Access.Database
Returns a structural representation of an Microsoft Access database.
Syntax: Access.Database(database as binary, optional options as nullable record) as table

AccessControlEntry.ConditionToIdentities
Returns a list of identities that the condition will accept.
Syntax: AccessControlEntry.ConditionToIdentities(identityProvider as function, condition as function) as

AccessControlKind.Allow
Access is allowed.
Value: 1

AccessControlKind.Deny
Access is denied.
Value: 0

ActiveDirectory.Domains
Returns a list of Active Directory domains in the same forest as the specified domain or of the current machine's domain if none is specified.
Syntax: ActiveDirectory.Domains(optional forestRootDomainName as nullable text) as table

AdobeAnalytics.Cubes
Returns the report suites in Adobe Analytics.
Syntax: AdobeAnalytics.Cubes(optional options as nullable record) as table

AdoDotNet.DataSource
Returns the schema collection for an ADO.NET data source.
Syntax: AdoDotNet.DataSource(providerName as text, connectionString as any, optional options as nullable

AdoDotNet.Query
Returns the schema collection for an ADO.NET data source.
Syntax: AdoDotNet.Query(providerName as text, connectionString as any, query as text, optional options as

AnalysisServices.Database
Returns a table of multidimensional cubes or tabular models from the Analysis Services database.
Syntax: AnalysisServices.Database(server as text, database as text, optional options as nullable record)

AnalysisServices.Databases
Returns the Analysis Services databases on a particular host.
Syntax: AnalysisServices.Databases(server as text, optional options as nullable record) as table

AzureStorage.BlobContents
Returns the content of the specified blob from an Azure storage vault.
Syntax: AzureStorage.BlobContents(url as text, optional options as nullable record) as binary

AzureStorage.Blobs
Returns a navigational table containing all containers found in the Azure Storage account. Each row has the container name and a link to the container blobs.
Syntax: AzureStorage.Blobs(account as text, optional options as nullable record) as table

AzureStorage.DataLake
Returns a navigational table containing the documents found in the specified container and its subfolders from Azure Data Lake Storage.
Syntax: AzureStorage.DataLake(endpoint as text, optional options as nullable record) as table

AzureStorage.DataLakeContents
Returns the content of the specified file from an Azure Data Lake Storage filesystem.
Syntax: AzureStorage.DataLakeContents(url as text, optional options as nullable record) as binary

AzureStorage.Tables
Returns a navigational table containing a row for each table found at the account URL from an Azure storage vault. Each row contains a link to the azure table.
Syntax: AzureStorage.Tables(account as text) as table

Binary.Buffer
Buffers the binary value in memory. The result of this call is a stable binary value, which means it will have a deterministic length and order of bytes.
Syntax: Binary.Buffer(binary as nullable binary) as nullable binary
Example: Binary.Buffer(Binary.FromList({0..10}))
Answer: #binary({0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10})

Binary.Combine
Combines a list of binaries into a single binary.
Syntax: Binary.Combine(binaries as list) as binary

Binary.Compress
Compresses a binary value using the given compression type.
Syntax: Binary.Compress(binary as nullable binary, compressionType as number) as nullable binary
Example: Binary.Compress(Binary.FromList(List.Repeat({10}, 1000)), Compression.Deflate)
Answer: #binary({227, 226, 26, 5, 163, 96, 20, 12, 119, 0, 0})

Binary.Decompress
Decompresses a binary value using the given compression type.
Syntax: Binary.Decompress(binary as nullable binary, compressionType as number) as nullable binary
Example: Binary.Decompress(#binary({115, 103, 200, 7, 194, 20, 134, 36, 134, 74, 134, 84, 6, 0}), Compression.Deflate)
Answer: #binary({71, 0, 111, 0, 111, 0, 100, 0, 98, 0, 121, 0, 101, 0})

Binary.From
Returns a binary value from the given value.
Syntax: Binary.From(value as any, optional encoding as nullable number) as nullable binary
Example: Binary.From("1011")
Answer: Binary.FromText("1011", BinaryEncoding.Base64)

Binary.FromList
Converts a list of numbers into a binary value
Syntax: Binary.FromList(list as list) as binary

Binary.FromText
Decodes data from a text form into binary.
Syntax: Binary.FromText(text as nullable text, optional encoding as nullable number) as nullable binary
Example: Binary.FromText("1011")
Answer: Binary.FromText("1011", BinaryEncoding.Base64)
Example: Binary.FromText("1011", BinaryEncoding.Hex)
Answer: Binary.FromText("EBE=", BinaryEncoding.Base64)

Binary.InferContentType
Returns a record with field Content.Type that contains the inferred MIME-type.
Syntax: Binary.InferContentType(source as binary) as record

Binary.Length
Returns the length of binary values.
Syntax: Binary.Length(binary as nullable binary) as nullable number

Binary.ToList
Converts a binary value into a list of numbers
Syntax: Binary.ToList(binary as binary) as list

Binary.ToText
Encodes binary data into a text form.
Syntax: Binary.ToText(binary as nullable binary, optional encoding as nullable number) as nullable text

BinaryEncoding.Base64
Constant to use as the encoding type when base-64 encoding is required.
Value: 0

BinaryEncoding.Hex
Constant to use as the encoding type when hexadecimal encoding is required.
Value: 1

BinaryFormat.7BitEncodedSignedInteger
A binary format that reads a 64-bit signed integer that was encoded using a 7-bit variable-length encoding.
Syntax: BinaryFormat.7BitEncodedSignedInteger(binary as binary) as any

BinaryFormat.7BitEncodedUnsignedInteger
A binary format that reads a 64-bit unsigned integer that was encoded using a 7-bit variable-length encoding.
Syntax: BinaryFormat.7BitEncodedUnsignedInteger(binary as binary) as any

BinaryFormat.Binary
Returns a binary format that reads a binary value.
Syntax: BinaryFormat.Binary(optional length as any) as function

BinaryFormat.Byte

A binary format that reads an 8-bit unsigned integer.

Syntax: BinaryFormat.Byte(binary as binary) as any

BinaryFormat.ByteOrder

Returns a binary format with the byte order specified by a function.

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

BinaryFormat.Choice

Returns a binary format that chooses the next binary format based on a value that has already been read.

Syntax: BinaryFormat.Choice(binaryFormat as function, chooseFunction as function, optional type as

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

Answer: BinaryFormat.List(BinaryFormat.Byte, length)) in listFormat(binaryData)

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

Answer: BinaryFormat.Record([length = length, list = BinaryFormat.List(BinaryFormat.Byte, length)])) in

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

Answer: BinaryFormat.List(BinaryFormat.Byte, length), type list) in listFormat(binaryData)

BinaryFormat.Decimal

A binary format that reads a .NET 16-byte decimal value.

Syntax: BinaryFormat.Decimal(binary as binary) as any

BinaryFormat.Double

A binary format that reads an 8-byte IEEE double-precision floating point value.

Syntax: BinaryFormat.Double(binary as binary) as any

BinaryFormat.Group

Returns a binary format that reads a group of items. Each item value is preceded by a unique key value. The result is a list of item values.

Syntax: BinaryFormat.Group(binaryFormat as function, group as list, optional extra as nullable function,

Example: Key 1 is required, and does appear with value 11.

Answer: Key 2 repeats, and appears twice with value 22, and results in a value of { 22, 22 }.

Example: let b = #binary({ 1, 101, 1, 102 }), f = BinaryFormat.Group(BinaryFormat.Byte, { { 1, BinaryFormat.Byte,

Answer: BinaryOccurrence.Repeating, 0, (list) => List.Sum(list) }, { 2, BinaryFormat.Byte, BinaryOccurrence.Optional,

BinaryFormat.Length

Returns a binary format that limits the amount of data that can be read. Both BinaryFormat.List and BinaryFormat.Binary can be used to read until end of the data. BinaryFormat.Length can be used to limit the number of bytes that are read.

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

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

Answer: 2) in listFormat(binaryData)

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

BinaryFormat.List
Returns a binary format that reads a sequence of items and returns a list.
Syntax: BinaryFormat.List(binaryFormat as function, optional countOrCondition as any) as function
Example: let binaryData = #binary({1, 2, 3}), listFormat = BinaryFormat.List(BinaryFormat.Byte) in
Answer: listFormat(binaryData)
Example: let binaryData = #binary({1, 2, 3}), listFormat = BinaryFormat.List(BinaryFormat.Byte, 2) in
Answer: listFormat(binaryData)
Example: let binaryData = #binary({1, 2, 3}), listFormat = BinaryFormat.List(BinaryFormat.Byte, (x) => x < 2) in
Answer: listFormat(binaryData)

BinaryFormat.Null
A binary format that reads zero bytes and returns null.
Syntax: BinaryFormat.Null(binary as binary) as any

BinaryFormat.Record
Returns a binary format that reads a record. Each field in the record can have a different binary format.
Syntax: BinaryFormat.Record(record as record) as function
Example: let binaryData = #binary({ 0x00, 0x01, 0x00, 0x00, 0x00, 0x02}), recordFormat = BinaryFormat.Record([A =
Answer: BinaryFormat.UnsignedInteger16, B = BinaryFormat.UnsignedInteger32]) in recordFormat(binaryData)

BinaryFormat.SignedInteger16
A binary format that reads a 16-bit signed integer.
Syntax: BinaryFormat.SignedInteger16(binary as binary) as any

BinaryFormat.SignedInteger32
A binary format that reads a 32-bit signed integer.
Syntax: BinaryFormat.SignedInteger32(binary as binary) as any

BinaryFormat.SignedInteger64
A binary format that reads a 64-bit signed integer.
Syntax: BinaryFormat.SignedInteger64(binary as binary) as any

BinaryFormat.Single
A binary format that reads a 4-byte IEEE single-precision floating point value.
Syntax: BinaryFormat.Single(binary as binary) as any

BinaryFormat.Text
Returns a binary format that reads a text value. The optional encoding value specifies the encoding of the text.
Syntax: BinaryFormat.Text(length as any, optional encoding as nullable number) as function
Example: let binaryData = #binary({65, 66, 67}), textFormat = BinaryFormat.Text(2, TextEncoding.Ascii) in
Answer: textFormat(binaryData)

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

BinaryFormat.Transform
Returns a binary format that will transform the values read by another binary format.
Syntax: BinaryFormat.Transform(binaryFormat as function, function as function) as function
Example: let binaryData = #binary({1}), transformFormat = BinaryFormat.Transform(BinaryFormat.Byte, (x) => x + 1) in
Answer: transformFormat(binaryData)

BinaryFormat.UnsignedInteger16
A binary format that reads a 16-bit unsigned integer.
Syntax: BinaryFormat.UnsignedInteger16(binary as binary) as any

BinaryFormat.UnsignedInteger32
A binary format that reads a 32-bit unsigned integer.
Syntax: BinaryFormat.UnsignedInteger32(binary as binary) as any

BinaryFormat.UnsignedInteger64
A binary format that reads a 64-bit unsigned integer.
Syntax: BinaryFormat.UnsignedInteger64(binary as binary) as any

BinaryOccurrence.Optional
The item is expected to appear zero or one time in the input.
Value: 0

BinaryOccurrence.Repeating
The item is expected to appear zero or more times in the input.
Value: 2

BinaryOccurrence.Required
The item is expected to appear once in the input.
Value: 1

Byte.From
Returns a 8-bit integer number value from the given value.
Syntax: Byte.From(value as any, optional culture as nullable text, optional roundingMode as nullable
Example: Byte.From("4")
Answer: 4
Example: Byte.From("4.5", null, RoundingMode.AwayFromZero)
Answer: 5

ByteOrder.BigEndian
A possible value for the byteOrder parameter in BinaryFormat.ByteOrder. The most signficant byte appears first in Big Endian byte order.
Value: 1

ByteOrder.LittleEndian
A possible value for the byteOrder parameter in BinaryFormat.ByteOrder. The least signficant byte appears first in Little Endian byte order.

Value: 0

Character.FromNumber
Returns a number to its character value.
Syntax: Character.FromNumber(number as nullable number) as nullable text
Example: Character.FromNumber(9)

Character.ToNumber
Returns a character to its number value.
Syntax: Character.ToNumber(character as nullable text) as nullable number
Example: Character.ToNumber("#(tab)")

Combiner.CombineTextByDelimiter
Returns a function that combines a list of text into a single text using the specified delimiter.
Syntax: Combiner.CombineTextByDelimiter(delimiter as text, optional quoteStyle as nullable number) as

Combiner.CombineTextByEachDelimiter
Returns a function that combines a list of text into a single text using each specified delimiter in sequence.
Syntax: Combiner.CombineTextByEachDelimiter(delimiters as list, optional quoteStyle as nullable number) as

Combiner.CombineTextByLengths
Returns a function that combines a list of text into a single text using the specified lengths.
Syntax: Combiner.CombineTextByLengths(lengths as list, optional template as nullable text) as function

Combiner.CombineTextByPositions
Returns a function that combines a list of text into a single text using the specified positions.
Syntax: Combiner.CombineTextByPositions(positions as list, optional template as nullable text) as function

Combiner.CombineTextByRanges
Returns a function that combines a list of text into a single text using the specified positions and lengths.
Syntax: Combiner.CombineTextByRanges(ranges as list, optional template as nullable text) as function

Comparer.Equals
Returns a logical value based on the equality check over the two given values.
Syntax: Comparer.Equals(comparer as function, x as any, y as any) as logical
Example: Comparer.Equals(Comparer.FromCulture("en-us"), "1", "A")

Comparer.FromCulture
Returns a comparer function given the culture and a logical value for case sensitivity for the comparison. The default value for ignoreCase is false. The value for culture are well known text representations of locales used in the .NET framework.
Syntax: Comparer.FromCulture(culture as text, optional ignoreCase as nullable logical) as function
Example: Comparer.FromCulture("en-us")("a", "A")
Example: Comparer.FromCulture("en-us", true)("a", "A")

Comparer.Ordinal

Returns a comparer function which uses Ordinal rules to compare values.
Syntax: Comparer.Ordinal(x as any, y as any) as number
Example: Comparer.Equals(Comparer.Ordinal, "encyclop√¶dia", "encyclopaedia")
Answer: FALSE

Comparer.OrdinalIgnoreCase

Returns a case-insensitive comparer function which uses Ordinal rules to compare the provided values x and y.
Syntax: Comparer.OrdinalIgnoreCase(x as any, y as any) as number

Compression.Deflate

The compressed data is in the 'Deflate' format.
Value: 1

Compression.GZip

The compressed data is in the 'GZip' format.
Value: 0

Csv.Document

Returns the contents of a CSV document as a table using the specified encoding.
Syntax: Csv.Document(source as any, optional columns as any, optional delimiter as any, optional
Example: Table.PromoteHeaders(Csv.Document("OrderID,Item 1,Fishing rod 2,1 lb. worms"))
Answer: ORDERID ITEM

CsvStyle.QuoteAfterDelimiter

Quotes in a field are only significant immediately following the delimiter.
Syntax: CsvStyle.QuoteAfterDelimiter
Value: 0

CsvStyle.QuoteAlways

Quotes in a field are always significant regardless of where they appear.
Syntax: CsvStyle.QuoteAlways
Value: 1

Cube.AddAndExpandDimensionColumn

Merges the specified dimension table, dimensionSelector, into the cube’s, cube, filter context and changes the dimensional granularity by expanding the specified set, attributeNames, of dimension attributes.
Syntax: Cube.AddAndExpandDimensionColumn(**cube** as table, **dimensionSelector** as any,

Cube.AddMeasureColumn

Adds a column with the name column to the cube that contains the results of the measure measureSelector applied in the row context of each row.
Syntax: Cube.AddMeasureColumn(**cube** as table, **column** as text, **measureSelector** as any) as table

Cube.ApplyParameter

Returns a cube after applying parameter with arguments to cube.
Syntax: Cube.ApplyParameter(cube as table, parameter as any, optional arguments as nullable list) as table

Cube.AttributeMemberId
Returns the unique member identifier from a member property value.
Syntax: Cube.AttributeMemberId(attribute as any) as any

Cube.AttributeMemberProperty
Returns the property propertyName of dimension attribute attribute.
Syntax: Cube.AttributeMemberProperty(attribute as any, propertyName as text) as any

Cube.CollapseAndRemoveColumns
Changes the dimensional granularity of the filter context for the cube by collapsing the attributes mapped to the specified columns columnNames.
Syntax: Cube.CollapseAndRemoveColumns(**cube** as table, **columnNames** as list) as table

Cube.Dimensions
Returns a table containing the set of available dimensions within the cube.
Syntax: Cube.Dimensions(**cube** as table) as table

Cube.DisplayFolders
Returns a nested tree of tables representing the display folder hierarchy of the objects (e.g. dimensions and measures) available for use in the cube.
Syntax: Cube.DisplayFolders(**cube** as table) as table

Cube.MeasureProperties
Returns a table containing the set of available properties for measures that are expanded in the cube.
Syntax: Cube.MeasureProperties(cube as table) as table

Cube.MeasureProperty
Returns the property of a measure.
Syntax: Cube.MeasureProperty(measure as any, propertyName as text) as any

Cube.Measures
Returns a table containing the set of available measures within the cube.
Syntax: Cube.Measures(**cube** as any) as table

Cube.Parameters
Returns a table containing the set of parameters that can be applied to cube.
Syntax: Cube.Parameters(cube as table) as table

Cube.Properties
Returns a table containing the set of available properties for dimensions that are expanded in the cube.
Syntax: Cube.Properties(cube as table) as table

Cube.PropertyKey
Returns the key of property property.
Syntax: Cube.PropertyKey(property as any) as any

Cube.ReplaceDimensions
0

Syntax: Cube.ReplaceDimensions(cube as table, dimensions as table) as table

Cube.Transform
Applies the list cube functions, transforms, on the cube.
Syntax: Cube.Transform(cube as table, transforms as list) as table

Culture.Current
Returns the current culture of the system.

