,Item="1 lb.
worms",Price=5],[OrderID=3,CustomerID=2,Item="Fishing
net",Price=25],[OrderID=4,CustomerID=3,Item="Fish
tazer",Price=200],[OrderID=5,CustomerID=3,Item="Bandaids",Price=2],[OrderID=6,CustomerI
D=1,Item="Tackle
box",Price=20],[OrderID=7,CustomerID=5,Item="Bait",Price=3.25],[OrderID=8,CustomerID=5,
Item="Fishing Rod",Price=100],[OrderID=9,CustomerID=6,Item="Bait",Price=3.25]})
```

Пример №-3

Описание

Сортировка таблицы по столбцу OrderID в порядке убывания.

Код

```
Table.Sort(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price
= 100.0], [OrderID = 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID =
3, CustomerID = 2, Item = "Fishing net", Price = 25.0], [OrderID = 4, CustomerID = 3,
Item = "Fish tazer", Price = 200.0], [OrderID = 5, CustomerID = 3, Item = "Bandaids",
Price = 2.0], [OrderID = 6, CustomerID = 1, Item = "Tackle box", Price = 20.0],
```

```
[OrderID = 7, CustomerID = 5, Item = "Bait", Price = 3.25], [OrderID = 8, CustomerID =
5, Item = "Fishing Rod", Price = 100.0], [OrderID = 9, CustomerID = 6, Item = "Bait",
Price = 3.25]}), {"OrderID", Order.Descending})
```

Результат

```
Table.FromRecords({[OrderID=9,CustomerID=6,Item="Bait",Price=3.25],[OrderID=8,CustomerI
D=5,Item="Fishing
Rod",Price=100],[OrderID=7,CustomerID=5,Item="Bait",Price=3.25],[OrderID=6,CustomerID=1
,Item="Tackle
box",Price=20],[OrderID=5,CustomerID=3,Item="Bandaids",Price=2],[OrderID=4,CustomerID=3
,Item="Fish tazer",Price=200],[OrderID=3,CustomerID=2,Item="Fishing
net",Price=25],[OrderID=2,CustomerID=1,Item="1 lb.
worms",Price=5],[OrderID=1,CustomerID=1,Item="Fishing rod",Price=100]})
```

Table.SplitColumn

Table.SplitColumn(table as table, sourceColumn as text, splitter as function, optional columnNamesOrNumber as any, optional default as any, optional extraColumns as any) as table

Описание

Разбивает указанные столбцы в набор дополнительных столбцов с помощью заданной функции разделения.

Пример №-1

Описание

Разделить столбец [Name] на два столбца на позиции буквы "i"

Код

```
let
  Customers = Table.FromRecords({
        [CustomerID = 1, Name = "Bob", Phone = "123-4567"],
        [CustomerID = 2, Name = "Jim", Phone = "987-6543"],
        [CustomerID = 3, Name = "Paul", Phone = "543-7890"],
        [CustomerID = 4, Name = "Cristina", Phone = "232-1550"]
    })
in
    Table.SplitColumn(Customers,"Name",Splitter.SplitTextByDelimiter("i"),2)
```

Результат

```
Table.FromRecords({
    [CustomerID = 1, Name.1 = "Bob", Name.2 = null, Phone = "123-4567"],
    [CustomerID = 2, Name.1 = "J", Name.2 = "m", Phone = "987-6543"],
    [CustomerID = 3, Name.1 = "Paul", Name.2 = null, Phone = "543-7890"],
    [CustomerID = 4, Name.1 = "Cr", Name.2 = "st",  Phone = "232-1550"]
})
```

Table.ToColumns

Table.ToColumns(table as table) as list

Описание

Создает список вложенных списков из таблицы `table`. Каждый элемент списка является внутренним списком, содержащим значения столбца.

Пример №-1

Описание

Создание списка значений столбца из таблицы.

Код

```
Table.ToColumns(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],
[CustomerID = 2, Name = "Jim", Phone = "987-6543"] }))
```

Результат

```
{{1, 2},{"Bob",  "Jim"},{ "123-4567", "987-6543"}}
```

Table.ToList

Table.ToList(table as table, optional combiner as function) as list

Описание

Преобразует таблицу в список путем применения заданной функции объединения к каждой строке значений в таблице.

Пример №-1

Описание

Объединить текст каждой строки с запятой.

Код

```
Table.ToList(Table.FromRows({{Number.ToText(1),"Bob", "123-4567" }, {Number.ToText(2),
"Jim", "987-6543" }, {Number.ToText(3), "Paul", "543-7890" }}),
Combiner.CombineTextByDelimiter(","))
```

Результат

```
{"1,Bob,123-4567", "2,Jim,987-6543", "3,Paul,543-7890"}
```

Table.ToRecords

Table.ToRecords(table as table) as list

Описание

Преобразует таблицу `table` в список записей.

Пример №-1

Описание

Преобразовать таблицу в список записей.

Код

```
Table.ToRecords(Table.FromRows({{1, "Bob", "123-4567"} , {2, "Jim", "987-6543"}, {3, "Paul", "543-7890"} },{"CustomerID", "Name", "Phone"}))
```

Результат

```
{[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"]}
```

Table.ToRows

Table.ToRows(table as table) as list

Описание

Создает список вложенных списков из таблицы `table`. Каждый элемент списка является внутренним списком, содержащим значения строки.

Пример №-1

Описание

Создание списка значений строк из таблицы.

Код

```
Table.ToRows(Table.FromRecords({[CustomerID =1, Name ="Bob", Phone = "123-4567"],[CustomerID =2, Name ="Jim", Phone = "987-6543"],[CustomerID =3, Name ="Paul", Phone = "543-7890"]}))
```

Результат

```
{{1, "Bob", "123-4567"},{2, "Jim", "987-6543"},{3,  "Paul", "543-7890"}}
```

Table.TransformColumnNames

Table.TransformColumnNames(table as table, nameGenerator as function, optional options as record) as table

Описание

Преобразует имена столбцов с помощью предоставленной функции `nameGenerator`. Допустимые параметры:

`MaxLength`: максимальная длина имен новых столбцов. Если функция создает более длинное имя, оно будет усечено.
`Comparer` используется для управления сравнением при создании имен столбцов. Функции сравнения можно использовать для предоставления сравнений без учета регистра или с учетом языка, региональных параметров и языковых стандартов.
В языке формул доступны следующие встроенные функции сравнения:

- `Comparer.Ordinal`: используется для точного сравнения по порядковому номеру
- `Comparer.OrdinalIgnoreCase`: используется для точного сравнения по порядковому номеру без учета регистра
- `Comparer.FromCulture`: используется для сравнения с учетом языка и региональных параметров

Пример №-1

Описание

Удаление символа `#(tab)` из имен столбцов

Код

```
Table.TransformColumnNames(Table.FromRecords({[#"Col#(tab)umn" = 1]}), Text.Clean)
```

Результат

```
Table.FromRecords({[Column = 1]}, {"Column"})
```

Пример №-2

Описание

Преобразование имен столбцов для создания имен длиной 6 символов без учета регистра.

Код

```
Table.TransformColumnNames(Table.FromRecords({[ColumnNum = 1, cOlumnnum = 2, coLumnNUM
= 3]}), Text.Clean, [MaxLength = 6, Comparer = Comparer.OrdinalIgnoreCase])
```

Результат

```
Table.FromRecords({[Column = 1, cOlum1 = 2, coLum2 = 3]}, {"Column", "cOlum1",
"coLum2"})
```

Table.TransformColumnTypes

Table.TransformColumnTypes(table as table, typeTransformations as list, optional culture as text) as table

Описание

Возвращает таблицу из входных данных `table`, применяя операцию преобразования к столбцам, указанным в параметре `typeTransformations` (с форматом {имя столбца, имя типа}), используя язык и региональные параметры, указанные в параметре `culture`. Если столбец не существует, возникает исключение.

Пример №-1

Описание

Преобразовать числовые значения в столбце [a] в текстовые значения из таблицы `({[a = 1, b = 2], [a = 3, b = 4]})`.

Код

```
Table.TransformColumnTypes(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {"a", type text}, "en-US")
```

Результат