Currency.From
Returns a currency value from the given value.
Syntax: Currency.From(value as any, optional culture as nullable text, optional roundingMode as nullable
Example: Currency.From("1.23455")
Answer: 1.2346
Example: Currency.From("1.23455", "en-Us", RoundingMode.Down)
Answer: 1.2345

DataLake.Contents

DataLake.Files

Date.AddDays
Returns a Date/DateTime/DateTimeZone value with the day portion incremented by the number of days provided. It also handles incrementing the month and year potions of the value as appropriate.
Syntax: Date.AddDays(dateTime as any, numberOfDays as number) as any
Example: Date.AddDays(#date(2011, 5, 14), 5)

Date.AddMonths
Returns a DateTime value with the month portion incremented by n months.
Syntax: Date.AddMonths(dateTime as any, numberOfMonths as number) as any
Example: Date.AddMonths(#datetime(2011, 5, 14, 8, 15, 22), 18)

Date.AddQuarters
Returns a Date/DateTime/DateTimeZone value incremented by the number of quarters provided. Each quarter is defined as a duration of three months. It also handles incrementing the year potion of the value as appropriate.
Syntax: Date.AddQuarters(dateTime as any, numberOfQuarters as number) as any
Example: Date.AddQuarters(#date(2011, 5, 14), 1)

Date.AddWeeks
Returns a Date/DateTime/DateTimeZone value incremented by the number of weeks provided. Each week is defined as a duration of seven days. It also handles incrementing the month and year potions of the value as appropriate.
Syntax: Date.AddWeeks(dateTime as any, numberOfWeeks as number) as any
Example: Date.AddWeeks(#date(2011, 5, 14), 2)

Date.AddYears
Returns a DateTime value with the year portion incremented by n years.

Syntax: Date.AddYears(dateTime as any, numberOfYears as number) as any
Example: Date.AddYears(#datetime(2011, 5, 14, 8, 15, 22), 10)

Date.Day
Returns the day for a DateTime value.
Syntax: Date.Day(dateTime as any) as nullable number
Example: Date.Day(#datetime(2011, 5, 14, 17, 0, 0))
Answer: 14

Date.DayOfWeek
Returns a number (from 0 to 6) indicating the day of the week of the provided value.
Syntax: Date.DayOfWeek(dateTime as any, optional firstDayOfWeek as nullable number [varies according to culture]) as nullable number
Example: Date.DayOfWeek(#date(2011, 02, 21), Day.Monday)
Answer: 0 Monday

Date.DayOfWeekName
Returns the day of the week name.
Syntax: Date.DayOfWeekName(date as any, optional culture as nullable text)
Example: Date.DayOfWeekName(#date(2011, 12, 31), "en-US")

Date.DayOfYear
Returns a number that represents the day of the year from a DateTime value.
Syntax: Date.DayOfYear(dateTime as any) as nullable number
Example: Date.DayOfYear(#date(2011, 03, 01))
Answer: 60

Date.DaysInMonth
Returns the number of days in the month from a DateTime value.
Syntax: Date.DaysInMonth(dateTime as any) as nullable number
Example: Date.DaysInMonth(#date(2011, 12, 01))
Answer: 31

Date.EndOfDay
Returns a DateTime value for the end of the day.
Example: Date.EndOfDay(#datetime(2011, 5, 14, 17, 0, 0))
Answer: #datetime(2011, 5, 14, 23, 59, 59.9999999)
Example: Date.EndOfDay(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0))
Answer: #datetimezone(2011, 5, 17, 23, 59, 59.9999999, -7, 0)

Date.EndOfMonth
Returns a DateTime value for the end of the month.
Syntax: Date.EndOfMonth(dateTime as any) as any
Example: Date.EndOfMonth(#date(2011, 5, 14))

Date.EndOfQuarter
Returns a Date/DateTime/DateTimeZone value representing the end of the quarter. The date and time portions are reset to their terminating values for the quarter. The timezone information is persisted.
Syntax: Date.EndOfQuarter(dateTime as any) as any
Example: Date.EndOfQuarter(#datetime(2011, 10, 10, 8, 0, 0))

Date.EndOfWeek
Returns a DateTime value for the end of the week.
Syntax: Date.EndOfWeek(dateTime as any, optional firstDayOfWeek as nullable number [Day.Sunday]) as any
Example: Date.EndOfWeek(#date(2011, 5, 14))
Answer: #date(2011, 5, 14)
Example: Date.EndOfWeek(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0), Day.Sunday)
Answer: #datetimezone(2011, 5, 21, 23, 59, 59.9999999, -7, 0)

Date.EndOfYear
Returns a DateTime value for the end of the year.
Syntax: Date.EndOfYear(dateTime as any) as any
Example: Date.EndOfYear(#datetime(2011, 5, 14, 17, 0, 0))
Answer: #datetime(2011, 12, 31, 23, 59, 59.9999999)

Date.From
Returns a date value from a value.
Syntax: Date.From(value as any, optional culture as nullable text) as nullable date
Example: Date.From(43910)
Answer: #date(2020, 3, 20)
Example: Date.From(#datetime(1899, 12, 30, 06, 45, 12))
Answer: #date(1899, 12, 30)

Date.FromText
Returns a Date value from a set of date formats and culture value.
Syntax: Date.FromText(text as nullable text, optional culture as nullable text) as nullable date
Example: Date.FromText("2010-12-31")
Example: Date.FromText("2010, 12, 31")
Example: Date.FromText("2010, 12")
Example: Date.FromText("2010")

Date.IsInCurrentDay
Indicates whether the given datetime value dateTime occurs during the current day, as determined by the current date and time on the system.
Syntax: Date.IsInCurrentDay(dateTime as any) as nullable logical
Example: Date.IsInCurrentDay(DateTime.FixedLocalNow())

Date.IsInCurrentMonth
Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the current month, as determined by the current date and time on the system.
Syntax: Date.IsInCurrentMonth(dateTime as any) as nullable logical
Example: Date.IsInCurrentMonth(DateTime.FixedLocalNow())

Date.IsInCurrentQuarter
Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the current quarter, as determined by the current date and time on the system.
Syntax: Date.IsInCurrentQuarter(dateTime as any) as nullable logical
Example: Date.IsInCurrentQuarter(DateTime.FixedLocalNow())

Date.IsInCurrentWeek

Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the current week, as determined by the current date and time on the system.
Syntax: Date.IsInCurrentWeek(dateTime as any) as nullable logical
Example: Date.IsInCurrentWeek(DateTime.FixedLocalNow())

Date.IsInCurrentYear

Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the current year, as determined by the current date and time on the system.
Syntax: Date.IsInCurrentYear(dateTime as any) as nullable logical
Example: Date.IsInCurrentYear(DateTime.FixedLocalNow())

Date.IsInNextDay

Indicates whether the given datetime value dateTime occurs during the next day, as determined by the current date and time on the system.
Syntax: Date.IsInNextDay(dateTime as any) as nullable logical
Example: Date.IsInNextDay(Date.AddDays(DateTime.FixedLocalNow(), 1))

Date.IsInNextMonth

Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the next month, as determined by the current date and time on the system.
Syntax: Date.IsInNextMonth(dateTime as any) as nullable logical
Example: Date.IsInNextMonth(Date.AddMonths(DateTime.FixedLocalNow(), 1))

Date.IsInNextNDays

Indicates whether the given datetime value dateTime occurs during the next number of days, as determined by the current date and time on the system.
Syntax: Date.IsInNextNDays(dateTime as any, days as number) as nullable logical
Example: Date.IsInNextNDays(Date.AddDays(DateTime.FixedLocalNow(), 1), 2)

Date.IsInNextNMonths

Indicates whether the given datetime value dateTime occurs during the next number of months, as determined by the current date and time on the system.
Syntax: Date.IsInNextNMonths(dateTime as any, months as number) as nullable logical
Example: Date.IsInNextNMonths(Date.AddMonths(DateTime.FixedLocalNow(), 1), 2)

Date.IsInNextNQuarters

Indicates whether the given datetime value dateTime occurs during the next number of quarters, as determined by the current date and time on the system.
Syntax: Date.IsInNextNQuarters(dateTime as any, quarters as number) as nullable logical
Example: Date.IsInNextNQuarters(Date.AddQuarters(DateTime.FixedLocalNow(), 1), 2)

Date.IsInNextNWeeks

Indicates whether the given datetime value dateTime occurs during the next number of weeks, as determined by the current date and time on the system.
Syntax: Date.IsInNextNWeeks(dateTime as any, weeks as number) as nullable logical
Example: Date.IsInNextNWeeks(Date.AddDays(DateTime.FixedLocalNow(), 7), 2)

Date.IsInNextNYears

Indicates whether the given datetime value dateTime occurs during the next number of years, as determined by the current date and time on the system.

Syntax: Date.IsInNextNYears(dateTime as any, years as number) as nullable logical
Example: Date.IsInNextNYears(Date.AddYears(DateTime.FixedLocalNow(), 1), 2)

Date.IsInNextQuarter

Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the next quarter, as determined by the current date and time on the system.
Syntax: Date.IsInNextQuarter(dateTime as any) as nullable logical
Example: Date.IsInNextQuarter(Date.AddQuarters(DateTime.FixedLocalNow(), 1))

Date.IsInNextWeek

Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the next week, as determined by the current date and time on the system.
Syntax: Date.IsInNextWeek(dateTime as any) as nullable logical
Example: Date.IsInNextWeek(Date.AddDays(DateTime.FixedLocalNow(), 7))

Date.IsInNextYear

Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the next year, as determined by the current date and time on the system.
Syntax: Date.IsInNextYear(dateTime as any) as nullable logical
Example: Date.IsInNextYear(Date.AddYears(DateTime.FixedLocalNow(), 1))

Date.IsInPreviousDay

Indicates whether the given datetime value dateTime occurs during the previous day, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousDay(dateTime as any) as nullable logical
Example: Date.IsInPreviousDay(Date.AddDays(DateTime.FixedLocalNow(), -1))

Date.IsInPreviousMonth

Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the previous month, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousMonth(dateTime as any) as nullable logical
Example: Date.IsInPreviousMonth(Date.AddMonths(DateTime.FixedLocalNow(), -1))

Date.IsInPreviousNDays

Indicates whether the given datetime value dateTime occurs during the previous number of days, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousNDays(dateTime as any, days as number) as nullable logical
Example: Date.IsInPreviousNDays(Date.AddDays(DateTime.FixedLocalNow(), -1), 2)

Date.IsInPreviousNMonths

Indicates whether the given datetime value dateTime occurs during the previous number of months, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousNMonths(dateTime as any, months as number) as nullable logical
Example: Date.IsInPreviousNMonths(Date.AddMonths(DateTime.FixedLocalNow(), -1), 2)

Date.IsInPreviousNQuarters

Indicates whether the given datetime value dateTime occurs during the previous number of quarters, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousNQuarters(dateTime as any, quarters as number) as nullable logical
Example: Date.IsInPreviousNQuarters(Date.AddQuarters(DateTime.FixedLocalNow(), -1), 2)

Date.IsInPreviousNWeeks
Indicates whether the given datetime value dateTime occurs during the previous number of weeks, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousNWeeks(dateTime as any, weeks as number) as nullable logical
Example: Date.IsInPreviousNWeeks(Date.AddDays(DateTime.FixedLocalNow(), -7), 2)

Date.IsInPreviousNYears
Indicates whether the given datetime value dateTime occurs during the previous number of years, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousNYears(dateTime as any, years as number) as nullable logical
Example: Date.IsInPreviousNYears(Date.AddYears(DateTime.FixedLocalNow(), -1), 2)

Date.IsInPreviousQuarter
Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the previous quarter, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousQuarter(dateTime as any) as nullable logical
Example: Date.IsInPreviousQuarter(Date.AddQuarters(DateTime.FixedLocalNow(), -1))

Date.IsInPreviousWeek
Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the previous week, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousWeek(dateTime as any) as nullable logical
Example: Date.IsInPreviousWeek(Date.AddDays(DateTime.FixedLocalNow(), -7))

Date.IsInPreviousYear
Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred during the previous year, as determined by the current date and time on the system.
Syntax: Date.IsInPreviousYear(dateTime as any) as nullable logical
Example: Date.IsInPreviousYear(Date.AddYears(DateTime.FixedLocalNow(), -1))

Date.IsInYearToDate
Returns a logical value indicating whether the given Date/DateTime/DateTimeZone occurred in the period starting January 1st of the current year and ending on the current day, as determined by the current date and time on the system.
Syntax: Date.IsInYearToDate(dateTime as any) as nullable logical
Example: Date.IsInYearToDate(DateTime.FixedLocalNow())

Date.IsLeapYear
Returns a logical value indicating whether the year portion of a DateTime value is a leap year.
Syntax: Date.IsLeapYear(dateTime as any) as nullable logical
Example: Date.IsLeapYear(#date(2012, 01, 01))

Date.Month
Excel equivalent: MONTH
Returns the month from a DateTime value.
Syntax: Date.Month(dateTime as any) as nullable number
Example: Date.Month(#datetime(2011, 12, 31, 9, 15, 36))

Date.MonthName
Excel equivalent: TEXT
Returns the name of the month component.

Syntax: Date.MonthName(date as any, optional culture as nullable text) as nullable text
Example: Date.MonthName(#datetime(2011, 12, 31, 5, 0, 0), "en-US")

Date.QuarterOfYear
Returns a number between 1 and 4 for the quarter of the year from a DateTime value.
Syntax: Date.QuarterOfYear(dateTime as any) as nullable number
Example: Date.QuarterOfYear(#date(2011, 12, 31))

Date.StartOfDay
Excel equivalent: INT
Returns a DateTime value for the start of the day.
Syntax: Date.StartOfDay(dateTime as any) as any
Example: Date.StartOfDay(#datetime(2011, 10, 10, 8, 0, 0))

Date.StartOfMonth
Excel equivalent: DATE
Returns a DateTime value representing the start of the month.
Syntax: Date.StartOfMonth(dateTime as any) as any
Example: Date.StartOfMonth(#datetime(2011, 10, 10, 8, 10, 32))

Date.StartOfQuarter
Excel equivalent: DATE
Returns a DateTime value representing the start of the quarter.
Syntax: Date.StartOfQuarter(dateTime as any) as any
Example: Date.StartOfQuarter(#datetime(2011, 10, 10, 8, 0, 0))

Date.StartOfWeek
Excel equivalent: WEEKDAY
Returns a DateTime value representing the start of the week.
Syntax: Date.StartOfWeek(dateTime as any, optional firstDayOfWeek as nullable number) as any
Example: Date.StartOfWeek(#datetime(2011, 10, 10, 8, 10, 32))

Date.StartOfYear
Returns a DateTime value representing the start of the year.
Syntax: Date.StartOfYear(dateTime as any) as any
Example: Date.StartOfYear(#datetime(2011, 10, 10, 8, 10, 32))

Date.ToRecord
Returns a record containing parts of a Date value.
Syntax: Date.ToRecord(date as date) as record
Example: Date.ToRecord(#date(2011, 12, 31))

Date.ToText
Returns a text value from a Date value.
Syntax: Date.ToText(date as nullable date, optional format as nullable text, optional culture as nullable text) as nullable text
Example: Date.ToText(#date(2010, 12, 31))
Example: Date.ToText(#date(2010, 12, 31), "yyyy/MM/dd")

Date.WeekOfMonth
Returns a number for the count of week in the current month.