```
Table.FromRecords({[a = "1", b = 2],
    [a = "3", b = 4]})
```

Table.TransformColumns

Table.TransformColumns(table as table, transformOperations as list, optional defaultTransformation as function, optional missingField as number) as table

Описание

Возвращает таблицу из входных данных `table`, применяя операцию преобразования к столбцу, указанному в параметре `transformOperations` (в формате {имя столбца, преобразование}). Если столбец не существует, возникает исключение, если необязательным параметром `defaultTransformation` не задана альтернатива (например, `MissingField.UseNull` или `MissingField.Ignore`).

Пример №-1

Описание

Преобразовать числовые значения в отсутствующем столбце [X] в текстовые значения, пропуская несуществующие столбцы.

Код

```
Table.TransformColumns(Table.FromRecords({[A="1", B=2], [A="5", B=10]}), {"X", Number.FromText}, null, MissingField.Ignore)
```

Результат

```
Table.FromRecords({[A="1", B=2], [A="5", B=10]})
```

Пример №-2

Описание

Преобразовать числовые значения в отсутствующем столбце [X] в текстовые значения, возвращая ошибку для несуществующих столбцов.

Код

```
Table.TransformColumns(Table.FromRecords({[A="1",B=2], [A="5", B=10]}), {"X",
Number.FromText})
```

Результат

```
[Expression.Error] The column 'X' of the table wasn't found.
```

Пример №-3

Описание

Преобразовать числовые значения в отсутствующем столбце [X] в текстовые значения, по умолчанию присваивая значение NULL для несуществующих столбцов.

Код

```
Table.TransformColumns(Table.FromRecords({[A="1",B=2], [A="5", B=10]}), {"X",
Number.FromText}, null, MissingField.UseNull)
```

Результат

```
Table.FromRecords({[A="1", B=2, X=null], [A="5", B=10, X=null]})
```

Пример №-4

Описание

Преобразовать числовые значения в столбце [A] в числовые значения.

Код

```
Table.TransformColumns(Table.FromRecords({[A="1", B=2], [A="5", B=10]}),{"A",
Number.FromText})
```

Результат

```
Table.FromRecords({[A=1, B=2], [A=5, B=10]})
```

Table.TransformRows

Table.TransformRows(table as table, transform as function) as list

Описание

Создает таблицу из `table`, применяя операцию `transform` к строкам. Если указан возвращаемый тип функции `transform`, то результатом будет таблица со строками этого типа. Во всех остальных случаях результатом этой функции будет список с типом элементов согласно возвращаемому типу функции преобразования.

Пример №-1

Описание

Преобразовать строки в столбце [A] в текстовые значения в столбце [B] из таблицы `({[A = 1], [A = 2], [A = 3], [A = 4], [A = 5])`.

Код

```
Table.TransformRows(Table.FromRecords({[a = 1], [a = 2], [a = 3], [a = 4], [a = 5]}),
(row) as record => [B = Number.ToText(row[a])])
```

Результат

```
{ [
        B = "1"
   ], [
        B = "2"
   ], [
        B = "3"
   ], [
        B = "4"
   ], [
        B = "5"
   ] }
```

Пример №-2

Описание

Преобразовать строки в список чисел из таблицы `({[A = 1], [A = 2], [A = 3], [A = 4], [A = 5]})`.

Код

```
Table.TransformRows(Table.FromRecords({[a = 1], [a = 2], [a = 3], [a = 4], [a = 5]}),
each [a])
```

Результат

```
{1, 2, 3, 4, 5}
```

Table.Transpose

Table.Transpose(table as table, optional columns as any) as table

Описание

Превращает столбцы в строки, а строки в столбцы.

Пример №-1

Описание

Превратить строки таблицы пар "имя-значение" в столбцы.

Код

```
Table.Transpose(Table.FromRecords({[Name = "Full Name", Value = "Fred"], [Name = "Age",
Value = 42], [Name = "Country", Value = "UK"]}))
```

Результат

```
Table.FromRecords({ [
        Column1 = "Full Name",
        Column2 = "Age",
        Column3 = "Country"
    ], [
        Column1 = "Fred",
        Column2 = 42,
        Column3 = "UK"
    ]
}, {
    "Column1",
    "Column2",
    "Column3"
})
```

Table.Unpivot

Table.Unpivot(table as table, pivotColumns as list, attributeColumn as text, valueColumn as text) as table

Описание

Преобразует набор столбцов в таблице в пары "атрибут-значение" в сочетании с другими значениями в каждой строке.

Пример №-1

Описание

Взять столбцы "a", "b" и "c" в таблице `({[ key = "x", a = 1, b = null, c = 3 ], [ key = "y", a = 2, b = 4, c = null ]})` *и преобразовать их в пары "атрибут-значение".*

Код

```
Table.Unpivot(Table.FromRecords({[ key = "x", a = 1, b = null, c = 3 ], [ key = "y", a
= 2, b = 4, c = null ]}), { "a", "b", "c" }, "attribute", "value")
```

Результат

```
Table.FromRecords({ [ key = "x", attribute = "a", value = 1 ], [ key = "x", attribute =
"c", value = 3 ], [ key = "y", attribute = "a", value = 2 ], [ key = "y", attribute =
"b", value = 4 ] })
```

Table.UnpivotOtherColumns

Table.UnpivotOtherColumns(table as table, pivotColumns as list, attributeColumn as text, valueColumn as text) as table

Описание

Преобразует все столбцы, отличные от заданного набора, в пары "атрибут-значение", скомбинированные с остальными значениями в каждой строке.

Пример №-1

Описание

Преобразует все столбцы, отличные от заданного набора, в пары "атрибут-значение", скомбинированные с остальными значениями в каждой строке.

Код

```
Table.UnpivotOtherColumns(Table.FromRecords({
    [ key = "key1", attribute1 = 1, attribute2 = 2, attribute3 = 3 ],
    [ key = "key2", attribute1 = 4, attribute2 = 5, attribute3 = 6 ]
}), { "key" }, "column1", "column2")
```

Результат

```
Table.FromRecords({
    [ key = "key1", column1 = "attribute1", column2 = 1 ],
    [ key = "key1", column1 = "attribute2", column2 = 2 ],
    [ key = "key1", column1 = "attribute3", column2 = 3 ],
    [ key = "key2", column1 = "attribute1", column2 = 4 ],
    [ key = "key2", column1 = "attribute2", column2 = 5 ],
    [ key = "key2", column1 = "attribute3", column2 = 6 ]
})
```

Table.View

Table.View(table as table, handlers as record) as table

Описание

Возвращает представление `table`, в котором функции, указанные в `handlers`, используются вместо заданного по умолчанию порядка выполнения операции, когда операция применяется к представлению.

Функции обработчика являются необязательными. Если для операции не указана функция обработчика, к `table` применяется порядок выполнения операции, заданный по умолчанию (кроме случая `GetExpression`).

Функции обработчика должны возвращать значение, семантически эквивалентное результату применения операции к `table` (или результирующее представление в случае `GetExpression`).

Если функция обработчика создает ошибку, к представлению применяется порядок выполнения операции, заданный по умолчанию.

Для реализации свертывания к источнику данных можно использовать `Table.View` — преобразование M-запросов в запросы, соответствующие источнику (например, для создания инструкций T-SQL из M-запросов).

Более подробное описание `Table.View` см. в опубликованной документации.

Tables

Tables.GetRelationships

Tables.GetRelationships(tables as table, optional dataColumn as text) as table

Описание

Получает связи между набором таблиц. Предполагается, что набор "`tables`" должен иметь структуру, аналогичную таблице переходов. Столбец, указанный с помощью "`dataColumn`", содержит фактические таблицы данных.

Teradata

Teradata.Database

Teradata.Database(server as text, optional options as record) as table

Описание

Возвращает список таблиц и представлений SQL из базы данных Teradata на сервере `server`. Дополнительно к имени сервера через двоеточие может быть указан порт. Необязательный параметр записи `options` может быть указан для управления следующими параметрами.

- `CreateNavigationProperties` : Логическое значение (True или False), которое указывает, следует ли создавать свойства навигации в возвращаемых значениях. Значение по умолчанию — True.
- `NavigationPropertyNameGenerator` : Функция, которая используется для создания имен свойств навигации.
- `Query` : Собственный запрос SQL для извлечения данных. Если он создает несколько результирующих наборов, возвращается только первый из них.
- `CommandTimeout` : Длительность, контролирующая время выполнения запроса на стороне сервера до его отмены. Значение по умолчанию — десять минут.
- `ConnectionTimeout` : Длительность, контролирующая время ожидания до отмены попытки подключения к серверу. Значение по умолчанию зависит от драйвера.
- `HierarchicalNavigation` : Логическое значение (True или False), которое указывает, следует ли просматривать таблицы, сгруппированные по именам схем. Значение по умолчанию — False.

Пример параметра записи: [option1 = value1, option2 = value2...] или [Query = "select ..."].

Text

Text.AfterDelimiter

Text.AfterDelimiter(text as text, delimiter as text, optional index as any) as any

Описание

Возвращает часть `text` после указанного `delimiter`. Необязательный числовой `index` указывает, какое вхождение `delimiter` следует рассматривать. Необязательный список `index` указывает, какое вхождение `delimiter` следует рассматривать, а также откуда вести индексацию — с начала или с конца входных данных.

Пример №-1

Описание

Получить часть строки "111-222-333" после второго дефиса с конца.

Код

```
Text.AfterDelimiter("111-222-333", "-", {1, RelativePosition.FromEnd})
```

Результат

```
"222-333"
```

Пример №-2

Описание

Получить часть строки "111-222-333" после (первого) дефиса.

Код

```
Text.AfterDelimiter("111-222-333", "-")
```

Результат

```
"222-333"
```

Пример №-3

Описание

Получить часть строки "111-222-333" после второго дефиса.

Код

```
Text.AfterDelimiter("111-222-333", "-", 1)
```

Результат

```
"333"
```

Text.At

Text.At(text as text, index as number) as text

Описание

Возвращает символ в текстовом значении `text` в позиции `index`. Первый символ в тексте находится в позиции 0.

Пример №-1

Описание

Найти символ в позиции 4 в строке "Hello, World".

Код

```
Text.At("Hello, World", 4)
```

Результат

```
"o"
```

Text.BeforeDelimiter

Text.BeforeDelimiter(text as text, delimiter as text, optional index as any) as any

Описание

Возвращает часть `text` перед указанным `delimiter`. Необязательный числовой `index` указывает, какое вхождение `delimiter` следует рассматривать. Необязательный список `index` указывает, какое вхождение `delimiter` следует рассматривать, а также откуда вести индексацию — с начала или с конца входных данных.

Пример №-1

Описание

Получить часть строки "111-222-333" перед (первым) дефисом.

Код

```
Text.BeforeDelimiter("111-222-333", "-")
```

Результат

```
"111"
```

Пример №-2

Описание

Получить часть строки "111-222-333" перед вторым дефисом с конца.

Код

```
Text.BeforeDelimiter("111-222-333", "-", {1, RelativePosition.FromEnd})
```

Результат

```
"111"
```

Пример №-3

Описание

Получить часть строки "111-222-333" перед вторым дефисом.

Код

```
Text.BeforeDelimiter("111-222-333", "-", 1)
```

Результат

```
"111-222"
```

Text.BetweenDelimiters

Text.BetweenDelimiters(text as text, startDelimiter as text, endDelimiter as text, optional startIndex as any, optional endIndex as any) as any

Описание

Возвращает часть `text` между указанными `startDelimiter` и `endDelimiter`. Необязательный числовой `startIndex` указывает, какое вхождение `startDelimiter` следует рассматривать. Необязательный список `startIndex` указывает, какое вхождение `startDelimiter` следует рассматривать, а также откуда вести индексацию — с начала или с конца входных данных. Для `endIndex` все аналогично с тем исключением, что индексация ведется относительно `startIndex`.

Пример №-1

Описание

Получить часть строки "111 (222) 333 (444)" между второй открывающей скобкой с конца и второй закрывающей скобкой после нее.

Код

```
Text.BetweenDelimiters("111 (222) 333 (444)", "(", ")", {1, RelativePosition.FromEnd},
{1, RelativePosition.FromStart})
```

Результат

```
"222) 333 (444"
```

Пример №-2

Описание

Получить часть строки "111 (222) 333 (444)" между (первой) открывающей скобкой и (первой) закрывающей скобкой после нее.

Код

```
Text.BetweenDelimiters("111 (222) 333 (444)", "(", ")")
```

Результат

```
"222"
```

Пример №-3

Описание

Получить часть строки "111 (222) 333 (444)" между второй открывающей скобкой и первой закрывающей скобкой после нее.

Код

```
Text.BetweenDelimiters("111 (222) 333 (444)", "(", ")", 1, 0)
```

Результат

```
"444"
```

Text.Clean

Text.Clean(text as text) as text

Описание

Возвращает текстовое значение после удаления из `text` непечатаемых символов.

Пример №-1

Описание

Удалить перевод строки и другие непечатаемые символы из текстового значения.

Код

```
Text.Clean("ABC#(lf)D")
```

Результат

```
"ABCD"
```

Text.Combine

Text.Combine(texts as list, optional separator as text) as text

Описание

Возвращает результат объединения списка текстовых значений `texts` в одном текстовом значении. Можно указать дополнительный разделитель `separator`, используемый в итоговом объединенном тексте.

Пример №-1

Описание

Объединить текстовые значения "Seattle" и "WA".

Код

```
Text.Combine({"Seattle", "WA"})
```

Результат

```
"SeattleWA"
```

Пример №-2

Описание

Объединить текстовые значения "Seattle" и "WA", разделенные запятыми и пробелом, ", ".

Код

```
Text.Combine({"Seattle", "WA"}, ", ")
```

Результат

```
"Seattle, WA"
```

Text.Contains

Text.Contains(text as text, substring as text, optional comparer as function) as logical

Описание

Определяет, содержит ли текст `text` в себе текст `substring`. Возвращает значение true, если текст найден.

`comparer` — это модуль `Comparer`, который используется для управления сравнением. Функции сравнения можно использовать для сравнений, учитывающих или не учитывающих регистр, язык и региональные параметры, а также локаль.
В языке формул доступны следующие встроенные функции сравнения:

- `Comparer.Ordinal` — используется для точного сравнения по порядковому номеру
- `Comparer.OrdinalIgnoreCase` — используется для точного сравнения по порядковому номеру без учета регистра
- `Comparer.FromCulture` — используется для сравнения с учетом языка и региональных параметров

Пример №-1

Описание

Найти, содержит ли текст "Hello World" подстроку "hello".

Код

```
Text.Contains("Hello World", "hello")
```

Результат

```
false
```

Пример №-2

Описание

Найти, содержит ли текст "Hello World" подстроку "Hello".

Код

```
Text.Contains("Hello World", "Hello")
```

Результат

```
true
```

Text.End

Text.End(text as text, count as number) as text

Описание

Возвращает значение `text`, представляющее собой последние `count` символов значения `text` типа `text`.

Пример №-1

Описание

Получить последние 5 символов строки "Hello, World".

Код

```
Text.End("Hello, World", 5)
```

Результат

```
"World"
```

Text.EndsWith

Text.EndsWith(text as text, substring as text, optional comparer as function) as logical

Описание

Указывает, завершается ли данный текст `text` указанным значением `substring`. Проверка выполняется с учетом регистра.

`comparer` — это модуль `Comparer`, который используется для управления сравнением. Функции сравнения можно использовать для сравнений, учитывающих или не учитывающих регистр, язык и региональные параметры, а также локаль.
В языке формул доступны следующие встроенные функции сравнения:

- `Comparer.Ordinal` — используется для точного сравнения по порядковому номеру
- `Comparer.OrdinalIgnoreCase`: используется для точного сравнения по порядковому номеру без учета регистра
- `Comparer.FromCulture` — используется для сравнения с учетом языка и региональных параметров

Пример №-1

Описание

Проверить, оканчивается ли "Hello, World" на "World".

Код

```
Text.EndsWith("Hello, World", "World")
```

Результат

```
true
```

Пример №-2

Описание

Проверить, оканчивается ли "Hello, World" на "world".

Код

```
Text.EndsWith("Hello, World", "world")
```

Результат

```
false
```

Text.Format

Text.Format(formatString as text, arguments as any, optional culture as text) as text

Описание

Возвращает отформатированный текст, который создается путем применения `arguments` из списка или записи к строке форматирования `formatString`. Дополнительно можно указать язык и региональные параметры.

Пример №-1

Описание

Форматирование списка чисел.

Код