Syntax: Date.WeekOfMonth(dateTime as any, optional firstDayOfWeek as nullable number) as nullable number
Example: Date.WeekOfMonth(#date(2011, 03, 15))

Date.WeekOfYear
Excel equivalent: WEEKNUM
Returns a number for the count of week in the current year.
Syntax: Date.WeekOfYear(dateTime as any, optional firstDayOfWeek as nullable number) as nullable number
Example: Date.WeekOfYear(#date(2011, 03, 27))

Date.Year
Excel equivalent: YEAR
Returns the year from a DateTime value.
Syntax: Date.Year(dateTime as any) as nullable number
Example: Date.Year(#datetime(2011, 12, 31, 9, 15, 36))

DateTime.AddZone
Adds the timezonehours as an offset to the input datetime value and returns a new datetimezone value.
Syntax: DateTime.AddZone(dateTime as nullable datetime, timezoneHours as number, optional timezoneMinutes as nullable number) as nullable datetimezone
Example: DateTime.AddZone(#datetime(2010, 12, 31, 11, 56, 02), 7, 30)
Answer: #datetimezone(2010, 12, 31, 11, 56, 2, 7, 30)

DateTime.Date
Returns a date part from a DateTime value
Syntax: DateTime.Date(dateTime as any) as nullable date
Example: DateTime.Date(#datetime(2010, 12, 31, 11, 56, 02))
Answer: #date(2010, 12, 31)

DateTime.FixedLocalNow
Returns a DateTime value set to the current date and time on the system.
Syntax: DateTime.FixedLocalNow() as datetime

DateTime.From
Returns a datetime value from a value.
Syntax: DateTime.From(value as any, optional culture as nullable text) as nullable datetime
Example: DateTime.From(#time(06, 45, 12))
Answer: #datetime(1899, 12, 30, 06, 45, 12)
Example: DateTime.From(#date(1975, 4, 4))
Answer: #datetime(1975, 4, 4, 0, 0, 0)

DateTime.FromFileTime
Returns a DateTime value from the supplied number.
Syntax: DateTime.FromFileTime(fileTime as nullable number) as nullable datetime
Example: DateTime.FromFileTime(129876402529842245)
Answer: #datetime(2012, 7, 24, 14, 50, 52.9842245)

DateTime.FromText
Returns a DateTime value from a set of date formats and culture value.

Syntax: DateTime.FromText(text as nullable text, optional culture as nullable text) as nullable datetime
Example: DateTime.FromText("2010-12-31T01:30:25")
Answer: #datetime(2010, 12, 31, 1, 30, 25)
Example: DateTime.FromText("2010-12-31T01:30")
Answer: #datetime(2010, 12, 31, 1, 30, 0)
Example: DateTime.FromText("20101231T013025")
Answer: #datetime(2010, 12, 31, 1, 30, 25)
Example: DateTime.FromText("20101231T01:30:25")
Answer: #datetime(2010, 12, 31, 1, 30, 25)
Example: DateTime.FromText("20101231T01:30:25.121212")
Answer: #datetime(2010, 12, 31, 1, 30, 25.121212)

DateTime.IsInCurrentHour
Indicates whether the given datetime value occurs during the current hour, as determined by the current date and time on the system.
Syntax: DateTime.IsInCurrentHour(dateTime as any) as nullable logical
Example: DateTime.IsInCurrentHour(DateTime.FixedLocalNow())

DateTime.IsInCurrentMinute
Indicates whether the given datetime value occurs during the current minute, as determined by the current date and time on the system.
Syntax: DateTime.IsInCurrentMinute(dateTime as any) as nullable logical
Example: DateTime.IsInCurrentMinute(DateTime.FixedLocalNow())

DateTime.IsInCurrentSecond
Indicates whether the given datetime value occurs during the current second, as determined by the current date and time on the system.
Syntax: DateTime.IsInCurrentSecond(dateTime as any) as nullable logical
Example: DateTime.IsInCurrentSecond(DateTime.FixedLocalNow())

DateTime.IsInNextHour
Indicates whether the given datetime value occurs during the next hour, as determined by the current date and time on the system.
Syntax: DateTime.IsInNextHour(dateTime as any) as nullable logical
Example: DateTime.IsInNextHour(DateTime.FixedLocalNow() + #duration(0,1,0,0))

DateTime.IsInNextMinute
Indicates whether the given datetime value occurs during the next minute, as determined by the current date and time on the system.
Syntax: DateTime.IsInNextMinute(dateTime as any) as nullable logical
Example: DateTime.IsInNextMinute(DateTime.FixedLocalNow() + #duration(0,0,1,0))

DateTime.IsInNextNHours
Indicates whether the given datetime value occurs during the next number of hours, as determined by the current date and time on the system.
Syntax: DateTime.IsInNextNHours(dateTime as any, hours as number) as nullable logical
Example: DateTime.IsInNextNHours(DateTime.FixedLocalNow() + #duration(0,2,0,0), 2)

DateTime.IsInNextNMinutes
Indicates whether the given datetime value occurs during the next number of minutes, as determined by the current date and time on the system.
Syntax: DateTime.IsInNextNMinutes(dateTime as any, minutes as number) as nullable logical
Example: DateTime.IsInNextNMinutes(DateTime.FixedLocalNow() + #duration(0,0,2,0), 2)

DateTime.IsInNextNSeconds
Indicates whether the given datetime value occurs during the next number of seconds, as determined by the current date and time on the system.
Syntax: DateTime.IsInNextNSeconds(dateTime as any, seconds as number) as nullable logical
Example: DateTime.IsInNextNSeconds(DateTime.FixedLocalNow() + #duration(0,0,0,2), 2)

DateTime.IsInNextSecond
Indicates whether the given datetime value occurs during the next second, as determined by the current date and time on the system.
Syntax: DateTime.IsInNextSecond(dateTime as any) as nullable logical
Example: DateTime.IsInNextSecond(DateTime.FixedLocalNow() + #duration(0,0,0,1))

DateTime.IsInPreviousHour
Indicates whether the given datetime value occurs during the previous hour, as determined by the current date and time on the system.
Syntax: DateTime.IsInPreviousHour(dateTime as any) as nullable logical
Example: DateTime.IsInPreviousHour(DateTime.FixedLocalNow() - #duration(0,1,0,0))

DateTime.IsInPreviousMinute
Indicates whether the given datetime value occurs during the previous minute, as determined by the current date and time on the system.
Syntax: DateTime.IsInPreviousMinute(dateTime as any) as nullable logical
Example: DateTime.IsInPreviousMinute(DateTime.FixedLocalNow() - #duration(0,0,1,0))

DateTime.IsInPreviousNHours
Indicates whether the given datetime value occurs during the previous number of hours, as determined by the current date and time on the system.
Syntax: DateTime.IsInPreviousNHours(dateTime as any, hours as number) as nullable logical
Example: DateTime.IsInPreviousNHours(DateTime.FixedLocalNow() - #duration(0,2,0,0), 2)

DateTime.IsInPreviousNMinutes
Indicates whether the given datetime value occurs during the previous number of minutes, as determined by the current date and time on the system.
Syntax: DateTime.IsInPreviousNMinutes(dateTime as any, minutes as number) as nullable logical
Example: DateTime.IsInPreviousNMinutes(DateTime.FixedLocalNow() - #duration(0,0,2,0), 2)

DateTime.IsInPreviousNSeconds
Indicates whether the given datetime value occurs during the previous number of seconds, as determined by the current date and time on the system.
Syntax: DateTime.IsInPreviousNSeconds(dateTime as any, seconds as number) as nullable logical
Example: DateTime.IsInPreviousNSeconds(DateTime.FixedLocalNow() - #duration(0,0,0,2), 2)

DateTime.IsInPreviousSecond
Indicates whether the given datetime value occurs during the previous second, as determined by the current date and time on the system.

Syntax: DateTime.IsInPreviousSecond(dateTime as any) as nullable logical
Example: DateTime.IsInPreviousSecond(DateTime.FixedLocalNow() - #duration(0,0,0,1))

DateTime.LocalNow
Returns a datetime value set to the current date and time on the system.
Syntax: DateTime.LocalNow() as datetime

DateTime.Time
Returns a time part from a DateTime value.
Syntax: DateTime.Time(dateTime as any) as nullable time
Example: DateTime.Time(#datetime(2010, 12, 31, 11, 56, 02))
Answer: #time(11, 56, 2)

DateTime.ToRecord
Returns a record containing parts of a DateTime value.
Syntax: DateTime.ToRecord(dateTime as datetime) as record
Example: DateTime.ToRecord(#datetime(2011, 12, 31, 11, 56, 2))

DateTime.ToText
Returns a text value from a DateTime value.
Syntax: DateTime.ToText(dateTime as nullable datetime, optional format as nullable text, optional culture as nullable text) as nullable text
Example: DateTime.ToText(#datetime(2010, 12, 31, 11, 56, 2))
Answer: "12/31/2010 11:56:02 AM"
Example: DateTime.ToText(#datetime(2010, 12, 31, 11, 56, 2), "yyyy/MM/ddThh:mm:ss")
Answer: "2010/12/31T11:56:02"

DateTimeZone.FixedLocalNow
Returns a DateTimeZone value set to the current date, time, and timezone offset on the system.
Syntax: DateTimeZone.FixedLocalNow() as datetimezone

DateTimeZone.FixedUtcNow
Returns the current date and time in UTC (the GMT timezone).
Syntax: DateTimeZone.FixedUtcNow() as datetimezone

DateTimeZone.From
Returns a datetimezone value from a value.
Syntax: DateTimeZone.From(value as any, optional culture as nullable text) as nullable datetimezone
Example: DateTimeZone.From("2020-10-30T01:30:00-08:00")
Answer: #datetimezone(2020, 10, 30, 01, 30, 00, -8, 00)

DateTimeZone.FromFileTime
Returns a DateTimeZone from a number value.
Syntax: DateTimeZone.FromFileTime(fileTime as nullable number) as nullable datetimezone
Example: DateTimeZone.FromFileTime(129876402529842245)
Answer: #datetimezone(2012, 7, 24, 14, 50, 52.9842245, -7, 0)

DateTimeZone.FromText
Returns a DateTimeZone value from a set of date formats and culture value.
Syntax: DateTimeZone.FromText(text as nullable text, optional culture as nullable text) as nullable datetimezone

Example: DateTimeZone.FromText("2010-12-31T01:30:00-08:00")
Answer: #datetimezone(2010, 12, 31, 1, 30, 0, -8, 0)
Example: DateTimeZone.FromText("2010-12-31T01:30:00.121212-08:00")
Answer: #datetimezone(2010, 12, 31, 1, 30, 0.121212, -8, 0)
Example: DateTimeZone.FromText("2010-12-31T01:30:00Z")
Answer: #datetimezone(2010, 12, 31, 1, 30, 0, 0, 0)
Example: DateTimeZone.FromText("20101231T013000+0800")
Answer: #datetimezone(2010, 12, 31, 1, 30, 0, 8, 0)

DateTimeZone.LocalNow
Returns a DateTime value set to the current system date and time.
Syntax: DateTimeZone.LocalNow() as datetimezone

DateTimeZone.RemoveZone
Returns a datetime value with the zone information removed from the input datetimezone value.
Syntax: DateTimeZone.RemoveZone(dateTimeZone as nullable datetimezone) as nullable datetime
Example: DateTimeZone.RemoveZone(#datetimezone(2011, 12, 31, 9, 15, 36,-7, 0))
Answer: #datetime(2011, 12, 31, 9, 15, 36)

DateTimeZone.SwitchZone
Changes the timezone information for the input DateTimeZone.
Syntax: DateTimeZone.SwitchZone(dateTimeZone as nullable datetimezone, timezoneHours as number, optional timezoneMinutes as nullable number) as nullable datetimezone
Example: DateTimeZone.SwitchZone(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30), 8)
Answer: #datetimezone(2010, 12, 31, 12, 26, 2, 8, 0)
Example: DateTimeZone.SwitchZone(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30), 0, -30)
Answer: #datetimezone(2010, 12, 31, 3, 56, 2, 0, -30)

DateTimeZone.ToLocal
Returns a DateTime value from the local time zone.
Syntax: DateTimeZone.ToLocal(dateTimeZone as nullable datetimezone) as nullable datetimezone
Example: DateTimeZone.ToLocal(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30))
Answer: #datetimezone(2010, 12, 31, 12, 26, 2, -8, 0)

DateTimeZone.ToRecord
Returns a record containing parts of a DateTime value.
Syntax: DateTimeZone.ToRecord(dateTimeZone as datetimezone) as record
Example: DateTimeZone.ToRecord(#datetimezone(2011, 12, 31, 11, 56, 2, 8, 0))

DateTimeZone.ToText
Returns a text value from a DateTime value.
Syntax: DateTimeZone.ToText(dateTimeZone as nullable datetimezone, optional format as nullable text, optional culture as nullable text) as nullable text
Example: DateTimeZone.ToText(#datetimezone(2010, 12, 31, 11, 56, 2, 8, 0))
Answer: "12/31/2010 11:56:02 AM +08:00"
Example: DateTimeZone.ToText(#datetimezone(2010, 12, 31, 11, 56, 2, 10, 12), "yyyy/MM/ddThh:mm:sszzz")
Answer: "2010/12/31T11:56:02+10:12"

DateTimeZone.ToUtc
Returns a DateTime value to the Utc time zone.

Syntax: DateTimeZone.ToUtc(dateTimeZone as nullable datetimezone) as nullable datetimezone
Example: DateTimeZone.ToUtc(#datetimezone(2010, 12, 31, 11, 56, 02, 7, 30))
Answer: #datetimezone(2010, 12, 31, 4, 26, 2, 0, 0)

DateTimeZone.UtcNow
Returns a DateTime value set to the current system date and time in the Utc timezone.
Syntax: DateTimeZone.UtcNow() as datetimezone
Example: DateTimeZone.UtcNow()

DateTimeZone.ZoneHours
Returns a time zone hour value from a DateTime value.
Syntax: DateTimeZone.ZoneHours(dateTimeZone as nullable datetimezone) as nullable number

DateTimeZone.ZoneMinutes
Returns a time zone minute value from a DateTime value.
Syntax: DateTimeZone.ZoneMinutes(dateTimeZone as nullable datetimezone) as nullable number

Day.Friday
Represents Friday.
Value: 5

Day.Monday
Represents Monday.
Value: 1

Day.Saturday
Represents Saturday.
Value: 6

Day.Sunday
Represents Sunday.
Value: 0

Day.Thursday
Represents Thursday.
Value: 4

Day.Tuesday
Represents Tuesday.
Value: 2

Day.Wednesday
Represents Wednesday.
Value: 3

DB2.Database
Returns a table of SQL tables and views available in a Db2 database.
Syntax: DB2.Database(server as text, database as text, optional options as nullable record) as table

Decimal.From
Returns a decimal number value from the given value.

Syntax: Decimal.From(value as any, optional culture as nullable text) as nullable number
Example: Decimal.From("4.5")
Answer: 4.5

Diagnostics.ActivityId
Returns an opaque identifier for the currently-running evaluation.
Syntax: Diagnostics.ActivityId() as nullable text

Diagnostics.Trace
Writes a trace message, if tracing is enabled, and returns value.
Syntax: Diagnostics.Trace(traceLevel as number, message as anynonnull, value as any, optional delayed as
Example: Diagnostics.Trace(TraceLevel.Information, "TextValueFromNumber", () => Text.From(123), true)

DirectQueryCapabilities.From
DirectQueryCapabilities.From
Syntax: DirectQueryCapabilities.From(value as any) as table

Double.From
Returns a Double number value from the given value.
Syntax: Double.From(value as any, optional culture as nullable text) as nullable number
Example: Double.From("4.5")
Answer: 4.5

Duration.Days
Returns the day component of a Duration value.
Syntax: Duration.Days(duration as nullable duration) as nullable number
Example: Duration.Days(#duration(5, 4, 3, 2))
Answer: 5

Duration.From
Returns a duration value from a value.
Syntax: Duration.From(value as any) as nullable duration
Example: Duration.From(2.525)
Answer: #duration(2, 12, 36, 0)

Duration.FromText
Returns a Duration value from a text value.
Syntax: Duration.FromText(text as nullable text) as nullable duration
Example: Duration.FromText("2.05:55:20")
Answer: #duration(2, 5, 55, 20)

Duration.Hours
Returns an hour component of a Duration value.
Syntax: Duration.Hours(duration as nullable duration) as nullable number
Example: Duration.Hours(#duration(5, 4, 3, 2))

Duration.Minutes
Returns a minute component of a Duration value.
Syntax: Duration.Minutes(duration as nullable duration) as nullable number

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

Duration.Seconds
Returns a second component of a Duration value.
Syntax: Duration.Seconds(duration as nullable duration) as nullable number
Example: Duration.Seconds(#duration(5, 4, 3, 2))

Duration.ToRecord
Returns a record with parts of a Duration value.
Syntax: Duration.ToRecord(duration as duration) as record
Example: Duration.ToRecord(#duration(2, 5, 55, 20))

Duration.TotalDays
Returns the total magnitude of days from a Duration value.
Syntax: Duration.TotalDays(duration as nullable duration) as nullable number
Example: Duration.TotalDays(#duration(5, 4, 3, 2))
Answer: 5.1687731481481478

Duration.TotalHours
Returns the total magnitude of hours from a Duration value.
Syntax: Duration.TotalHours(duration as nullable duration) as nullable number
Example: Duration.TotalHours(#duration(5, 4, 3, 2))
Answer: 124.05055555555555

Duration.TotalMinutes
Returns the total magnitude of minutes from a Duration value.
Syntax: Duration.TotalMinutes(duration as nullable duration) as nullable number
Example: Duration.TotalMinutes(#duration(5, 4, 3, 2))
Answer: 7443.0333333333338

Duration.TotalSeconds
Returns the total magnitude of seconds from a duration value.
Syntax: Duration.TotalSeconds(duration as nullable duration) as nullable number
Example: Duration.TotalSeconds(#duration(5, 4, 3, 2))
Answer: 446582

Duration.ToText
Returns a text value from a Duration value.
Syntax: Duration.ToText(duration as nullable duration, optional format as nullable text) as nullable text
Example: Duration.ToText(#duration(2, 5, 55, 20))
Answer: "2.05:55:20"

Embedded.Value
Accesses a value by name in an embedded mashup.
Syntax: Embedded.Value(value as any, path as text) as any

Error.Record
Returns a record containing fields “Reason”, “Message”, and “Detail” set to the provided values. The record can be used to raise or throw an error.

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

Excel.CurrentWorkbook
Returns the tables in the current Excel Workbook.
Syntax: Excel.CurrentWorkbook() as table

Excel.Workbook
Returns a table representing sheets in the given excel workbook.
Syntax: Excel.Workbook(workbook as binary, optional useHeaders as nullable logical, optional delayTypes as

Exchange.Contents
Returns a table of contents from a Microsoft Exchange account.
Syntax: Exchange.Contents (optional mailboxAddress as nullable text) as table

Expression.Constant
Returns the M source code representation of a constant value.
Syntax: Expression.Constant(value as any) as text
Example: Expression.Constant(123)
Example: Expression.Constant(#date(2035, 01, 02))
Answer: #date(2035, 1, 2)
Example: Expression.Constant("abc")

Expression.Evaluate
Returns the result of evaluating an M expression.
Syntax: Expression.Evaluate(document as text, optional environment as nullable record) as any
Example: Expression.Evaluate("1 + 1")
Example: Expression.Evaluate("List.Sum({1, 2, 3})", [List.Sum = List.Sum])
Example: Expression.Evaluate(Expression.Constant("""abc") & " & " & Expression.Identifier("x"), [x="def"""])

Expression.Identifier
Returns the M source code representation of an identifier.
Syntax: Expression.Identifier(name as text) as text
Example: Expression.Identifier("MyIdentifier")
Answer: MyIdentifier
Example: Expression.Identifier("My Identifier")
Answer: #"My Identifier"

ExtraValues.Error
If the splitter function returns more columns than the table expects, an error should be raised.
Value: 1

ExtraValues.Ignore
If the splitter function returns more columns than the table expects, they should be ignored.
Value: 2

ExtraValues.List
If the splitter function returns more columns than the table expects, they should be collected into a list.

Value: 0

Facebook.Graph
Returns a record containing content from the Facebook graph.
Syntax: Facebook.Graph(url as text) as any

File.Contents
Returns the binary contents of the file located at a path.
Syntax: File.Contents(path as text) as binary

Folder.Contents
Returns a table containing the properties and contents of the files and folders found at path.
Syntax: Folder.Contents(path as text) as table

Folder.Files
Returns a table containing a row for each file found at a folder path, and subfolders. Each row contains properties of the folder or file and a link to its content.
Syntax: Folder.Files(path as text) as table

Function.From
Takes a unary function function and creates a new function with the type functionType that constructs a list out of its arguments and passes it to function.
Syntax: Function.From(functionType as type, function as function) as function
Example: Function.From(type function (a as number, b as number) as number, List.Sum)(2, 1)
Example: Function.From(type function (a as text, b as text) as text, (list) => list{0} & list{1})("2", "1")

Function.Invoke
Invokes the given function using the specified and returns the result.
Syntax: Function.Invoke(function as function, args as list) as any
Example: Function.Invoke(Record.FieldNames, {[A=1,B=2]})

Function.InvokeAfter
Returns the result of invoking function after duration delay has passed.
Syntax: Function.InvokeAfter(function as function, delay as duration) as any

Function.IsDataSource
Returns whether or not function is considered a data source.
Syntax: Function.IsDataSource(function as function) as logical

Function.ScalarVector
Returns a scalar function of type scalarFunctionType that invokes vectorFunction with a single row of arguments and returns its single output.
Syntax: Function.ScalarVector(scalarFunctionType as type, vectorFunction as function) as function

GroupKind.Global
GroupKind.Global
Value: 1

GroupKind.Local
GroupKind.Local
Syntax: GroupKind.Local

Value: 0

Guid.From
Returns a Guid.Type value from the given value.
Syntax: Guid.From(value as nullable text) as nullable text
Example: Guid.From("05FE1DADC8C24F3BA4C2D194116B4967")
Answer: 05fe1dad-c8c2-4f3b-a4c2-d194116b4967
Example: Guid.From("05FE1DAD-C8C2-4F3B-A4C2-D194116B4967")
Answer: 05fe1dad-c8c2-4f3b-a4c2-d194116b4967
Example: Guid.From("{05FE1DAD-C8C2-4F3B-A4C2-D194116B4967}")
Answer: 05fe1dad-c8c2-4f3b-a4c2-d194116b4967
Example: Guid.From("(05FE1DAD-C8C2-4F3B-A4C2-D194116B4967)")
Answer: 05fe1dad-c8c2-4f3b-a4c2-d194116b4967

Hdfs.Contents
Returns a table containing a row for each folder and file found at the folder url, {0}, from a Hadoop file system. Each row contains properties of the folder or file and a link to its content.
Syntax: Hdfs.Contents(url as text) as table

Hdfs.Files
Returns a table containing a row for each file found at the folder url, {0}, and subfolders from a Hadoop file system. Each row contains properties of the file and a link to its content.
Syntax: Hdfs.Files(url as text) as table

HdInsight.Containers
Returns a navigational table containing all containers found in the HDInsight account. Each row has the container name and table containing its files.
Syntax: HdInsight.Containers(account as text) as table

HdInsight.Contents
Returns a navigational table containing all containers found in the HDInsight account. Each row has the container name and table containing its files.
Syntax: HdInsight.Contents(account as text) as table

HdInsight.Files
Returns a table containing a row for each folder and file found at the container URL, and subfolders from an HDInsight account. Each row contains properties of the file/folder and a link to its content.
Syntax: HdInsight.Files(account as text, containerName as text) as table

Identity.From
Creates an identity.
Syntax: Identity.From(identityProvider as function, value as any) as record

Identity.IsMemberOf
Determines whether an identity is a member of an identity collection.
Syntax: Identity.IsMemberOf(identity as record, collection as record) as logical

IdentityProvider.Default
The default identity provider for the current host.
Syntax: IdentityProvider.Default() as any

Informix.Database
Returns a table of SQL tables and views available in an Informix database on server server in the database instance named database.
Syntax: Informix.Database(server as text, database as text, optional options as nullable record) as table

Int16.From
Returns a 16-bit integer number value from the given value.
Syntax: Int16.From(value as any, optional culture as nullable text, optional roundingMode as nullable
Example: Int64.From("4")
Answer: 4
Example: Int16.From("4.5", null, RoundingMode.AwayFromZero)
Answer: 5

Int32.From
Returns a 32-bit integer number value from the given value.
Syntax: Int32.From(value as any, optional culture as nullable text, optional roundingMode as nullable
Example: Int32.From("4")
Answer: 4
Example: Int32.From("4.5", null, RoundingMode.AwayFromZero)
Answer: 5

Int64.From
Returns a 64-bit integer number value from the given value.
Syntax: Int64.From(value as any, optional culture as nullable text, optional roundingMode as nullable
Example: Int64.From("4")
Answer: 4
Example: Int64.From("4.5", null, RoundingMode.AwayFromZero)
Answer: 5

Int8.From
Returns a signed 8-bit integer number value from the given value.
Syntax: Int8.From(value as any, optional culture as nullable text, optional roundingMode as nullable
Example: Int8.From("4")
Answer: 4
Example: Int8.From("4.5", null, RoundingMode.AwayFromZero)
Answer: 5

ItemExpression.From
Returns the AST for the body of a function.
Syntax: ItemExpression.From(function as function) as record
Example: ItemExpression.From(each _ <> null)
Answer: KIND Binary

ItemExpression.Item
An AST node representing the item in an item expression.

JoinAlgorithm.Dynamic
JoinAlgorithm.Dynamic
Value: 0

JoinAlgorithm.LeftHash
JoinAlgorithm.LeftHash
Value: 3

JoinAlgorithm.LeftIndex
JoinAlgorithm.LeftIndex
Value: 5

JoinAlgorithm.PairwiseHash
JoinAlgorithm.PairwiseHash
Value: 1

JoinAlgorithm.RightHash
JoinAlgorithm.RightHash
Value: 4

JoinAlgorithm.RightIndex
JoinAlgorithm.RightIndex
Value: 6

JoinAlgorithm.SortMerge
JoinAlgorithm.SortMerge
Value: 2

JoinKind.FullOuter
A possible value for the optional JoinKind parameter in Table.Join. A full outer join ensures that all rows of both tables appear in the result. Rows that did not have a match in the other table are joined with a default row containing null values for all of its columns.
Value: 3

JoinKind.Inner
A possible value for the optional JoinKind parameter in Table.Join. The table resulting from an inner join contains a row for each pair of rows from the specified tables that were determined to match based on the specified key columns.
Value: 0

JoinKind.LeftAnti
A possible value for the optional JoinKind parameter in Table.Join. A left anti join returns that all rows from the first table which do not have a match in the second table.
Value: 4

JoinKind.LeftOuter
A possible value for the optional JoinKind parameter in Table.Join. A left outer join ensures that all rows of the first table appear in the result.
Value: 1

JoinKind.RightAnti
A possible value for the optional JoinKind parameter in Table.Join. A right anti join returns that all rows from the second table which do not have a match in the first table.
Value: 5

JoinKind.RightOuter
A possible value for the optional JoinKind parameter in Table.Join. A right outer join ensures that all rows of the second table appear in the result.
Value: 2

JoinSide.Left
Specifies the left table of a join.
Value: 0

JoinSide.Right
Specifies the right table of a join.
Value: 1

Json.Document
Returns the contents of a JSON document. The contents may be directly passed to the function as text, or it may be the binary value returned by a function like File.Contents.
Syntax: Json.Document(jsonText as any, optional encoding as nullable number) as any

Json.FromValue
Produces a JSON representation of a given value.
Syntax: Json.FromValue(value as any, optional encoding as nullable number) as binary
Example: Text.FromBinary(Json.FromValue([A={1, true, "3"}, B=#date(2012, 3, 25)]))
Answer: {"A":[1,true,"3"],"B":"2012-03-25"}

Kusto.Contents

Kusto.Databases

LimitClauseKind.AnsiSql2008
Value: 4

LimitClauseKind.Limit
Value: 3

LimitClauseKind.LimitOffset
Value: 2

LimitClauseKind.None
Value: 0

LimitClauseKind.Top
Value: 1

Lines.FromBinary
Converts a binary value to a list of text values split at lines breaks.
Syntax: Lines.FromBinary(binary as binary, optional quoteStyle as nullable number, optional

Lines.FromText
Converts a text value to a list of text values split at lines breaks.
Syntax: Lines.FromText(text as text, optional quoteStyle as nullable number, optional

Lines.ToBinary
Converts a list of text into a binary value using the specified encoding and lineSeparator.The specified lineSeparator is appended to each line. If not specified then the carriage return and line feed characters are used.
Syntax: Lines.ToBinary(lines as list, optional lineSeparator as nullable text, optional encoding as

Lines.ToText
Converts a list of text into a single text. The specified lineSeparator is appended to each line. If not specified then the carriage return and line feed characters are used.
Syntax: Lines.ToText(lines as list, optional lineSeparator as nullable text) as text

List.Accumulate
Accumulates a result from the list. Starting from the initial value seed this function applies the accumulator function and returns the final result.
Syntax: List.Accumulate(list as list, seed as any, accumulator as function) as any
Example: List.Accumulate({1, 2, 3, 4, 5}, 0, (state, current) => state + current)
Answer: 15

List.AllTrue
Excel equivalent: AND
Returns true if all expressions in a list are true
Syntax: List.AllTrue(list as list) as logical
Example: List.AllTrue({true, true, 2 > 0})
Answer: true
Example: List.AllTrue({true, false, 2 < 0})
Answer: false

List.Alternate
Returns a list with the items alternated from the original list based on a count, optional repeatInterval, and an optional offset.
Syntax: List.Alternate(list as list, count as number, optional repeatInterval as nullable number, optional offset as nullable number) as list
Example: List.Alternate({1..10}, 1)
Answer: Create a list from {1..10} that skips the first number.
Example: List.Alternate({1..10}, 1, 1)
Answer: Create a list from {1..10} that skips the every other number.
Example: List.Alternate({1..10}, 1, 1, 1) - Create a list from {1..10} that starts at 1 and skips every other number.
Example: List.Alternate({1..10}, 1, 2, 1)
Answer: Create a list from {1..10} that starts at 1, skips one value, keeps two values and so on.

List.AnyTrue
Excel equivalent: OR

Returns true if any expression in a list in true
Syntax: List.AnyTrue(list as list) as logical
Example: List.AnyTrue({true, false, 2>0})
Answer: true
Example: List.AnyTrue({2 = 0, false, 2 < 0})
Answer: false

List.Average
Excel equivalent: AVERAGE
Returns an average value from a list in the datatype of the values in the list.
Syntax: List.Average(list as list, optional precision as nullable number) as any
Example: List.Average({3, 4, 6})
Answer: 4.3333333333333333
Example: List.Average({#date(2011, 1, 1), #date(2011, 1, 2), #date(2011, 1, 3)})
Answer: #date(2011, 1, 2)

List.Buffer
Buffers the list in memory. The result of this call is a stable list, which means it will have a determinimic count, and order of items.
Syntax: List.Buffer(list as list) as list
Example: List.Buffer({1..10})

List.Combine
Merges a list of lists into single list.
Syntax: List.Combine(lists as list) as list
Example: List.Combine({{1, 2}, {3, 4}})
Example: List.Combine({{1, 2}, {3, {4, 5}}})

List.Contains
Excel equivalent: OR
Returns true if a value is found in a list.
Syntax: List.Contains(list as list, value as any, optional equationCriteria as any) as logical
Example: List.Contains({1, 2, 3, 4, 5}, 3)
Answer: true

List.ContainsAll
Excel equivalent: AND
Returns true if all items in values are found in a list.
Syntax: List.ContainsAll(list as list, values as list, optional equationCriteria as any) as logical
Example: List.ContainsAll({1, 2, 3, 4, 5}, {3, 4})
Answer: True

List.ContainsAny
Excel equivalent: OR
Returns true if any item in values is found in a list.
Syntax: List.ContainsAny(list as list, values as list, optional equationCriteria as any) as logical
Example: List.ContainsAny({1, 2, 3, 4, 5}, {3, 9})
Answer: true

List.Count
Excel equivalent: COUNT

Returns the number of items in a list.
Syntax: List.Count(list as list) as number
Example: List.Count({1, 2, 3})
Answer: 3

List.Covariance

Returns the covariance from two lists as a number.
Syntax: List.Covariance(numberList1 as list, numberList2 as list) as nullable number
Example: List.Covariance({1, 2, 3},{1, 2, 3})
Answer: 0.666666666667

List.Dates

Returns a list of date values from size count, starting at start and adds an increment to every value.
Syntax: List.Dates(start as date, count as number, step as duration) as list
Example: List.Dates(#date(2011, 12, 31), 5, #duration(1, 0, 0, 0))

List.DateTimes

Returns a list of datetime values from size count, starting at start and adds an increment to every value.
Syntax: List.DateTimes(start as datetime, count as number, step as duration) as list
Example: List.DateTimes(#datetime(2011, 12, 31, 23, 55, 0), 10, #duration(0, 0, 1, 0))

List.DateTimeZones

Returns a list of of datetimezone values from size count, starting at start and adds an increment to every value.
Syntax: List.DateTimeZones(start as datetimezone, count as number, step as duration) as list
Example: List.DateTimeZones(#datetimezone(2011, 12, 31, 23, 55, 0, -8, 0), 10, #duration(0, 0, 1, 0))

List.Difference

Returns the items in list 1 that do not appear in list 2. Duplicate values are supported.
Syntax: List.Difference(list1 as list, list2 as list, optional equationCriteria as any) as list
Example: List.Difference({1, 2, 3, 4, 5},{4, 5, 3})
Answer: {1, 2}
Example: List.Difference({1, 2}, {1, 2, 3})

List.Distinct

Filters a list down by removing duplicates. An optional equation criteria value can be specified to control equality comparison. The first value from each equality group is chosen.
Syntax: List.Distinct(list as list, optional equationCriteria as any) as list
Example: List.Distinct({1, 1, 2, 3, 3, 3})

List.Durations

Returns a list of durations values from size count, starting at start and adds an increment to every value.
Syntax: List.Durations(start as duration, count as number, step as duration) as list
Example: List.Durations(#duration(0, 1, 0, 0), 5, #duration(0, 1, 0, 0))

List.FindText

Searches a list of values, including record fields, for a text value.
Syntax: List.FindText(list as list, text as text) as list
Example: List.FindText({"a", "b", "ab"}, "a")

Answer: {"a", "ab"}

List.First

Returns the first value of the list or the specified default if empty. Returns the first item in the list, or the optional default value, if the list is empty. If the list is empty and a default value is not specified, the function returns.