```
Text.Format("#{0}, #{1}, and #{2}.", { 17, 7, 22 })
```

Результат

```
"17, 7, and 22."
```

Пример №-2

Описание

Форматирование разных типов данных из записи в соответствии с языком и региональными параметрами.

Код

```
Text.Format("The time for the #[distance] km run held in #[city] on #[date] was
#[duration].", [city = "Seattle", date = #date(2015, 3, 10), duration =
#duration(0,0,54,40), distance = 10], "en-US")
```

Результат

```
"The time for the 10 km run held in Seattle on 3/10/2015 was 00:54:40."
```

Text.From

Text.From(value as any, optional culture as text) as text

Описание

Возвращает текстовое представление `value`. Значение `value` может иметь тип `number`, `date`, `time`, `datetime`, `datetimezone`, `logical`, `duration` или `binary`. Если данное значение — NULL, то `Text.From` возвращает значение NULL. Также может быть указан необязательный параметр `culture`.

Пример №-1

Описание

Создать текстовое значение из числа 3.

Код

```
Text.From(3)
```

Результат

```
"3"
```

Text.FromBinary

Text.FromBinary(binary as binary, optional encoding as number) as text

Описание

Декодирует данные `binary` из двоичного значения в текстовое, используя тип `encoding`.

Text.Insert

Text.Insert(text as text, offset as number, newText as text) as text

Описание

Возвращает результат вставки текстового значения `newText` в текстовое значение `text` в позиции `offset`. Позиции начинаются с номера 0.

Пример №-1

Описание

Вставить "C" между "B" и "D" в "ABD".

Код

```
Text.Insert("ABD", 2, "C")
```

Результат

```
"ABCD"
```

Text.Length

Text.Length(text as text) as number

Описание

Возвращает число символов в тексте `text`.

Пример №-1

Описание

Определить, сколько символов содержится в тексте "Hello World".

Код

```
Text.Length("Hello World")
```

Результат

```
11
```

Text.Lower

Text.Lower(text as text, optional culture as text) as text

Описание

Возвращает результат преобразования всех символов в `text` в нижний регистр.

Пример №-1

Описание

Получить версию строки "AbCd" в нижнем регистре.

Код

```
Text.Lower("AbCd")
```

Результат

```
"abcd"
```

Text.Middle

Text.Middle(text as text, start as number, optional count as number) as text

Описание

Возвращает символы (`count` шт.) или идет до конца "`text`"; при смещении `start`.

Пример №-1

Описание

Возвращает подстроку из текста Hello World, начиная с индекса 6. Длина области, в которой производится поиск, начинается с указанного индекса и распространяется на 5 символов.

Код

```
Text.Middle("Hello World", 6, 5)
```

Результат

```
"World"
```

Пример №-2

Описание

Отыскивает подстроку в тексте Hello World, начиная с индекса 6 до конца.

Код

```
Text.Middle("Hello World", 6, 20)
```

Результат

```
"World"
```

Text.NewGuid

Text.NewGuid() as text

Описание

Возвращает новый случайный глобальный уникальный идентификатор (GUID).

Text.PadEnd

Text.PadEnd(text as text, count as number, optional character as text) as text

Описание

Возвращает значение `text`, дополненное символами до длины `count` посредством вставки пробелов в конце текстового значения `text`. Для задания символа, который следует использовать для заполнения, можно указать необязательный символ `character`. Символом для заполнения по умолчанию является пробел.

Пример №-1

Описание

Заполнить конец текстового значения знаком "|" так, чтобы в значении было 10 символов.

Код

```
Text.PadEnd("Name", 10, "|")
```

Результат

```
"Name||||||"
```

Пример №-2

Описание

Заполнить конец текстового значения так, чтобы в значении было 10 символов.

Код

```
Text.PadEnd("Name", 10)
```

Результат

```
"Name      "
```

Text.PadStart

Text.PadStart(text as text, count as number, optional character as text) as text

Описание

Возвращает значение `text`, дополненное символами до длины `count` посредством вставки пробелов в начале текстового значения `text`. Для задания символа, который следует использовать для заполнения, можно указать необязательный символ `character`. Символом для заполнения по умолчанию является пробел.

Пример №-1

Описание

Заполнить начало текстового значения так, чтобы в значении было 10 символов.

Код

```
Text.PadStart("Name", 10)
```

Результат

```
"       Name"
```

Пример №-2

Описание

Заполнить начало текстового значения знаком "|" так, чтобы в значении было 10 символов.

Код

```
Text.PadStart("Name", 10, "|")
```

Результат

```
"||||||Name"
```

Text.PositionOf

Text.PositionOf(text as text, substring as text, optional occurrence as number, optional comparer as function) as any

Описание

Возвращает позицию указанного вхождения текстового значения `substring`, найденного в `text`. Для указания позиции возвращаемого вхождения может использоваться необязательный параметр `occurrence` (по умолчанию — первое вхождение).

`comparer` — это модуль `Comparer`, который используется для управления сравнением. Функции сравнения можно использовать для сравнений, учитывающих или не учитывающих регистр, язык и региональные параметры, а также языковой стандарт.
В языке формул доступны следующие встроенные функции сравнения:

- `Comparer.Ordinal` — используется для точного сравнения по порядковому номеру
- `Comparer.OrdinalIgnoreCase`: используется для точного сравнения по порядковому номеру без учета регистра
- `Comparer.FromCulture` — используется для сравнения с учетом языка и региональных параметров

Пример №-1

Описание

Возвращает позицию первого вхождения "World" в тексте "Hello, World! Hello, World!".

Код

```
Text.PositionOf("Hello, World! Hello, World!", "World")
```

Результат

```
7
```

Пример №-2

Описание

Получить позицию последнего вхождения "World" в "Hello, World! Hello, World!".

Код

```
Text.PositionOf("Hello, World! Hello, World!", "World", Occurrence.Last)
```

Результат

```
21
```

Text.PositionOfAny

Text.PositionOfAny(text as text, characters as list, optional occurrence as number) as any

Описание

Возвращает позицию первого вхождения любого из символов в списке символов `text`, обнаруженного в текстовом значении `characters`. Для указания позиции возвращаемого вхождения может использоваться необязательный параметр `occurrence`.

Пример №-1

Описание

Найти позицию "W" в тексте "Hello, World!".

Код

```
Text.PositionOfAny("Hello, World!", {"W"})
```

Результат

```
7
```

Пример №-2

Описание

Найти позицию "W" или "H" в тексте "Hello, World!".

Код

```
Text.PositionOfAny("Hello, World!", {"H","W"})
```

Результат

```
0
```

Text.Proper

Text.Proper(text as text, optional culture as text) as text

Описание

Возвращает результат перевода в верхний регистр только первой буквы каждого слова в текстовом значении `text`. Все остальные буквы возвращаются в нижнем регистре.

Пример №-1

Описание

Использовать `Text.Proper` для простого предложения.

Код

```
Text.Proper("the QUICK BrOWn fOx jUmPs oVER tHe LAzy DoG")
```

Результат

```
"The Quick Brown Fox Jumps Over The Lazy Dog"
```

Text.Range

Text.Range(text as text, offset as number, optional count as number) as text

Описание

Возвращает подстроку из текста "`text`", найденную по смещению "`offset`". Необязательный параметр "`count`" можно включить, чтобы указать число символов, которое необходимо получить. Возвращает ошибку при недостаточном количестве символов.

Пример №-1

Описание

Найдите подстроку из текста "Hello World", начиная с индекса 6.

Код

```
Text.Range("Hello World", 6)
```

Результат

```
"World"
```

Пример №-2

Описание

Найти подстроку из текста "Hello World Hello" длиной 5 символов, начиная с индекса 6.

Код

```
Text.Range("Hello World Hello", 6, 5)
```

Результат

```
"World"
```

Text.Remove

Text.Remove(text as text, removeChars as any) as text

Описание

Возвращает копию текстового значения `text`, в которой удалены все символы с `removeChars`.

Пример №-1

Описание

Удалить символы "," и ";" из текстового значения.

Код