Syntax: List.First(list as list, optional defaultValue as any) as any

Example: List.First({1, 2, 3})

Answer: 1

Example: List.First({}, -1)

Answer: -1

List.FirstN

Returns the first set of items in the list by specifying how many items to return or a qualifying condition provided by countOrCondition.

Syntax: List.FirstN(list as list, countOrCondition as any) as any

Example: List.FirstN({3, 4, 5, -1, 7, 8, 2},each _ > 0)

Answer: {3, 4, 5}

List.Generate

Generates a list from a value function, a condition function, a next function, and an optional transformation function on the values.

Syntax: List.Generate(initial as function, condition as function, next as function, optional selector as nullable function) as list

Example: List.Generate(()=>10, each _ > 0, each _ - 1)

List.InsertRange

Inserts items from values at the given index in the input list.

Syntax: List.InsertRange(list as list, index as number, values as list) as list

Example: List.InsertRange({1, 2, 5}, 2, {3, 4})

Answer: {1, 2, 3, 4, 5}

List.Intersect

Returns a list from a list of lists and intersects common items in individual lists. Duplicate values are supported.

Syntax: List.Intersect(lists as list, optional equationCriteria as any) as list

Example: List.Intersect({{1..5}, {2..6}, {3..7}})

Answer: {3, 4, 5}

List.IsDistinct

Returns whether a list is distinct.

Syntax: List.IsDistinct(list as list, optional equationCriteria as any) as logical

Example: List.IsDistinct({1, 2, 3})

List.IsEmpty

Returns whether a list is empty.

Syntax: List.IsEmpty(list as list) as logical

Example: List.IsEmpty({})

List.Last
Returns the last set of items in the list by specifying how many items to return or a qualifying condition provided by countOrCondition.
Syntax: List.Last(list as list, optional defaultValue as any) as any
Example: List.Last({1, 2, 3})
Answer: 3
Example: List.Last({}, -1)
Answer: -1

List.LastN
Returns the last set of items in a list by specifying how many items to return or a qualifying condition.
Syntax: List.LastN(list as list, optional countOrCondition as any) as any
Example: List.LastN({3, 4, 5, -1, 7, 8, 2},1)
Answer: 2
Example: List.LastN({3, 4, 5, -1, 7, 8, 2}, each _ > 0)
Answer: {7, 8, 2}

List.MatchesAll
Excel equivalent: AND
Returns true if all items in a list meet a condition.
Syntax: List.MatchesAll(list as list, condition as function) as logical
Example: List.MatchesAll({11, 12, 13},each _ > 10)
Answer: true
Example: List.MatchesAll({1, 2, 3},each _ > 10)
Answer: false

List.MatchesAny
Excel equivalent: OR
Returns true if any item in a list meets a condition.
Syntax: List.MatchesAny(list as list, condition as function) as logical
Example: List.MatchesAny({9, 10, 11},each _ > 10)
Answer: true
Example: List.MatchesAny({1, 2, 3},each _ > 10)
Answer: false

List.Max
Excel equivalent: MAX
Returns the maximum item in a list, or the optional default value if the list is empty.
Syntax: List.Max(list as list, optional default as any, optional comparisonCriteria as any, optional includeNulls as nullable logical) as any
Example: List.Max({1, 4, 7, 3, -2, 5},1)
Answer: 7
Example: List.Max({}, -1)
Answer: -1

List.MaxN
Excel equivalent: MAX
Returns the maximum values in the list. After the rows are sorted, optional parameters may be specified to further filter the result

Syntax: List.MaxN(list as list, countOrCondition as any, optional comparisonCriteria as any, optional includeNulls as nullable logical) as list

List.Median

Excel equivalent: MEDIAN
Returns the median item from a list.
Syntax: List.Median(list as list, optional comparisonCriteria as any) as any
Example: powerquery-mList.Median({5, 3, 1, 7, 9})
Answer: 5

List.Min

Excel equivalent: MIN
Returns the minimum item in a list, or the optional default value if the list is empty.
Syntax: List.Min(list as list, optional default as any, optional comparisonCriteria as any, optional includeNulls as nullable logical) as any
Example: List.Min({1, 4, 7, 3, -2, 5})
Answer: -2
Example: List.Min({}, -1)
Answer: -1

List.MinN

Excel equivalent: MIN
Returns the minimum values in a list.
Syntax: List.MinN(list as list, countOrCondition as any, optional comparisonCriteria as any, optional includeNulls as nullable logical) as list
Example: List.MinN({3, 4, 5, -1, 7, 8, 2}, 5)
Answer: {-1, 2, 3, 4, 5}

List.Mode

Excel equivalent: MODE
Returns an item that appears most commonly in a list.
Syntax: List.Mode(list as list, optional equationCriteria [what if there is more than one Mode] as any) as any
Example: List.Mode({"A", 1, 2, 3, 3, 4, 5})
Answer: 3
Example: List.Mode({"A", 1, 2, 3, 3, 4, 5, 5})
Answer: 5

List.Modes

Excel equivalent: MODE
Returns all items that appear with the same maximum frequency.
Syntax: List.Modes(list as list, optional equationCriteria as any) as list
Example: List.Modes({"A", 1, 2, 3, 3, 4, 5, 5})
Answer: {3, 5}

List.NonNullCount

Excel equivalent: COUNTA
Returns the number of items in a list excluding null values
Syntax: List.NonNullCount(list as list) as number

List.Numbers

Returns a list of numbers from size count starting at initial, and adds an increment. The increment defaults to 1.

Syntax: List.Numbers(start as number, count as number, optional increment as nullable number) as list

Example: List.Numbers(1, 10)

Example: List.Numbers(1, 10, 2)

Answer: {1, 3, 5…, 19}

List.PositionOf

Finds the first occurrence of a value in a list and returns its position.

Syntax: List.PositionOf(list as list, value as any, optional occurrence as nullable number, optional equationCriteria as any) as any

Example: List.PositionOf({1, 2, 3}, 3)

Answer: 2 [0-based]

Example: List.PositionOf({1, 2, 3}, 4)

Answer: -1

List.PositionOfAny

Finds the first occurrence of any value in values and returns its position.

Syntax: List.PositionOfAny(list as list, values as list, optional occurrence as nullable number, optional equationCriteria as any) as any

Example: List.PositionOfAny({1, 2, 3}, {2, 3})

Answer: 1 [0-based]

List.Positions

Returns a list of positions for an input list.

Syntax: List.Positions(list as list) as list

Example: List.Positions({1, 2, 3, 4, null, 5})

Answer: {0..5}

List.Product

Excel equivalent: PRODUCT

Returns the product from a list of numbers.

Syntax: List.Product(numbersList as list, optional precision as nullable number) as nullable number

Example: List.Product({1, 2, 3, 3, 4, 5, 5})

List.Random

Returns a list of count random numbers, with an optional seed parameter.

Syntax: List.Random(count as number, optional seed as nullable number) as list

Example: List.Random(3)

List.Range

Returns a count items starting at an offset.

Syntax: List.Range(list as list, offset as number, optional count as nullable number) as list

Example: List.Range({1..10}, 6)

Answer: {7.10}

List.RemoveFirstN

Returns a list with the specified number of elements removed from the list starting at the first element. The number of elements removed depends on the optional countOrCondition parameter.

Syntax: List.RemoveFirstN(list as list, optional countOrCondition as any) as list
Example: List.RemoveFirstN({1, 2, 3, 4, 5}, 3)
Example: List.RemoveFirstN({5, 4, 2, 6, 1}, each _ > 3)

List.RemoveItems
Removes items from list1 that are present in list2, and returns a new list.
Syntax: List.RemoveItems(list1 as list, list2 as list) as list
Example: List.RemoveItems({1, 2, 3, 4, 2, 5, 5}, {2, 4, 6})

List.RemoveLastN
Returns a list with the specified number of elements removed from the list starting at the last element. The number of elements removed depends on the optional countOrCondition parameter.
Syntax: List.RemoveLastN(list as list, optional countOrCondition as any) as list
Example: List.RemoveLastN({1, 2, 3, 4, 5}, 3)

List.RemoveMatchingItems
Removes all occurrences of the given values in the list.
Syntax: List.RemoveMatchingItems(list1 as list, list2 as list, optional equationCriteria as any) as list
Example: List.RemoveMatchingItems({1, 2, 3, 4, 5, 5}, {1, 5})

List.RemoveNulls
Removes null values from a list.
Syntax: List.RemoveNulls(list as list) as list
Example: List.RemoveNulls({1, 2, 3, null, 4, 5, null, 6})

List.RemoveRange
Returns a list that removes count items starting at offset. The default count is 1.
Syntax: List.RemoveRange(list as list, index as number, optional count as nullable number) as list
Example: List.RemoveRange({1, 2, 3, 4, -6, -2, -1, 5}, 4, 3)
Answer: {1..5}

List.Repeat
Returns a list that repeats the contents of an input list count times.
Syntax: List.Repeat(list as list, count as number) as list
Example: List.Repeat({1, 2}, 3)

List.ReplaceMatchingItems
Replaces occurrences of existing values in the list with new values using the provided equationCriteria. Old and new values are provided by the replacements parameters. An optional equation criteria value can be specified to control equality comparisons. For details of replacement operations and equation criteria, see Parameter Values.
Syntax: ist.ReplaceMatchingItems(list as list, replacements as list, optional equationCriteria as any) as list
Example: List.ReplaceMatchingItems({1, 2, 3, 4, 5}, {{5, -5}, {1, -1}})

List.ReplaceRange
Returns a list that replaces count values in a list with a replaceWith list starting at an index.
Syntax: List.ReplaceRange(list as list, index as number, count as number, replaceWith as list) as list
Example: List.ReplaceRange({1, 2, 7, 8, 9, 5}, 2, 3, {3, 4})
Answer: {1..5}

List.ReplaceValue
Searches a list of values for the value and replaces each occurrence with the replacement value.
Syntax: List.ReplaceValue(list as list, oldValue as any, newValue as any, replacer as function) as list
Example: List.ReplaceValue({"a", "B", "a", "a"}, "a", "A", Replacer.ReplaceText)

List.Reverse
Returns a list that reverses the items in a list.
Syntax: List.Reverse(list as list) as list
Example: List.Reverse({1..10})

List.Select
Selects the items that match a condition.
Syntax: List.Select(list as list, selection as function) as list
Example: List.Select({1, -3, 4, 9, -2}, each _ > 0)

List.Single
Returns the single item of the list or throws an Expression.Error if the list has more than one item.
Syntax: List.Single(list as list) as any
Example: List.Single({1})
Answer: 1
Example: List.Single({1, 2, 3})
Answer: [Expression.Error] There were too many elements in the enumeration to complete the operation.

List.SingleOrDefault
Returns a single item from a list.
Syntax: List.SingleOrDefault(list as list, optional default as any) as any
Example: List.SingleOrDefault({1})
Answer: 1 (other answers = null or default)

List.Skip
Skips the first item of the list. Given an empty list, it returns an empty list. This function takes an optional parameter countOrCondition to support skipping multiple values.
Syntax: List.Skip(list as list, optional countOrCondition as any) as list
Example: List.Skip({1, 2, 3, 4, 5}, 3)
Example: List.Skip({5, 4, 2, 6, 1}, each _ > 3)

List.Sort
Returns a sorted list using comparison criterion.
Syntax: List.Sort(list as list, optional comparisonCriteria as any) as list
Example: List.Sort({2, 3, 1}, Order.Descending)
Example: List.Sort({2, 3, 1}, (x, y) => Value.Compare(1/x, 1/y))

List.Split
Splits the specified list into a list of lists using the specified page size.
Syntax: List.Split(list as list, pageSize as number) as list

List.StandardDeviation
Returns the standard deviation from a list of values. List.StandardDeviation performs a sample based estimate. The result is a number for numbers, and a duration for DateTimes and Durations.
Syntax: List.StandardDeviation(numbersList as list) as nullable number

Example: List.StandardDeviation({1..5})

List.Sum
Excel equivalent: SUM
Returns the sum from a list.
Syntax: List.Sum(list as list, optional precision as nullable number) as any
Example: List.Sum({1, 2, 3})
Answer: 6

List.Times
Returns a list of time values of size count, starting at start.
Syntax: List.Times(start as time, count as number, step as duration) as list
Example: List.Times(#time(12, 0, 0), 4, #duration(0, 1, 0, 0))

List.Transform
Performs the function on each item in the list and returns the new list.
Syntax: List.Transform(list as list, transform as function) as list
Example: List.Transform({1, 2}, each _ + 1)

List.TransformMany
Returns a list whose elements are projected from the input list.
Syntax: List.TransformMany(list as list, collectionTransform as function, resultTransform as function) as list

List.Union
Returns a list from a list of lists and unions the items in the individual lists. The returned list contains all items in any input lists. Duplicate values are matched as part of the Union.
Syntax: List.Union(lists as list, optional equationCriteria as any) as list
Example: List.Union({ {1..5}, {2..6}, {3..7} })
Answer: {1.7}

List.Zip
Returns a list of lists combining items at the same position.
Syntax: List.Zip(lists as list) as list
Example: List.Zip({{1, 2}, {3, 4}})
Example: List.Zip({{1, 2}, {3}})

Logical.From
Returns a logical value from a value.
Syntax: Logical.From(value as any) as nullable logical
Example: Logical.From(2)
Answer: true

Logical.FromText
Returns a logical value of true or false from a text value.
Syntax: Logical.FromText(text as nullable text) as nullable logical
Example: Logical.FromText("true")
Example: Logical.FromText("a")
Answer: [Expression.Error] Could not convert to a logical.