```
Text.Remove("a,b;c",{",",";"})
```

Результат

```
"abc"
```

Text.RemoveRange

Text.RemoveRange(text as text, offset as number, optional count as number) as text

Описание

Возвращает копию текстового значения `text`, в которой удалены все символы с позиции `offset`. Для указания количества удаляемых символов можно использовать необязательный параметр `count`. Значение `count` по умолчанию - 1. Значения позиций начинаются с 0.

Пример №-1

Описание

Удалить 2 символа из текстового значения "ABEFC", начиная с позиции 2.

Код

```
Text.RemoveRange("ABEFC", 2, 2)
```

Результат

```
"ABC"
```

Пример №-2

Описание

Удалить 1 символ из текстового значения "ABEFC" в позиции 2.

Код

```
Text.RemoveRange("ABEFC", 2)
```

Результат

```
"ABFC"
```

Text.Repeat

Text.Repeat(text as text, count as number) as text

Описание

Возвращает текстовое значение, состоящее из входного текста `text`, повторенного `count` раз.

Пример №-1

Описание

Повторить текст "helloworld" три раза.

Код

```
Text.Repeat("helloworld.", 3)
```

Результат

```
"helloworld.helloworld.helloworld."
```

Пример №-2

Описание

Повторить текст "a" 5 раз.

Код

```
Text.Repeat("a", 5)
```

Результат

```
"aaaaa"
```

Text.Replace

Text.Replace(text as text, old as text, new as text) as text

Описание

Возвращает результат замены всех вхождений текстового значения `old` в текстовом значении `text` текстовым значением `new`. В этой функции учитывается регистр.

Пример №-1

Описание

Заменить все вхождения "the" в предложении на "a".

Код

```
Text.Replace("the quick brown fox jumps over the lazy dog", "the", "a")
```

Результат

```
"a quick brown fox jumps over a lazy dog"
```

Text.ReplaceRange

Text.ReplaceRange(text as text, offset as number, count as number, newText as text) as text

Описание

Возвращает результат удаления заданного количества символов `count` из текстового значения `text`, начиная с позиции `offset`, затем вставляет текстовое значение `newText` в ту же позицию в `text`.

Пример №-1

Описание

Заменить один символ в позиции 2 в текстовом значении "ABGF" новым текстовым значением "CDE".

Код

```
Text.ReplaceRange("ABGF", 2, 1, "CDE")
```

Результат

```
"ABCDEF"
```

Text.Split

Text.Split(text as text, separator as text) as list

Описание

Возвращает список текстовых значений, полученных в результате разбиения текстового значения `text` на основе указанного разделителя `separator`.

Пример №-1

Описание

Создать список из текстового значения "Name|Address|PhoneNumber", разбитого с помощью разделителя "|".

Код

```
Text.Split("Name|Address|PhoneNumber", "|")
```

Результат

```
{
    "Name",
    "Address",
    "PhoneNumber"
}
```

Text.SplitAny

Text.SplitAny(text as text, separators as text) as list

Описание

Возвращает список текстовых значений, полученных в результате разбиения текстового значения `text` на основе любого символа в указанном разделителе `separators`.

Пример №-1

Описание

Создать список из текстового значения "Jamie|Campbell|Admin|Adventure Works|www.adventure-works.com".

Код

```
Text.SplitAny("Jamie|Campbell|Admin|Adventure Works|www.adventure-works.com", "|")
```

Результат

```
{"Jamie",
    "Campbell",
    "Admin",
    "Adventure Works",
    "www.adventure-works.com"}
```

Text.Start

Text.Start(text as text, count as number) as text

Описание

Возвращает первые `count` символов из `text` как текстовое значение.

Пример №-1

Описание

Получить первые 5 символов "Hello, World".

Код

```
Text.Start("Hello, World", 5)
```

Результат

```
"Hello"
```

Text.StartsWith

Text.StartsWith(text as text, substring as text, optional comparer as function) as logical

Описание

Возвращает значение true, если текстовое значение `text` начинается с текстового значения `substring`.

- `text`: Значение `text`, которое следует найти

  .

- `substring`: Значение `text`, представляющее подстроку для поиска в `substring`
- `comparer`: *[необязательно]* `Comparer`, используемый для управления сравнением. Например, `Comparer.IgnoreCase` может использоваться для поиска без учета регистра

`comparer` — это модуль `Comparer`, который используется для управления сравнением. Функции сравнения можно использовать для сравнений, учитывающих или не учитывающих регистр, язык и региональные параметры, а также локаль.
В языке формул доступны следующие встроенные функции сравнения:

- `Comparer.Ordinal` — используется для точного сравнения по порядковому номеру
- `Comparer.OrdinalIgnoreCase` — используется для точного сравнения по порядковому номеру без учета регистра
- `Comparer.FromCulture` — используется для сравнения с учетом языка и региональных параметров

Пример №-1

Описание

Проверить, начинается ли текст "Hello, World" с "hello".

Код

```
Text.StartsWith("Hello, World", "hello")
```

Результат

```
false
```

Пример №-2

Описание

Проверить, начинается ли текст "Hello, World" с "Hello".

Код

```
Text.StartsWith("Hello, World", "Hello")
```

Результат

```
true
```

Text.ToBinary

Text.ToBinary(text as text, optional encoding as number, optional includeByteOrderMark as logical) as binary

Описание

Кодирует заданное текстовое значение `text` в двоичное значение с помощью указанного `encoding`.

Text.ToList

Text.ToList(text as text) as list

Описание

Возвращает список значений символов из заданного текстового значения `text`.

Пример №-1

Описание

Создать список значений символов из текста "Hello World".

Код

```
Text.ToList("Hello World")
```

Результат

```
{"H",
    "e",
    "l",
    "l",
    "o",
    " ",
    "W",
    "o",
    "r",
    "l",
    "d"}
```

Text.Trim

Text.Trim(text as text, optional trim as any) as text

Описание

Возвращает результат удаления всех начальных и конечных пробелов из текстового значения `text`.

Пример №-1

Описание

Удалить начальные и конечные пробелы из " a b c d ".

Код

```
Text.Trim("     a b c d    ")
```

Результат

```
"a b c d"
```

Text.TrimEnd

Text.TrimEnd(text as text, optional trim as any) as text

Описание

Возвращает результат удаления всех конечных пробелов из текстового значения `text`.

Пример №-1

Описание

Удалить конечные пробелы из " a b c d ".

Код

```
Text.TrimEnd("     a b c d    ")
```

Результат

```
"     a b c d"
```

Text.TrimStart

Text.TrimStart(text as text, optional trim as any) as text

Описание

Возвращает результат удаления всех начальных пробелов из текстового значения `text`.

Пример №-1

Описание

Удалить начальные пробелы из " a b c d ".

Код

```
Text.TrimStart("   a b c d    ")
```

Результат

```
"a b c d    "
```

Text.Upper

Text.Upper(text as text, optional culture as text) as text

Описание

Возвращает результат преобразования всех символов в `text` в верхний регистр.

Пример №-1

Описание

Получить версию строки "aBcD" в нижнем регистре.

Код

```
Text.Upper("aBcD")
```

Результат

```
"ABCD"
```

Time

Time.EndOfHour

Time.EndOfHour(dateTime as any) as any

Описание

Возвращает значение `time`, `datetime` или `datetimezone`, представляющее конец часа в `dateTime`, включая дробные секунды. Данные о часовом поясе сохраняются.

- `dateTime`: значение `time`, `datetime` или `datetimezone`, на основе которого вычисляется конец часа.

Пример №-1

Описание

Получить конец часа для 17.05.2011 17:00:00-7:00.

Код

```
Time.EndOfHour(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0))
```

Результат

```
#datetimezone(2011, 5, 17, 5, 59, 59.9999999, -7, 0)
```

Пример №-2

Описание

Получить конец часа для 14.05.2011 17:00:00.

Код