Logical.ToText
Returns a text value from a logical value.
Syntax: Logical.ToText(logicalValue as nullable logical) as nullable text
Example: Logical.ToText(true)
Answer: TRUE

MissingField.Error
An optional parameter in record and table functions indicating that missing fields should result in an error. (This is the default parameter value.)
Value: 0

MissingField.Ignore
An optional parameter in record and table functions indicating that missing fields should be ignored.
Value: 1

MissingField.UseNull
An optional parameter in record and table functions indicating that missing fields should be included as null values.
Value: 2

MySQL.Database
Returns a table with data relating to the tables in the specified MySQL Database.
Syntax: MySQL.Database(server as text, database as text, optional options as nullable record) as table

Number.Abs
Excel equivalent: ABS
Returns the absolute value of a number.
Syntax: Number.Abs(number as nullable number) as nullable number
Example: Number.Abs(-9)

Number.Acos
Excel equivalent: ACOS
Returns the arccosine of a number.
Syntax: Number.Acos(number as nullable number) as nullable number

Number.Asin
Excel equivalent: ASIN
Returns the arcsine of a number.
Syntax: Number.Asin(number as nullable number) as nullable number

Number.Atan
Excel equivalent: ATAN
Returns the arctangent of a number.
Syntax: Number.Atan(number as nullable number) as nullable number

Number.Atan2
Excel equivalent: ATAN2
Returns the arctangent of the division of two numbers.
Syntax: Number.Atan2(y as nullable number, x as nullable number) as nullable number

Number.BitwiseAnd

Excel equivalent: BITAND
Returns the result of a bitwise AND operation on the provided operands.
Syntax: Number.BitwiseAnd(number1 as nullable number, number2 as nullable number) as nullable number

Number.BitwiseNot

Excel equivalent: Bitwise
Returns the result of a bitwise NOT operation on the provided operands.
Syntax: Number.BitwiseNot(number as any) as any

Number.BitwiseOr

Excel equivalent: BITOR
Returns the result of a bitwise OR operation on the provided operands.
Syntax: Number.BitwiseOr(number1 as nullable number, number2 as nullable number) as nullable number

Number.BitwiseShiftLeft

Excel equivalent: BITLSHIFT
Returns the result of a bitwise shift left operation on the operands.
Syntax: Number.BitwiseShiftLeft(number1 as nullable number, number2 as nullable number) as nullable number

Number.BitwiseShiftRight

Excel equivalent: BITRSHIFT
Returns the result of a bitwise shift right operation on the operands.
Syntax: Number.BitwiseShiftRight(number1 as nullable number, number2 as nullable number) as nullable number

Number.BitwiseXor

Excel equivalent: BITXOR
Returns the result of a bitwise XOR operation on the provided operands.
Syntax: Number.BitwiseXor(number1 as nullable number, number2 as nullable number) as nullable number

Number.Combinations

Excel equivalent: Miscellaneous
Returns the number of combinations of a given number of items for the optional combination size.
Syntax: Number.Combinations(setSize as nullable number, combinationSize as nullable number) as nullable number
Example: Number.Combinations(5, 3)

Number.Cos

Excel equivalent: Trigonometric Function
Returns the cosine of a number.
Syntax: Number.Cos(number as nullable number) as nullable number
Example: Number.Cos(0)

Number.Cosh

Excel equivalent: Trigonometric Function
Returns the hyperbolic cosine of a number.

Syntax: Number.Cosh(number as nullable number) as nullable number

Number.E

Returns 2.7182818284590451, the value of e up to 16 decimal digits.

Number.Epsilon

Returns the smallest possible number.

Value: 0

Number.Exp

Excel equivalent: EXP

Returns a number representing e raised to a power.

Syntax: Number.Exp(number as nullable number) as nullable number

Example: E (2.7182818…)

Number.Factorial

Excel equivalent: FACT

Returns the factorial of a number.

Syntax: Number.Factorial(number as nullable number) as nullable number

Example: Number.Factorial(10)

Number.From

Excel equivalent: Converts to Number from Number, Text, Logical (0 or 1), datetime, datetimezone, date, time and duration

Returns a number value from a value.

Syntax: Number.From(value as any, optional culture as nullable text) as nullable number

Example: Number.From("5")

Number.From(#datetime(2021, 4, 21, 7, 1, 2))

Number.From("25.6%")

Number.FromText

Returns a number value from a text value.

Syntax: Number.FromText(text as nullable text, optional culture as nullable text) as nullable number

Example: Number.FromText("5.0e-10")

Number.IntegerDivide

Divides two numbers and returns the whole part of the resulting number.

Syntax: Number.IntegerDivide(number1 as nullable number, number2 as nullable number, optional precision as nullable number) as nullable number

Example: Number.IntegerDivide(8.3, 3)   = 2

Number.IsEven

Returns true if a value is an even number.

Syntax: Number.IsEven(number as number) as logical

Example: Number.IsEven(625)

Answer: False

Number.IsNaN

Returns true if a value is Number.NaN.

Syntax: Number.IsNaN(number as number) as logical

Example: Number.IsNaN(number as number) as logical

Answer: True

Number.IsOdd
Returns true if a value is an odd number.
Syntax: Number.IsOdd(number as number) as logical
Example: Number.IsOdd(625)
Answer: True

Number.Ln
Excel equivalent: LN
Returns the natural logarithm of a number.
Syntax: Number.Ln(number as nullable number) as nullable number

Number.Log
Excel equivalent: LOG
Returns the logarithm of a number to the base.
Syntax: Number.Log(number as nullable number, optional base as nullable number [Number.Exp]) as nullable number
Example: Number.Log(2, 10)
Answer: 0.3010299956639812

Number.Log10
Excel equivalent: LOG10
Returns the base-10 logarithm of a number.
Syntax: Number.Log10(number as nullable number) as nullable number
Example: Number.Log10(2)

Number.Mod
Excel equivalent: MOD
Divides two numbers and returns the remainder of the resulting number.
Syntax: Number.Mod(number as nullable number, divisor as nullable number, optional precision as nullable number) as nullable number
Example: Number.Mod(5, 3)

Number.NaN
Excel equivalent: 0 divided by 0
Represents 0/0.

Number.NegativeInfinity
Excel equivalent: -1 divided by 0
Represents -1/0.

Number.Permutations
Returns the number of total permutatons of a given number of items for the optional permutation size.
Syntax: Number.Permutations(setSize as nullable number, permutationSize as nullable number) as nullable number
Example: Number.Permutations(5, 3)

Number.PI
Excel equivalent: PI (3.1415)

Returns 3.1415926535897931, the value for Pi up to 16 decimal digits.

Number.PositiveInfinity
Excel equivalent: 1 divided by 0
Represents 1/0.

Number.Power
Excel equivalent: POWER
Returns a number raised by a power.
Syntax: Number.Power(number as nullable number, power as nullable number) as nullable number
Example: Number.Power(5, 3)

Number.Random
Excel equivalent: RAND()
Returns a random fractional number between 0 and 1.
Syntax: Number.Random()
Example: 0.919303

Number.RandomBetween
Excel equivalent: RANDBETWEEN
Returns a random number between the two given number values.
Syntax: Number.RandomBetween(bottom as number, top as number) as number
Example: 2.546797

Number.Round
Excel equivalent: ROUND
Returns a nullable number (n) if value is an integer.
Syntax: Number.Round(number as nullable number, optional digits as nullable number [0], optional roundingMode as nullable number) as nullable number
Example: Number.Round(1.2345, 3, RoundingMode.Up) Number.Round(1.2345, 3, RoundingMode.Down)

Number.RoundAwayFromZero
Excel equivalent: ROUNDUP
Returns Number.RoundUp(value) when value >= 0 and Number.RoundDown(value) when value < 0.
Syntax: Number.RoundAwayFromZero(number as nullable number [0], optional digits as nullable number) as nullable number
Example: Number.RoundAwayFromZero(-1.234, 2)

Number.RoundDown
Excel equivalent: FLOOR
Returns the largest integer less than or equal to a number value.
Syntax: Number.RoundDown(number as nullable number, optional digits as nullable number) as nullable number
Example: Number.RoundDown(1.999, 2)
Answer: 1.99

Number.RoundTowardZero
Excel equivalent: ROUNDDOWN
Returns Number.RoundDown(x) when x >= 0 and Number.RoundUp(x) when x < 0.

Syntax: Number.RoundTowardZero(number as nullable number, optional digits as nullable number) as nullable number

Number.RoundUp

Excel equivalent: CEILING
Returns the larger integer greater than or equal to a number value.
Syntax: Number.RoundUp(number as nullable number, optional digits as nullable number) as nullable number
Example: Number.RoundUp(1.234)
Example: Number.RoundUp(1.999)
Example: Number.RoundUp(1.234, 2)

Number.Sign

Excel equivalent: SIGN
Returns 1 for positive numbers, -1 for negative numbers or 0 for zero.
Syntax: Number.Sign(number as nullable number) as nullable number
Example: Number.Sign(182)
Example: Number.Sign(-182)
Example: Number.Sign(0)

Number.Sin

Excel equivalent: SIN
Returns the sine of a number.
Syntax: Number.Sin(number as nullable number) as nullable number
Example: Number.Sin(0)

Number.Sinh

Excel equivalent: SINH
Returns the hyperbolic sine of a number.
Syntax: Number.Sinh(number as nullable number) as nullable number

Number.Sqrt

Excel equivalent: SQRT
Returns the square root of a number.
Syntax: Number.Sqrt(number as nullable number) as nullable number
Example: Number.Sqrt(625)
Example: Number.Sqrt(85)
Answer: 9.21954445729288

Number.Tan

Excel equivalent: TAN
Returns the tangent of a number.
Syntax: Number.Tan(number as nullable number) as nullable number
Example: Number.Tan(1)
Answer: 1.5574077246549

Number.Tanh

Excel equivalent: TANH
Returns the hyperbolic tangent of a number.
Syntax: Number.Tanh(number as nullable number) as nullable number

Number.ToText
Returns a text value from a number value.
Syntax: Number.ToText(number as nullable number, optional format as nullable text, optional culture as nullable text) as nullable text
Example: Format = D (decimal), E (exponential/scientific), F (fixed-point), G (General), N (#,###.## Number), P (Percent), R (round-trip), X (hexadecimal). Followed by a number precision specifier (e.g. P1).

Occurrence.All
A list of positions of all occurrences of the found values is returned.
Value: 2

Occurrence.First
The position of the first occurrence of the found value is returned.
Value: 0

Occurrence.Last
The position of the last occurrence of the found value is returned.
Value: 1

Occurrence.Optional
The item is expected to appear zero or one time in the input.
Value: 0

Occurrence.Repeating
The item is expected to appear zero or more times in the input.
Value: 2

Occurrence.Required
The item is expected to appear once in the input.
Value: 1

OData.Feed
Returns a table of OData feeds offered by an OData serviceUri.
Syntax: OData.Feed(serviceUri as text, optional headers as nullable record, optional options as any) as

ODataOmitValues.Nulls
Allows the OData service to omit null values.

Odbc.DataSource
Returns a table of SQL tables and views from the ODBC data source specified by the connection string connectionString.
Syntax: Odbc.DataSource(connectionString as any, optional options as nullable record) as table

Odbc.InferOptions
Returns the result of trying to infer SQL capabilities for an ODBC driver.
Syntax: Odbc.InferOptions(connectionString as any) as record

Odbc.Query
Connects to a generic provider with the given connection string and returns the result of evaluating the query.
Syntax: Odbc.Query(connectionString as any, query as text, optional options as nullable record) as table

OleDb.DataSource
Returns a table of SQL tables and views from the OLE DB data source specified by the connection string.
Syntax: OleDb.DataSource(connectionString as any, optional options as nullable record) as table

OleDb.Query
Returns the result of running a native query on an OLE DB data source.
Syntax: OleDb.Query(connectionString as any, query as text, optional options as nullable record) as table

Oracle.Database
Returns a table with data relating to the tables in the specified Oracle Database.
Syntax: Oracle.Database(server as text, optional options as nullable record) as table

Order.Ascending
Value: 0

Order.Descending
Value: 1

Percentage.From
Returns a percentage value from the given value.
Syntax: Percentage.From(value as any, optional culture as nullable text) as nullable number
Example: Percentage.From("12.3%")
Answer: 0.123

PostgreSQL.Database
Returns a table with data relating to the tables in the specified PostgreSQL Database.
Syntax: PostgreSQL.Database(server as text, database as text, optional options as nullable record) as

Precision.Decimal
An optional parameter for the built-in arthimetic operators to specify decimal precision.
Value: 1

Precision.Double
An optional parameter for the built-in arthimetic operators to specify double precision.
Value: 0

QuoteStyle.Csv
Quote characters indicate the start of a quoted string. Nested quotes are indicated by two quote characters.
Value: 1

QuoteStyle.None
Quote characters have no significance.
Value: 0

RData.FromBinary
Returns a record of data frames from the RData file.
Syntax: RData.FromBinary(stream as binary) as any

Record.AddField
Adds a field from a field name and value.
Syntax: Record.AddField(record as record, fieldName as text, value as any, optional delayed as nullable
Example: Record.AddField([CustomerID = 1, Name = "Bob", Phone = "123-4567"], "Address", "123 Main St.")
Answer: CUSTOMERID 1

Record.Combine
Combines the records in a list.
Syntax: Record.Combine(records as list) as record
Example: Record.Combine({ [CustomerID =1, Name ="Bob"] , [Phone = "123-4567"]})
Answer: CUSTOMERID 1

Record.Field
Returns the value of the given field. This function can be used to dynamically create field lookup syntax for a given record. In that way it is a dynamic verison of the record[field] syntax.
Syntax: Record.Field(record as record, field as text) as any
Example: Record.Field([CustomerID = 1, Name = "Bob", Phone = "123-4567"], "CustomerID")

Record.FieldCount
Returns the number of fields in a record.
Syntax: Record.FieldCount(record as record) as number
Example: Record.FieldCount([CustomerID = 1, Name = "Bob"])

Record.FieldNames
Returns a list of field names in order of the record's fields.
Syntax: Record.FieldNames(record as record) as list
Example: Record.FieldNames([OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0])

Record.FieldOrDefault
Returns the value of a field from a record, or the default value if the field does not exist.
Syntax: Record.FieldOrDefault(record as nullable record, field as text, optional defaultValue as any) as
Example: Record.FieldOrDefault([CustomerID =1, Name="Bob"], "Phone")
Example: Record.FieldOrDefault([CustomerID =1, Name="Bob"], "Phone", "123-4567")

Record.FieldValues
Returns a list of field values in order of the record's fields.
Syntax: Record.FieldValues(record as record) as list
Example: Record.FieldValues([CustomerID = 1, Name = "Bob", Phone = "123-4567"])

Record.FromList

Returns a record given a list of field values and a set of fields.

Syntax: Record.FromList(list as list, fields as any) as record

Example: Record.FromList({1, "Bob", "123-4567"}, {"CustomerID", "Name", "Phone"})

Record.FromTable

Returns a record from a table of records containing field names and values.

Syntax: Record.FromTable(table as table) as record

Example: Record.FromTable(Table.FromRecords({[Name = "CustomerID", Value = 1], [Name = "Name", Value = "Bob"], [Name = "Phone", Value = "123-4567"]}))

Record.HasFields

Returns true if the field name or field names are present in a record.

Syntax: Record.HasFields(record as record, fields as any) as logical

Example: Record.HasFields([CustomerID = 1, Name = "Bob", Phone = "123-4567"],"CustomerID")

Example: Record.HasFields([CustomerID = 1, Name = "Bob", Phone = "123-4567"],{"CustomerID", "Address"})

Record.RemoveFields

Returns a new record that reorders the given fields with respect to each other. Any fields not specified remain in their original locations.

Syntax: Record.RemoveFields(record as record, fields as any, optional missingField as nullable number) as

Example: Record.RemoveFields([CustomerID=1, Item = "Fishing rod", Price=18.00], "Price")

Answer: CUSTOMERID 1

Example: Record.RemoveFields([CustomerID=1, Item = "Fishing rod", Price=18.00], {"Price", "Item"})

Answer: CUSTOMERID 1

Record.RenameFields

Returns a new record that renames the fields specified. The resultant fields will retain their original order. This function supports swapping and chaining field names. However, all target names plus remaining field names must constitute a unique set or an error will occur.

Syntax: Record.RenameFields(record as record, renames as list, optional missingField as nullable number)

Example: Record.RenameFields([OrderID = 1, CustomerID = 1, Item = "Fishing rod", UnitPrice = 100.0],

Answer: {"UnitPrice","Price"})

Example: Record.RenameFields([OrderNum = 1, CustomerID = 1, Item = "Fishing rod", UnitPrice = 100.0], {{"UnitPrice",

Answer: "Price"}, {"OrderNum", "OrderID"}})

Record.ReorderFields

Returns a new record that reorders fields relative to each other. Any fields not specified remain in their original locations. Requires two or more fields.

Syntax: Record.ReorderFields(record as record, fieldOrder as list, optional missingField as nullable

Example: Record.ReorderFields([CustomerID= 1, OrderID = 1, Item = "Fishing rod", Price = 100.0], {"OrderID",

Answer: "CustomerID"})

Record.SelectFields

Returns a new record that contains the fields selected from the input record. The original order of the fields is maintained.

Syntax: Record.SelectFields(record as record, fields as any, optional missingField as nullable number) as

Example: Record.SelectFields([OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0] , {"Item", "Price"})

Answer: ITEM Fishing rod

Record.ToList

Returns a list of values containing the field values of the input record.

Syntax: Record.ToList([A = 1, B = 2, C = 3])

Example: Record.ToList([A = 1, B = 2, C = 3])

Record.ToTable

Returns a table of records containing field names and values from an input record.

Syntax: Record.ToTable(record as record) as table

Example: Record.ToTable([OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0])

Answer: NAME VALUE

Record.TransformFields

Transforms fields by applying transformOperations. For more more information about values supported by transformOperations, see Parameter Values.

Syntax: Record.TransformFields(record as record, transformOperations as list, optional missingField as nullable number) as record

Example: Record.TransformFields([OrderID ="1", CustomerID= 1, Item = "Fishing rod", Price = "100.0"], {{"OrderID", Number.FromText}, {"Price",Number.FromText}})

RelativePosition.FromEnd

Indicates indexing should be done from the end of the input.

Value: 1

RelativePosition.FromStart

Indicates indexing should be done from the start of the input.

Value: 0

Replacer.ReplaceText

Excel equivalent: SUBSTITUTE

This function be provided to List.ReplaceValue or Table.ReplaceValue to do replace of text values in list and table values respectively.

Syntax: Replacer.ReplaceText(text as nullable text, old as text, new as text) as nullable text

Example: Replacer.ReplaceText("hEllo world","hE","He")

Replacer.ReplaceValue

Excel equivalent: SUBSTITUTE for numbers

This function be provided to List.ReplaceValue or Table.ReplaceValue to do replace values in list and table values respectively.

Syntax: Replacer.ReplaceValue(value as any, old as any, new as any) as any

Example: Replacer.ReplaceValue(11, 11, 10)

Resource.Access

RoundingMode.AwayFromZero
RoundingMode.AwayFromZero
Value: 2

RoundingMode.Down
RoundingMode.Down
Value: 1

RoundingMode.ToEven
RoundingMode.ToEven
Value: 4

RoundingMode.TowardZero
RoundingMode.TowardZero
Value: 3

RoundingMode.Up
RoundingMode.Up
Value: 0

RowExpression.Column
Returns an AST that represents access to a column within a row expression.
Syntax: RowExpression.Column(columnName as text) as record
Example: KIND FieldAccess
Answer: EXPRESSION [Record]

RowExpression.From
Returns the AST for the body of a function.
Syntax: RowExpression.From(function as function) as record
Example: RowExpression.From(each [CustomerName] = "ALFKI")
Answer: KIND Binary

RowExpression.Row
An AST node representing the row in a row expression.

Salesforce.Data
Connects to the Salesforce Objects API and returns the set of available objects (i.e. Accounts).
Syntax: Salesforce.Data(optional loginUrl as any, optional options as nullable record) as table

Salesforce.Reports
Connects to the Salesforce Reports API and returns the set of available reports.
Syntax: Salesforce.Reports(optional loginUrl as nullable text, optional options as nullable record) as

SapBusinessWarehouse.Cubes
Returns the InfoCubes and queries in an SAP Business Warehouse system grouped by InfoArea.
Syntax: SapBusinessWarehouse.Cubes(server as text, systemNumberOrSystemId as text, clientId as text,

SapBusinessWarehouseExecutionMode.BasXml
'bXML flattening mode' option for MDX execution in SAP Business Warehouse.
Value: 64

SapBusinessWarehouseExecutionMode.BasXmlGzip
'Gzip compressed bXML flattening mode' option for MDX execution in SAP Business Warehouse. Recommended for low latency or high volume queries.
Value: 65

SapBusinessWarehouseExecutionMode.DataStream
'DataStream flattening mode' option for MDX execution in SAP Business Warehouse.
Value: 66

SapHana.Database
Returns the packages in an SAP HANA database.
Syntax: SapHana.Database(**server** as text, optional **options** as nullable record) as table

SapHanaDistribution.All
Returns the packages in an SAP HANA database.
Value: 3

SapHanaDistribution.Connection
'Connection' distribution option for SAP HANA.
Value: 1

SapHanaDistribution.Off
'Off' distribution option for SAP HANA.
Value: 0

SapHanaDistribution.Statement
'Statement' distribution option for SAP HANA.
Value: 2

SapHanaRangeOperator.Equals
'Equals' range operator for SAP HANA input parameters.
Value: 4

SapHanaRangeOperator.GreaterThan
'Greater than' range operator for SAP HANA input parameters.
Value: 0

SapHanaRangeOperator.GreaterThanOrEquals
'Greater than or equals' range operator for SAP HANA input parameters.
Value: 2

SapHanaRangeOperator.LessThan
'Less than' range operator for SAP HANA input parameters.
Value: 1

SapHanaRangeOperator.LessThanOrEquals
'Less than or equals' range operator for SAP HANA input parameters.
Value: 3

SapHanaRangeOperator.NotEquals
'Not equals' range operator for SAP HANA input parameters.
Value: 5

SharePoint.Contents
Returns a table containing a row for each folder and document found at the SharePoint site url. Each row contains properties of the folder or file and a link to its content.
Syntax: SharePoint.Contents(url as text, optional options as nullable record) as table

SharePoint.Files
Returns a table containing a row for each document found at the SharePoint site url, and subfolders. Each row contains properties of the folder or file and a link to its content.
Syntax: SharePoint.Files(url as text, optional options as nullable record) as table

SharePoint.Tables
Returns a table containing the result of a SharePoint List as an OData feed.
Syntax: SharePoint.Tables(url as text, optional options as nullable record) as table

Single.From
Returns a Single number value from the given value.
Syntax: Single.From(value as any, optional culture as nullable text) as nullable number
Example: Single.From("1.5")
Answer: 1.5

Soda.Feed
Returns the resulting table of a CSV file that can be accessed using the SODA 2.0 API. The URL must point to a valid SODA-compliant source that ends in a .csv extension.
Syntax: Soda.Feed(url as text) as table

Splitter.SplitByNothing
Returns a function that does no splitting, returning its argument as a single element list.
Syntax: Splitter.SplitByNothing() as function

Splitter.SplitTextByAnyDelimiter
Returns a function that splits text by any supported delimiter.
Syntax: Splitter.SplitTextByAnyDelimiter(delimiters as list, optional quoteStyle as nullable number,

Splitter.SplitTextByCharacterTransition
Returns a function that splits text into a list of text according to a transition from one kind of character to another.
Syntax: Splitter.SplitTextByCharacterTransition(before as anynonnull, after as anynonnull) as function

Splitter.SplitTextByDelimiter
Returns a function that will split text according to a delimiter.
Syntax: Splitter.SplitTextByDelimiter(delimiter as text, optional quoteStyle as nullable number) as

Splitter.SplitTextByEachDelimiter
Returns a function that splits text by each delimiter in turn.
Syntax: Splitter.SplitTextByEachDelimiter(delimiters as list, optional quoteStyle as nullable number,

Splitter.SplitTextByLengths
Returns a function that splits text according to the specified lengths.
Syntax: Splitter.SplitTextByLengths(lengths as list, optional startAtEnd as nullable logical) as function

Splitter.SplitTextByPositions
Returns a function that splits text according to the specified positions.
Syntax: Splitter.SplitTextByPositions(positions as list, optional startAtEnd as nullable logical) as

Splitter.SplitTextByRanges
Returns a function that splits text according to the specified ranges.
Syntax: Splitter.SplitTextByRanges(ranges as list, optional startAtEnd as nullable logical) as function

Splitter.SplitTextByRepeatedLengths
Returns a function that splits text into a list of text after the specified length repeatedly.
Syntax: Splitter.SplitTextByRepeatedLengths(length as number, optional startAtEnd as nullable logical) as

Splitter.SplitTextByWhitespace
Returns a function that splits text according to whitespace.
Syntax: Splitter.SplitTextByWhitespace(optional quoteStyle as nullable number) as function

Sql.Database
Returns a table containing SQL tables located on a SQL Server instance database.
Syntax: Sql.Database(server as text, database as text, optional options as nullable record) as table

Sql.Databases
Returns a table with references to databases located on a SQL Server instance. Returns a navigation table.
Syntax: Sql.Databases(server as text, optional options as nullable record) as table

SqlExpression.SchemaFrom
SqlExpression.SchemaFrom
Syntax: SqlExpression.SchemaFrom(schema as any) as any

SqlExpression.ToExpression
SqlExpression.ToExpression
Syntax: SqlExpression.ToExpression(sql as text, environment as record) as text

Sybase.Database
Returns a table with data relating to the tables in the specified Sybase Database.
Syntax: Sybase.Database(server as text, database as text, optional options as nullable record) as table

Table.AddColumn
Adds a column named newColumnName to a table.

Syntax: Table.AddColumn(table as table, newColumnName as text, columnGenerator as function, optional
Example: Table.AddColumn(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0, Shipping
Answer: = 10.00], [OrderID = 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0, Shipping = 15.00], [OrderID = 3,

Table.AddIndexColumn

Returns a table with a new column with a specific name that, for each row, contains an index of the row in the table.
Syntax: Table.AddIndexColumn(table as table, newColumnName as text, optional initialValue as nullable
Example: Table.AddIndexColumn(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name
Example: Table.AddIndexColumn(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name

Table.AddJoinColumn

Performs a nested join between table1 and table2 from specific columns and produces the join result as a newColumnName column for each row of table1.
Syntax: Table.AddJoinColumn(table1 as table, key1 as any, table2 as function, key2 as any, newColumnName
Example: Table.AddJoinColumn(Table.FromRecords({[saleID = 1, item = "Shirt"], [saleID = 2, item = "Hat"]}), "saleID",
Answer: () => Table.FromRecords({[saleID = 1, price = 20, stock = 1234], [saleID = 2, price = 10, stock = 5643]}),

Table.AddKey

Add a key to table.
Syntax: Table.AddKey(table as table, columns as list, isPrimary as logical) as table
Example: let tableType = type table [Id = Int32.Type, Name = text], table = Table.FromRecords({[Id = 1, Name = "Hello
Answer: There"], [Id = 2, Name = "Good Bye"]}), resultTable = Table.AddKey(table, {"Id"}, true) in resultTable

Table.AggregateTableColumn

Aggregates tables nested in a specific column into multiple columns containing aggregate values for those tables.
Syntax: Table.AggregateTableColumn(table as table, column as text, aggregations as list) as table
Example: [t.a] , the min and max of [t.b] , and the count of values in [t.a] .
Answer: Table.AggregateTableColumn(Table.FromRecords({[t = Table.FromRecords({[a=1, b=2, c=3], [a=2,b=4,c=6]}), b =

Table.AlternateRows

Returns a table containing an alternating pattern of the rows from a table.
Syntax: Table.AlternateRows(table as table, offset as number, skip as number, take as number) as table

Example: Table.AlternateRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"]}), 1, 1, 1)

Table.Buffer
Buffers a table into memory, isolating it from external changes during evaluation.
Syntax: Table.Buffer(table as table) as table

Table.Column
Returns the values from a column in a table.
Syntax: Table.Column(table as table, column as text) as list
Example: Table.Column(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.ColumnCount
Returns the number of columns in a table.
Syntax: Table.ColumnCount(table as table) as number
Example: Table.ColumnCount(Table.FromRecords({[CustomerID =1, Name ="Bob", Phone = "123-4567"],[CustomerID =2, Name
Answer: ="Jim", Phone = "987-6543"],[CustomerID =3, Name ="Paul", Phone = "543-7890"]}))

Table.ColumnNames
Returns the names of columns from a table.
Syntax: Table.ColumnNames(table as table) as list
Example: Table.ColumnNames(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4,

Table.ColumnsOfType
Returns a list with the names of the columns that match the specified types.
Syntax: Table.ColumnsOfType(table as table, listOfTypes as list) as list
Example: Table.ColumnsOfType(Table.FromRecords({[a=1,b="hello"]}, type table[a=Number.Type, b=Text.Type]), {type
Answer: number})

Table.Combine
Returns a table that is the result of merging a list of tables. The tables must all have the same row type structure.
Syntax: Table.Combine(tables as list, optional columns as any) as table
Example: Table.Combine({Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"]}),
Answer: Table.FromRecords({[CustomerID = 2, Name = "Jim", Phone = "987-6543"] }),Table.FromRecords({[CustomerID = 3,
Example: Table.Combine({Table.FromRecords({[Name="Bob",Phone="123-4567"]}), Table.FromRecords({[Fax="987-6543",
Answer: Phone="838-7171"] }),Table.FromRecords({[Cell = "543-7890"]})})
Example: Phone="838-7171"] }),Table.FromRecords({[Cell = "543-7890"]})}, {"CustomerID", "Name"})

Answer: CUSTOMERID NAME

Table.CombineColumns

Table.CombineColumns merges columns using a combiner function to produce a new column. Table.CombineColumns is the inverse of Table.SplitColumns.
Syntax: Table.CombineColumns(table as table, sourceColumns as list, combiner as function, column as text)

Table.Contains

Determines whether the a record appears as a row in the table.
Syntax: Table.Contains(table as table, row as record, optional equationCriteria as any) as logical
Example: Table.Contains(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =
Example: Table.Contains(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =
Example: Table.Contains(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.ContainsAll

Determines whether all of the specified records appear as rows in the table.
Syntax: Table.ContainsAll(table as table, rows as list, optional equationCriteria as any) as logical
Example: Table.ContainsAll(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4,
Example: Table.ContainsAll(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4,

Table.ContainsAny

Determines whether any of the specified records appear as rows in the table.
Syntax: Table.ContainsAny(table as table, rows as list, optional equationCriteria as any) as logical
Example: Table.ContainsAny(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {[a = 1, b = 2], [a = 3, b = 5]})
Answer: TRUE
Example: Table.ContainsAny(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {[a = 1, b = 3], [a = 3, b = 5]})
Answer: FALSE
Example: or [a = 3, b = 5] comparing only the column [a].
Answer: Table.ContainsAny(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {[a = 1, b = 3], [a = 3, b = 5]}, "a")

Table.DemoteHeaders

Demotes the header row down into the first row of a table.

Syntax: Table.DemoteHeaders(table as table) as table
Example: Table.DemoteHeaders(Table.FromRecords({[CustomerID=1, Name="Bob", Phone="123-4567"],[CustomerID=2, Name="Jim",
Answer: Phone="987-6543"]}))

Table.Distinct
Removes duplicate rows from a table, ensuring that all remaining rows are distinct.
Syntax: Table.Distinct(table as table, optional equationCriteria as any) as table
Example: Table.Distinct(Table.FromRecords({[a = "A", b = "a"], [a = "B", b = "b"], [a = "A", b = "a"]}))
Answer: A B
Example: ({[a = "A", b = "a"], [a = "B", b = "a"], [a = "A", b = "b"]})
Answer: Table.Distinct(Table.FromRecords({[a = "A", b = "a"], [a = "B", b = "a"], [a = "A", b = "b"]}), "b")

Table.DuplicateColumn
Duplicates a column with the specified name. Values and type are copied from the source column.
Syntax: Table.DuplicateColumn(table as table, columnName as text, newColumnName as text, optional

Table.ExpandListColumn
Given a column of lists in a table, create a copy of a row for each value in its list.
Syntax: Table.ExpandListColumn(table as table, column as text) as table
Example: Table.ExpandListColumn(Table.FromRecords({[Name= {"Bob", "Jim", "Paul"}, Discount = .15]}), "Name")
Answer: NAME DISCOUNT

Table.ExpandRecordColumn
Expands a column of records into columns with each of the values.
Syntax: Table.ExpandRecordColumn(table as table, column as text, fieldNames as list, optional
Example: Table.ExpandRecordColumn(Table.FromRecords({[a = [aa = 1, bb = 2, cc = 3], b = 2]}), "a", {"aa", "bb", "cc"})
Answer: AA BB CC B

Table.ExpandTableColumn
Expands a column of records or a column of tables into multiple columns in the containing table.
Syntax: Table.ExpandTableColumn(table as table, column as text, columnNames as list, optional
Example: [t.a] , [t.b] and [t.c] .
Answer: Table.ExpandTableColumn(Table.FromRecords({[t = Table.FromRecords({[a=1, b=2, c= 3],[a=2,b=4,c=6]}), b = 2]}),

Table.FillDown
Replaces null values in the specified column or columns of the table with the most recent non-null value in the column.
Syntax: Table.FillDown(table as table, columns as list) as table
Example: Table.FillDown(Table.FromRecords({[Place=1, Name="Bob"], [Place=null, Name="John"], [Place=2, Name="Brad"],
Answer: [Place=3, Name="Mark"], [Place=null, Name="Tom"], [Place=null, Name="Adam"]}), {"Place"})