```
Time.EndOfHour(#datetime(2011, 5, 14, 17, 0, 0))
```

Результат

```
#datetime(2011, 5, 14, 17, 59, 59.9999999)
```

Time.From

Time.From(value as any, optional culture as text) as time

Описание

Возвращает значение `time` из указанного `value`. Если данное `value` имеет значение `null`, `Time.From` возвращает `null`. Если данное `value` - `time`, возвращается `value`. Значения следующих типов можно преобразовать в значение `time`:

- `text`: значение `time` из текстового представления. Дополнительные сведения см. в разделе `Time.FromText` .
- `datetime`: компонент времени из `value`.
- `datetimezone`: компонент времени локального эквивалента datetime `value`.
- `number`: значение `time`, эквивалентное числу неполных дней, выраженному `value`. Если `value` отрицательно или больше или равно 1, возвращается ошибка.

Если `value` имеет другой тип, возвращается ошибка.

Пример №-1

Описание

Преобразовать `#datetime(1899, 12, 30, 06, 45, 12)` в значение `time`.

Код

```
Time.From(#datetime(1899, 12, 30, 06, 45, 12))
```

Результат

```
#time(06, 45, 12)
```

Пример №-2

Описание

Преобразовать `0.7575` в значение `time`.

Код

```
Time.From(0.7575)
```

Результат

```
#time(18,10,48)
```

Time.FromText

Time.FromText(text as text, optional culture as text) as time

Описание

Создает значение `time` из текстового представления `text` согласно стандарту формата ISO 8601.

- `Time.FromText("12:34:12")` // Время, чч:мм:сс
- `Time.FromText("12:34:12.1254425")` // чч:мм:сс.нннннnн

Пример №-1

Описание

Преобразование `"10"` в значение Time.

Код

```
Time.FromText("10")
```

Результат

```
#time(10, 00, 00)
```

Пример №-2

Описание

Преобразовать `"10:12:31am"` в значение времени.

Код

```
Time.FromText("10:12:31am")
```

Результат

```
#time(10, 12, 31)
```

Пример №-3

Описание

Преобразование `"1012"` в значение Time.

Код

```
Time.FromText("1012")
```

Результат

```
#time(10, 12, 00)
```

Time.Hour

Time.Hour(dateTime as any) as number

Описание

Возвращает компонент часов заданного значения `time`, `datetime` или `datetimezone`, `dateTime`.

Пример №-1

Описание

Найти часы в #datetime(2011, 12, 31, 9, 15, 36).

Код

```
Time.Hour(#datetime(2011, 12, 31, 9, 15, 36))
```

Результат

```
9
```

Time.Minute

Time.Minute(dateTime as any) as number

Описание

Возвращает компонент минут заданного значения `time`, `datetime` или `datetimezone`, `dateTime`.

Пример №-1

Описание

Найти минуты в #datetime(2011, 12, 31, 9, 15, 36).

Код

```
Time.Minute(#datetime(2011, 12, 31, 9, 15, 36))
```

Результат

```
15
```

Time.Second

Time.Second(dateTime as any) as number

Описание

Возвращает компонент секунд заданного значения `time`, `datetime` или `datetimezone`, `dateTime`.

Пример №-1

Описание

Поиск второго значения в значении даты и времени.

Код

```
Time.Second(#datetime(2011, 12, 31, 9, 15, 36.5))
```

Результат

```
36.5
```

Time.StartOfHour

Time.StartOfHour(dateTime as any) as any

Описание

Возвращает первое значение часа для заданного типа `time`, `datetime` или `datetimezone`.

Пример №-1

Описание

Поиск начала часа для 10 октября 2011 г., 8:10:32 (`#datetime(2011, 10, 10, 8, 10, 32)`).

Код

```
Time.StartOfHour(#datetime(2011, 10, 10, 8, 10, 32))
```

Результат

```
#datetime(2011, 10, 10, 8, 0, 0)
```

Time.ToRecord

Time.ToRecord(time as time) as record

Описание

Возвращает запись, содержащую части заданного значения времени, `time`.

- `time`: значение `time`, для которого необходимо вычислить запись частей.

Пример №-1

Описание

Преобразовать значение `#time(11, 56, 2)` в запись, содержащую значения времени.

Код

```
Time.ToRecord(#time(11, 56, 2))
```

Результат

```
[
      Hour = 11,
      Minute = 56,
      Second = 2
]
```

Time.ToText

Time.ToText(time as time, optional format as text, optional culture as text) as text

Описание

Возвращает текстовое представление `time` значения времени `time`. Эта функция принимает необязательный параметр формата `format`. Полный список поддерживаемых форматов см. в спецификации библиотеки.

Пример №-1

Описание

Получение текстового представления #time(11, 56, 2) с возможностью форматирования.

Код

```
Time.ToText(#time(11, 56, 2), "hh:mm")
```

Результат

```
"11:56"
```

Пример №-2

Описание

Получение текстового представления #time(11, 56, 2).

Код

```
Time.ToText(#time(11, 56, 2))
```

Результат

```
"11:56 AM"
```

Type

Type.AddTableKey

Type.AddTableKey(table as type, columns as list, isPrimary as logical) as type

Описание

Добавляет ключ к данному типу таблицы.

Type.ClosedRecord

Type.ClosedRecord(type as type) as type

Описание

Возвращает закрытую версию данной записи `record type` (или тот же тип, если он уже закрыт).

Пример №-1

Описание

Создать закрытую версию `type [ A = number,…]`.

Код

```
Type.ClosedRecord(type [ A = number,...])
```

Результат

```
type [
    A = number
]
```

Type.Facets

Type.Facets(type as type) as record

Описание

Возвращает запись, содержащую аспекты `type`.

Type.ForFunction

Type.ForFunction(signature as record, min as number) as type

Описание

Создает тип функции `function type` из `signature`, запись `ReturnType` и `Parameters`, и `min` - минимальное число аргументов, необходимых для вызова функции.

Пример №-1

Описание

Определить, является ли запись `type [ A = число, ...]` *открытой.*

Код

```
Type.ForFunction([ReturnType = type number, Parameters = [X = type number]], 1)
```

Результат

```
type  function (X as number) as number
```

Type.ForRecord

Type.ForRecord(fields as record, open as logical) as type

Описание

Возвращает тип, представляющий записи с определенными ограничениями на тип для полей.

Type.FunctionParameters

Type.FunctionParameters(type as type) as record

Описание

Возвращает запись со значениями полей, в качестве которых указаны имена параметров `type`, значения которых отражают их соответствующие типы.

Пример №-1

Описание

Найти типы параметров функции `(x как число, y как текст)`.

Код

```
Type.FunctionParameters(type function (x as number, y as text) as any)
```

Результат

```
[x = type number, y = type text]
```

Type.FunctionRequiredParameters

Type.FunctionRequiredParameters(type as type) as number

Описание

Возвращает число, обозначающее минимальное количество параметров, необходимых для вызова входных данных `type` функции.

Пример №-1

Описание

Найти число необходимых параметров для функции `(x как число, y как текст (дополнительно))`.

Код

```
Type.FunctionRequiredParameters(type function (x as number, optional y as text) as any)
```

Результат

```
1
```

Type.FunctionReturn

Type.FunctionReturn(type as type) as type

Описание

Возвращает тип, возвращенный функцией `type`.

Пример №-1

Описание

Найти возвращаемый тип `() как любой)`.

Код

```
Type.FunctionReturn(type function () as any)
```

Результат

```
type any
```

Type.Is

Type.Is(type1 as type, type2 as type) as logical

Описание

Type.Is

Type.IsNullable

Type.IsNullable(type as type) as logical

Описание

Возвращает `true`, если тип является типом `nullable`; в противном случае, возвращается `false`.

Пример №-1

Описание

Определяет, является ли `number` допускающим значение "null".

Код

```
Type.IsNullable(type number)
```

Результат

```
false
```

Пример №-2

Описание

Определяет, является ли `type nullable number` допускающим значение "null".

Код

```
Type.IsNullable(type nullable number)
```

Результат

```
true
```

Type.IsOpenRecord

Type.IsOpenRecord(type as type) as logical

Описание

Возвращает значение `logical`, указывающее, является ли запись `type` открытой.

Пример №-1

Описание

Определить, является ли запись `type [ A = число, ...]` открытой.

Код

```
Type.IsOpenRecord(type [ A = number,...])
```

Результат

```
true
```

Type.ListItem

Type.ListItem(type as type) as type

Описание

Возвращает тип элемента из списка. `type`.

Пример №-1

Описание

Найти тип элемента в списке `{number}`.

Код

```
Type.ListItem(type {number})
```

Результат

```
type number
```

Type.NonNullable

Type.NonNullable(type as type) as type

Описание

Возвращает тип, не являющийся `nullable`, из `type`.

Пример №-1

Описание

Возврат типа, не допускающего значение "null", `type nullable number`.

Код

```
Type.NonNullable(type nullable number)
```

Результат

```
type number
```

Type.OpenRecord

Type.OpenRecord(type as type) as type

Описание

Возвращает открытую версию данной записи `record type` (или такой же тип, если запись уже является открытой).

Пример №-1

Описание

Создать открытую версию `type [ A = number]`.

Код

```
Type.OpenRecord(type [ A = number])
```

Результат

```
type [
    A = number, ...
]
```

Type.RecordFields

Type.RecordFields(type as type) as record

Описание

Возвращает запись, описывающую поля записи `type`. Каждое поле возвращенного типа записи имеет соответствующее имя и значение в виде записи `[ Type = type, Optional = logical ]`.

Пример №-1

Описание

Найти имя и значение записи `[ A = number, optional B = any]`.

Код

```
Type.RecordFields(type [ A = number, optional B = any])
```

Результат

```
[ A = [Type = type number, Optional = false], B = [Type = type any, Optional = true] ]
```

Type.ReplaceFacets

Type.ReplaceFacets(type as type, facets as record) as type

Описание

Заменяет аспекты `type` на аспекты из записи `facets`.

Type.ReplaceTableKeys

Type.ReplaceTableKeys(tableType as type, keys as list) as type

Описание

Возвращает новый тип таблицы, где все ключи заменены указанным списком ключей.

Type.TableColumn

Type.TableColumn(tableType as type, column as text) as type

Описание

Возвращает тип столбца `column` в типе таблицы `tableType`.

Type.TableKeys

Type.TableKeys(tableType as type) as list

Описание

Возвращает (возможно, пустой) список ключей для данного типа таблицы.

Type.TableRow

Type.TableRow(table as type) as type

Описание

Type.TableRow

Type.TableSchema

Type.TableSchema(tableType as type) as table

Описание

Возвращает таблицу, описывающую столбцы `tableType`.

Описание результирующей таблицы см. в документации по `Table.Schema`.

Type.Union

Type.Union(types as list) as type

Описание

Возвращает объединение типов в `types`.

Uri

Uri.BuildQueryString

Uri.BuildQueryString(query as record) as text

Описание

Соберите запись `query` в строку запроса универсального кода ресурса, при необходимости добавив escape-символы.

Пример №-1

Описание

Закодируйте строку запроса, которая содержит специальные символы.

Код

```
Uri.BuildQueryString([a="1", b="+$"])
```

Результат

```
"a=1&b=%2B%24"
```

Uri.Combine

Uri.Combine(baseUri as text, relativeUri as text) as text

Описание

Возвращает абсолютный URI, представляющие собой сочетание входных данных `baseUri` и `relativeUri`.

Uri.EscapeDataString

Uri.EscapeDataString(data as text) as text

Описание

Кодирует специальные символы во входных данных `data` в соответствии с правилами RFC 3986.

Пример №-1

Описание

Закодируйте специальные символы в "+money$".

Код

```
Uri.EscapeDataString("+money$")
```

Результат

```
"%2Bmoney%24"
```

Uri.Parts

Uri.Parts(absoluteUri as text) as record

Описание

Возвращает (как запись) части введенного `absoluteUri` с такими значениями, как схема, узел, порт, путь, запрос, фрагмент, имя пользователя и пароль.

Пример №-1

Описание

Найти части абсолютного URI "www.adventure-works.com".

Код

```
Uri.Parts("www.adventure-works.com")
```

Результат

```
[ Scheme = "http",
    Host = "www.adventure-works.com",
    Port = 80,
    Path = "/",
    Query = [],
    Fragment = "",
    UserName = "",
    Password = "" ]
```

Пример №-2

Описание

Декодировать закодированную строку.

Код

```
let
    UriUnescapeDataString = (data as text) as text => Uri.Parts("http://contoso?a=" &
data)[Query][a]
in
    UriUnescapeDataString("%2Bmoney%24")
```

Результат

"+money$"

Value

Value.Add

Value.Add(value1 as any, value2 as any, optional precision as number) as any

Описание

Возвращает сумму значений value1 и value2. Можно указать необязательный параметр precision, по умолчанию используется Precision.Double.

Value.As

Value.As(value as any, type as type) as any

Описание

Value.As

Value.Compare

Value.Compare(value1 as any, value2 as any, optional precision as number) as number

Описание

Возвращает значение -1, 0 или 1, если первое значение соответственно меньше второго, равно ему или больше него.

Value.Divide

Value.Divide(value1 as any, value2 as any, optional precision as number) as any

Описание

Возвращает результат деления value1 на value2. Можно указать необязательный параметр precision, по умолчанию используется Precision.Double.

Value.Equals

Value.Equals(value1 as any, value2 as any, optional precision as number) as logical

Описание

Возвращает значение true, если значение value1 равно значению value2, в противном случае - false.

Value.Firewall

Value.Firewall(key as text) as any

Описание

Value.Firewall

Value.FromText

Value.FromText(text as any, optional culture as text) as any

Описание

Расшифровывает значение из текстового представления `text` и преобразует его в качестве значения с соответствующим типом. `Value.FromText` принимает текстовое значение и возвращает число, логическое значение, значение NULL, значение даты и времени, значение длительности или текстовое значение. Пустое текстовое значение интерпретируется как значение NULL.

Value.Is

Value.Is(value as any, type as type) as logical

Описание

Value.Is

Value.Metadata

Value.Metadata(value as any) as any

Описание

Возвращает запись, содержащую метаданные входных данных.

Value.Multiply

Value.Multiply(value1 as any, value2 as any, optional precision as number) as any

Описание

Возвращает результат умножения `value1` на `value2`. Можно указать необязательный параметр `precision`, по умолчанию используется `Precision.Double`.

Value.NativeQuery

Value.NativeQuery(target as any, query as text, optional parameters as any, optional options as record) as any

Описание

Вычисляет `query` в `target` с помощью параметров, указанных в `parameters` и в `options`.

Выходные данные запроса определяются `target`.

В `target` предоставляется контекст для операции, описанной `query`.

В `query` описывается запрос, который будет выполнен в `target`. `query` обычно выражается так же, как в `target` (например, в инструкции T-SQL).

Необязательное значение `parameters` может содержать соответствующий список или запись для предоставления значений параметров, ожидаемых `query`.

Необязательная запись `options` может содержать параметры, которые влияют на поведение выполнения `query` в `target`. Эти параметры относятся к `target`.

Value.NullableEquals

Value.NullableEquals(value1 as any, value2 as any, optional precision as number) as logical

Описание

Возвращает NULL, если любой из аргументов "`value1`" и "`value2`" равен NULL, в противном случае - эквивалент Value.Equals.

Value.RemoveMetadata

Value.RemoveMetadata(value as any, optional metaValue as any) as any

Описание

Удаляет входные данные метаданных.

Value.ReplaceMetadata

Value.ReplaceMetadata(value as any, metaValue as any) as any

Описание

Заменяет метаданные входных данных.

Value.ReplaceType

Value.ReplaceType(value as any, type as type) as any

Описание

Value.ReplaceType

Value.ResourceExpression

Value.ResourceExpression(value as any) as any

Описание

Value.ResourceExpression

Value.Subtract

Value.Subtract(value1 as any, value2 as any, optional precision as number) as any

Описание

Возвращает разность `value1` и `value2`. Можно указать необязательный параметр `precision`, по умолчанию используется `Precision.Double`.

Value.Type

Value.Type(value as any) as type

Описание

Возвращает тип данного значения.

Variable

Variable.Value

Variable.Value(identifier as text) as any

Описание

Variable.Value

Web

Web.Contents

Web.Contents(url as text, optional options as record) as binary

Описание

Возвращает содержимое, скачанное из `url`, в виде двоичного файла. Необязательный параметр записи `options` может содержать дополнительные свойства. Запись может содержать следующие поля.

- `Query`: добавление параметров запроса в URL-адрес программным способом без добавления escape-символов.
- `ApiKeyName`: если целевой сайт уведомляет о ключе API, этот параметр может быть использован для указания имени (но не значения) параметра ключа, который должен быть использован в URL-адресе. Фактическое значение ключа указывается в учетных данных.
- `Content`: указание этого значения меняет тип веб-запроса с GET на POST, используя значение поля `Content` в качестве содержимого POST-запроса.
- `Headers`: при указании этого значения в качестве записи в HTTP-запрос будут внесены дополнительные заголовки.
- `Timeout`: указание этого значения в качестве длительности изменит время ожидания для HTTP-запроса. Значение по умолчанию — 100 секунд.
- `ExcludedFromCacheKey`: при указании этого значения в качестве списка из вычислений для кэширования данных будут исключены соответствующие ключи заголовка HTTP.
- `IsRetry`: при указании этого логического значения равным true все существующие ответы в кэше будут пропущены при получении данных.
- `ManualStatusHandling`: указание этого значения как списка предотвратит встроенную обработку для HTTP-запросов, ответ которых содержит один из указанных кодов состояния.
- `RelativePath`: указание этого значения как текста добавляет его к базовому URL-адресу до выполнения запроса.

Web.Page

Web.Page(html as any) as table

Описание

Возвращает содержимое документа HTML, разбитого на составные структуры, а также представление полного документа и его текста после удаления тегов.

Xml

Xml.Document

Xml.Document(contents as any, optional encoding as number) as table

Описание

Возвращает содержимое XML-документа в виде иерархической таблицы.

Xml.Tables

Xml.Tables(contents as any, optional options as record, optional encoding as number) as table

Описание

Возвращает содержимое XML-документа как вложенную коллекцию преобразованных в плоскую структуру таблиц.

Встроенные константы

Имя перечисления	Значение	Описание
RelativePosition.FromStart	0	
RelativePosition.FromEnd	1	
TextEncoding.Utf8	65001	
TextEncoding.Utf16	1200	
TextEncoding.Ascii	20127	
TextEncoding.Unicode	1200	
TextEncoding.BigEndianUnicode	1201	
TextEncoding.Windows	1252	
Culture.Current	ru-RU	
Day.Sunday	0	
Day.Monday	1	
Day.Tuesday	2	
Day.Wednesday	3	
Day.Thursday	4	
Day.Friday	5	
Day.Saturday	6	
JoinKind.Inner	0	

JoinKind.LeftOuter	1
JoinKind.RightOuter	2
JoinKind.FullOuter	3
JoinKind.LeftAnti	4
JoinKind.RightAnti	5
MissingField.Error	0
MissingField.Ignore	1
MissingField.UseNull	2
GroupKind.Global	1
GroupKind.Local	0
Number.E	2,71828182845905
Number.PI	3,14159265358979
RoundingMode.Up	0
RoundingMode.Down	1
RoundingMode.AwayFromZero	2
RoundingMode.TowardZero	3
RoundingMode.ToEven	4
Precision.Double	0
Precision.Decimal	1
Number.NaN	Nam
Number.NegativeInfinity	-Infinity
Number.PositiveInfinity	Infinity
Number.Epsilon	4,94065645841247E-324
BinaryEncoding.Hex	1
BinaryEncoding.Base64	0
Compression.GZip	0
Compression.Deflate	1
Order.Ascending	0
Order.Descending	1
Occurrence.All	2
Occurrence.First	0
Occurrence.Last	1
TraceLevel.Critical	1
TraceLevel.Error	2
TraceLevel.Warning	4
TraceLevel.Information	8
TraceLevel.Verbose	16
ExtraValues.List	0
ExtraValues.Ignore	2
ExtraValues.Error	1
QuoteStyle.None	0
QuoteStyle.Csv	1
CsvStyle.QuoteAlways	1
CsvStyle.QuoteAfterDelimiter	0
ByteOrder.LittleEndian	0
ByteOrder.BigEndian	1
Occurrence.Optional	0
Occurrence.Required	1

Occurrence.Repeating		2	
BinaryOccurrence.Optional		0	
BinaryOccurrence.Required		1	
BinaryOccurrence.Repeating		2	
JoinAlgorithm.Dynamic		0	
JoinAlgorithm.PairwiseHash		1	
JoinAlgorithm.SortMerge		2	
JoinAlgorithm.LeftHash		3	
JoinAlgorithm.RightHash		4	
JoinAlgorithm.LeftIndex		5	
JoinAlgorithm.RightIndex		6	
WebMethod.Delete	DELETE		
WebMethod.Get	GET		
WebMethod.Head	HEAD		
WebMethod.Patch	PATCH		
WebMethod.Post	POST		
WebMethod.Put	PUT		
SapHanaRangeOperator.GreaterThan		0	Больше
SapHanaRangeOperator.LessThan		1	Меньше
SapHanaRangeOperator.GreaterThanOrEquals		2	Больше или равно
SapHanaRangeOperator.LessThanOrEquals		3	Меньше или равно
SapHanaRangeOperator.Equals		4	Равно
SapHanaRangeOperator.NotEquals		5	Не равно

Типы данных

Имя типа	Описание	Базовый
Any.Type	Тип, представляющий все значения.	any
Binary.Type	Тип, представляющий все двоичные значения.	binary
BinaryEncoding.Type	Указывает тип двоичного кодирования.	number
BinaryOccurrence.Type	Указывает возможное количество вхождений элемента в группе.	number
Byte.Type	Тип, представляющий все байты.	number
ByteOrder.Type	Указывает порядок байтов.	number
Character.Type	Тип, представляющий все символы.	text
Compression.Type	Указывает тип сжатия.	number
CsvStyle.Type	Описывает значение кавычек в CSV-документах.	number
Currency.Type	Тип, представляющий денежное значение.	number
Date.Type	Тип, представляющий все значения даты.	date
DateTime.Type	Тип, представляющий все значения даты и времени без связанного часового пояса.	datetime
DateTimeZone.Type	Тип, представляющий все значения даты и времени относительно часового пояса.	datetimezone
Day.Type	Указывает день недели.	number
Decimal.Type	Тип, представляющий десятичное число с фиксированной запятой.	number
Double.Type	Тип, представляющий число с плавающей запятой двойной точности.	number

Duration.Type	Тип, представляющий все значения длительности	duration
ExtraValues.Type	Указывает ожидаемое действие для дополнительных значений в строке, которая содержит больше столбцов, чем ожидается.	number
Function.Type	Тип, представляющий все функции.	function
GroupKind.Type	Указывает вид группировки.	number
Int16.Type	Тип, представляющий 16-разрядное целое число со знаком.	number
Int32.Type	Тип, представляющий 32-разрядное целое число со знаком.	number
Int64.Type	Тип, представляющий 64-разрядное целое число со знаком.	number
Int8.Type	Тип, представляющий 8-разрядное целое число со знаком.	number
JoinAlgorithm.Type	Указывает алгоритм соединения для использования в операции соединения.	number
JoinKind.Type	Указывает вид операции соединения.	number
List.Type	Тип, представляющий все списки.	list
Logical.Type	Тип, представляющий все логические значения.	logical
MissingField.Type	Указывает ожидаемое действие для отсутствующих значений в строке, которая содержит меньшее количество столбцов, чем ожидается.	number
None.Type	None.Type	none
Null.Type	Тип, представляющий значение NULL.	null
Number.Type	Тип, представляющий все числа.	number
Occurrence.Type	Указывает вхождение элемента в последовательности.	number
Order.Type	Указывает направление сортировки.	number
Percentage.Type	Тип, представляющий значение процента.	number
Precision.Type	Указывает точность сравнения.	number
QuoteStyle.Type	Указывает стиль выделения цитаты.	number
Record.Type	Тип, представляющий все записи.	record
RelativePosition.Type	Указывает, откуда вести индексацию — с начала или с конца входных данных.	number
RoundingMode.Type	Указывает направление округления, если округление возможно в обе стороны.	number
SapHanaRangeOperator.Type	Оператор диапазона для входных параметров диапазона SAP HANA.	number
Single.Type	Тип, представляющий число с плавающей запятой одинарной точности.	number
Table.Type	Тип, представляющий все таблицы.	table
Text.Type	Тип, представляющий все текстовые значения.	text
TextEncoding.Type	Указывает тип кодировки текста.	number
Time.Type	Тип, представляющий все значения времени.	time
TraceLevel.Type	Указывает уровень трассировки.	number
Type.Type	Тип, представляющий все типы.	type
WebMethod.Type	Указывает метод HTTP.	number