Table.FillUp

Returns a table from the table specified where the value of the next cell is propagated to the null values cells above in the column specified.
Syntax: Table.FillUp(table as table, columns as list) as table
Example: Table.FillUp(Table.FromRecords({[Column1 = 1, Column2 = 2], [Column1 = 3, Column2 = null], [Column1 = 5,
Answer: Column2 = 3]}), {"Column2"})

Table.FilterWithDataTable

0
Syntax: Table.FilterWithDataTable(**table** as table, **dataTableIdentifier** as text) as any

Table.FindText

Returns a table containing only the rows that have the specified text within one of their cells or any part thereof.
Syntax: Table.FindText(table as table, text as text) as table
Example: Table.FindText(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.First

Returns the first row from a table.
Syntax: Table.First(table as table, optional default as any) as any
Example: Table.First(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"]}))
Example: Table.First(Table.FromRecords({}), [a = 0, b = 0])
Answer: A 0

Table.FirstN

Returns the first row(s) of a table, depending on the countOrCondition parameter.
Syntax: Table.FirstN(table as table, countOrCondition as any) as table
Example: Table.FirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"]}), 2)
Example: Table.FirstN(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4], [a = -5, b = -6]}), each [a] > 0)
Answer: A B

Table.FirstValue

Returns the first column of the first row of the table or a specified default value.
Syntax: Table.FirstValue(table as table, optional default as any) as any

Table.FromColumns

Returns a table from a list containing nested lists with the column names and values.
Syntax: Table.FromColumns(lists as list, optional columns as any) as table
Example: Table.FromColumns({ {1, "Bob", "123-4567"} , {2, "Jim", "987-6543"}, {3, "Paul", "543-7890"} })
Answer: COLUMN1 COLUMN2 COLUMN3
Example: Table.FromColumns({ {1, "Bob", "123-4567"} , {2, "Jim", "987-6543"}, {3, "Paul", "543-7890"}}, {"CustomerID",

Answer: "Name", "Phone"})
Example: Table.FromColumns({ {1, 2, 3}, {4, 5}, {6, 7, 8, 9} }, {"column1", "column2", "column3"})
Answer: COLUMN1 COLUMN2 COLUMN3

Table.FromList

Converts a list into a table by applying the specified splitting function to each item in the list.
Syntax: Table.FromList(list as list, optional splitter as nullable function, optional columns as any,
Example: Table.FromList({"a", "b", "c", "d"}, null, {"Letters"})
Example: Table.FromList({[CustomerID=1,Name="Bob"],[CustomerID=2,Name="Jim"]} ,
Record.FieldValues, {"CustomerID",
Answer: "Name"})

Table.FromPartitions

Returns a table that is the result of combining a set of partitioned tables into new columns. The type of the column can optionally be specified, the default is any.
Syntax: Table.FromPartitions(partitionColumn as text, partitions as list, optional
partitionColumnType as
Example: Table.FromPartitions("Year", { { 1994, Table.FromPartitions("Month", { { "Jan",
Table.FromPartitions("Day",
Answer: { { 1, #table({"Foo"},{{"Bar"}}) }, { 2, #table({"Foo"},{{"Bar"}}) } }) }, { "Feb",
Table.FromPartitions(

Table.FromRecords

Returns a table from a list of records.
Syntax: Table.FromRecords(records as list, optional columns as any, optional missingField as nullable
Example: Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID =
2, Name = "Jim", Phone =
Answer: "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"]})
Example: Table.ColumnsOfType(Table.FromRecords({[CustomerID=1, Name="Bob"]}, type
table[CustomerID=Number.Type,
Answer: Name=Text.Type]), {type number})

Table.FromRows

Creates a table from the list where each element of the list is a list that contains the column values for a single row.
Syntax: Table.FromRows(rows as list, optional columns as any) as table
Example: Table.FromRows({ {1, "Bob", "123-4567"},{2, "Jim", "987-6543"}},{"CustomerID", "Name",
"Phone"})
Answer: CUSTOMERID NAME PHONE
Example: text types.
Answer: Table.FromRows({{1, "Bob", "123-4567"}, {2, "Jim", "987-6543"}}, type table [CustomerID =
number, Name = text,

Table.FromValue

Returns a table with a column containing the provided value or list of values.
Syntax: Table.FromValue(value as any, optional options as nullable record) as table
Example: Table.FromValue(1)
Example: Table.FromValue({1, "Bob", "123-4567"})
Example: Table.FromValue(1, [DefaultColumnName = "MyValue"])

Table.FuzzyJoin

Joins the rows from the two tables that fuzzy match based on the given keys.
Syntax: Table.FuzzyJoin(table1 as table, key1 as any, table2 as table, key2 as any, optional joinKind as

Table.FuzzyNestedJoin

Performs a fuzzy join between tables on supplied columns and produces the join result in a new column.
Syntax: Table.FuzzyNestedJoin(table1 as table, key1 as any, table2 as table, key2 as any, newColumnName as

Table.Group

Groups table rows by the values of key columns for each row.
Syntax: Table.Group(table as table, key as any, aggregatedColumns as list, optional groupKind as nullable
Example: Table.Group(Table.FromRecords({[CustomerID= 1, price = 20], [CustomerID= 2, price = 10], [CustomerID= 2, price
Answer: = 20], [CustomerID= 1, price = 10], [CustomerID= 3, price = 20], [CustomerID= 3, price = 5]}), "CustomerID",

Table.HasColumns

Returns true if a table has the specified column or columns.
Syntax: Table.HasColumns(table as table, columns as any) as logical
Example: Table.HasColumns(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =
Example: Table.HasColumns(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.InsertRows

Returns a table with the list of rows inserted into the table at an index. Each row to insert must match the row type of the table..
Syntax: Table.InsertRows(table as table, offset as number, rows as list) as table
Example: Table.InsertRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"]}), 1, {[CustomerID = 3, Name = "Paul", Phone = "543-7890"]})
Example: Table.InsertRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"]}), 1, {[CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"],[CustomerID = 3, Name = "Paul", Phone = "543-7890"] })

Table.IsDistinct

Determines whether a table contains only distinct rows.
Syntax: Table.IsDistinct(table as table, optional comparisonCriteria as any) as logical
Example: Table.IsDistinct(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4, Name =

Example: Table.IsDistinct(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 5, Name =

Table.IsEmpty
Returns true if the table does not contain any rows.
Syntax: Table.IsEmpty(table as table) as logical
Example: Table.IsEmpty(Table.FromRecords({[CustomerID =1, Name ="Bob", Phone = "123-4567"],[CustomerID =2, Name ="Jim",
Answer: Phone = "987-6543"],[CustomerID =3, Name ="Paul", Phone = "543-7890"]}))
Example: Table.IsEmpty(Table.FromRecords({}))
Answer: TRUE

Table.Join
Joins the rows of table1 with the rows of table2 based on the equality of the values of the key columns selected by table1, key1 and table2, key2.
Syntax: Table.Join(table1 as table, key1 as any, table2 as table, key2 as any, optional joinKind as
Example: Table.Join
Answer: (Table.FromRecords({

Table.Keys
Returns a list of key column names from a table.
Syntax: Table.Keys(table as table) as list

Table.Last
Returns the last row of a table.
Syntax: Table.Last(table as table, optional default as any) as any
Example: Table.Last(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"]}))
Example: Table.Last(Table.FromRecords({}), [a = 0, b = 0])
Answer: A 0

Table.LastN
Returns the last row(s) from a table, depending on the countOrCondition parameter.
Syntax: Table.LastN(table as table, countOrCondition as any) as table
Example: Table.LastN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"]}), 2)
Example: Table.LastN(Table.FromRecords({[a = -1, b = -2], [a = 3, b = 4], [a = 5, b = 6]}), each _ [a] > 0)
Answer: A B

Table.MatchesAllRows
Returns true if all of the rows in a table meet a condition.
Syntax: Table.MatchesAllRows(table as table, condition as function) as logical
Example: Table.MatchesAllRows(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8]}), each Number.Mod([a], 2) = 0)
Example: Table.MatchesAllRows(Table.FromRecords({[a = 1, b = 2], [a = -3, b = 4]}), each _ = [a = 1, b = 2])
Answer: FALSE

Table.MatchesAnyRows

Returns true if any of the rows in a table meet a condition.
Syntax: Table.MatchesAnyRows(table as table, condition as function) as logical
Example: Table.MatchesAnyRows(Table.FromRecords({[a = 1, b = 4], [a = 3, b = 8]}), each Number.Mod([a], 2) = 0)
Answer: FALSE
Example: Table.MatchesAnyRows(Table.FromRecords({[a = 1, b = 2], [a = -3, b = 4]}), each _ = [a = 1, b = 2])
Answer: TRUE

Table.Max

Returns the largest row or rows from a table using a comparisonCriteria.
Syntax: Table.Max(table as table, comparisonCriteria as any, optional default as any) as any
Example: Table.Max(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8]}), "a")
Answer: A 6
Example: Table.Max(#table({"a"},{}), "a", -1)
Answer: -1

Table.MaxN

Returns the largest N rows from a table. After the rows are sorted, the countOrCondition parameter must be specified to further filter the result.
Syntax: Table.MaxN(table as table, comparisonCriteria as any, countOrCondition as any) as table
Example: Table.MaxN(Table.FromRecords({[a = 2, b = 4], [a = 0, b = 0], [a = 6, b = 2]}), "a", each [a] > 0)
Answer: A B
Example: Table.MaxN(Table.FromRecords({[a = 2, b = 4], [a = 8, b = 0], [a = 6, b = 2]}), "a", each [b] > 0)
Answer: Table.Min

Table.Min

Returns the smallest row or rows from a table using a comparisonCriteria.
Syntax: Table.Min(table as table, comparisonCriteria as any, optional default as any) as any
Example: Table.Min(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8]}), "a")
Answer: A 2
Example: Table.Min(#table({"a"},{}), "a", -1)

Table.MinN

Returns the smallest N rows in the given table. After the rows are sorted, the countOrCondition parameter must be specified to further filter the result.
Syntax: Table.MinN(table as table, comparisonCriteria as any, countOrCondition as any) as table
Example: Table.MinN(Table.FromRecords({[a = 2, b = 4], [a = 0, b = 0], [a = 6, b = 4]}), "a", each [a] < 3)
Answer: A B
Example: Table.MinN(Table.FromRecords({[a = 2, b = 4], [a = 8, b = 0], [a = 6, b = 2]}), "a", each [b] < 0)
Answer: Table.NestedJoin

Table.NestedJoin

Joins the rows of the tables based on the equality of the keys. The results are entered into a new column.

Syntax: Table.NestedJoin(table1 as table, key1 as any, table2 as any, key2 as any, newColumnName as text,

Table.Partition

Partitions the table into a list of groups number of tables, based on the value of the column of each row and a hash function. The hash function is applied to the value of the column of a row to obtain a hash value for the row. The hash value modulo groups determines in which of the returned tables the row will be placed.
Syntax: Table.Partition(table as table, column as text, groups as number, hash as function) as list

Table.PartitionValues

Returns information about how a table is partitioned.
Syntax: Table.Partition(table as table, column as text, groups as number, hash as function) as list
Example: using the value of the columns as the hash function.
Answer: Table.Partition(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), "a", 2,

Table.Pivot

Given a table and attribute column containing pivotValues, creates new columns for each of the pivot values and assigns them values from the valueColumn. An optional aggregationFunction can be provided to handle multiple occurrence of the same key value in the attribute column.
Syntax: Table.Pivot(table as table, pivotValues as list, attributeColumn as text, valueColumn as text,
Example: = "a", value = 2], [key = "y", attribute = "b", value = 4] })
Answer: and pivot them into their own column.
Example: = "c", value = 5], [key = "y", attribute = "a", value = 2], [key = "y", attribute = "b", value = 4] })
Answer: and pivot them into their own column. The attribute "c" for key "x" has multiple values associated with it, so use the

Table.PositionOf

Determines the position or positions of a row within a table.
Syntax: Table.PositionOf(table as table, row as record, optional occurrence as any, optional
Example: ({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]})
Answer: Table.PositionOf(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), [a = 2,
Example: Table.PositionOf(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), [a = 2,
Answer: b = 4], 1)
Example: ({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]})
Answer: Table.PositionOf(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), [a = 2,

Table.PositionOfAny

Determines the position or positions of any of the specified rows within the table.
Syntax: Table.PositionOfAny(table as table, rows as list, optional occurrence as nullable number, optional
Example: Table.PositionOfAny(Table.FromRecords({[a = 2, b = 4], [a = 1, b = 4], [a = 2, b = 4], [a = 1, b = 4]}), {[a =
Answer: 2, b = 4], [a = 6, b = 8]})
Example: Table.PositionOfAny(Table.FromRecords({[a = 2, b = 4], [a = 6, b = 8], [a = 2, b = 4], [a = 1, b = 4]}), {[a =

Answer: 2, b = 4], [a = 6, b = 8]}, Occurrence.All)

Table.PrefixColumns

Returns a table where the columns have all been prefixed with a text value.
Syntax: Table.PrefixColumns(table as table, prefix as text) as table
Example: Table.PrefixColumns(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"]}), "MyTable")
Answer: MYTABLE.CUSTOMERID MYTABLE.NAME MYTABLE.PHONE

Table.Profile

Returns a profile of the columns of a table.
Syntax: Table.Profile(table as table, optional additionalAggregates as nullable list) as table

Table.PromoteHeaders

Promotes the first row of the table into its header or column names.
Syntax: Table.PromoteHeaders(table as table, optional options as nullable record) as table
Example: Table.PromoteHeaders(Table.FromRecords({[Column1 = "CustomerID", Column2 = "Name", Column3 = #date(1980,1,1)],
Answer: [Column1 = 1, Column2 = "Bob", Column3 = #date(1980,1,1)]}))
Example: Table.PromoteHeaders(Table.FromRecords({[Rank = 1, Name = "Name", Date = #date(1980,1,1)],[Rank = 1, Name =
Answer: "Bob", Date = #date(1980,1,1)]}), [PromoteAllScalars = true, Culture = "en-US"])

Table.Range

Returns the specified number of rows from a table starting at an offset.
Syntax: Table.Range(table as table, offset as number, optional count as nullable number) as table
Example: Table.Range(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =
Example: Table.Range(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.RemoveColumns

Returns a table without a specific column or columns.
Syntax: Table.RemoveColumns(table as table, columns as any, optional missingField as nullable number) as
Example: Table.RemoveColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-4567"]}), "Phone")
Answer: CUSTOMERID NAME
Example: Table.RemoveColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-4567"]}), "Address")
Answer: [Expression.Error] The field 'Address' of the record was not found.

Table.RemoveFirstN

Returns a table with the specified number of rows removed from the table starting at the first row. The number of rows removed depends on the optional countOrCondition parameter.
Syntax: Table.RemoveFirstN(table as table, countOrCondition as any) as table

Example: Table.RemoveFirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name
Example: Table.RemoveFirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name
Example: Table.RemoveFirstN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,
Answer: Name = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4,

Table.RemoveLastN

Returns a table with the specified number of rows removed from the table starting at the last row. The number of rows removed depends on the optional countOrCondition parameter.
Syntax: Table.RemoveLastN(table as table, optional countOrCondition as any) as table
Example: Table.RemoveLastN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],[CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"],[CustomerID = 3, Name = "Paul", Phone = "543-7890"],[CustomerID = 4, Name =
Example: Table.RemoveLastN(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],[CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"],[CustomerID = 3, Name = "Paul", Phone = "543-7890"],[CustomerID = 4, Name =

Table.RemoveMatchingRows

Removes all occurrences of rows from a table.
Syntax: Table.RemoveMatchingRows(table as table, rows as list, optional equationCriteria as any) as table
Example: Table.RemoveMatchingRows(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4], [a = 1, b = 6]}), {[a = 1]}, "a")
Answer: A B

Table.RemoveRows

Returns a table with the specified number of rows removed from the table starting at an offset.
Syntax: Table.RemoveRows(table as table, offset as number, optional count as nullable number) as table
Example: Table.RemoveRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =
Example: Table.RemoveRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =
Example: Table.RemoveRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.RemoveRowsWithErrors

Returns a table with all rows removed from the table that contain an error in at least one of the cells in a row.

Syntax: Table.RemoveRowsWithErrors(table as table, optional columns as nullable list) as table

Example: Table.RemoveRowsWithErrors(Table.FromRecords({[Column1=...],[Column1=2], [Column1=3]}))

Table.RenameColumns

Returns a table with the columns renamed as specified.

Syntax: Table.RenameColumns(table as table, renames as list, optional missingField as nullable number) as

Example: Table.RenameColumns(Table.FromRecords({[CustomerNum=1, Name="Bob", Phone = "123-4567"]}), {"CustomerNum",

Answer: "CustomerID"})

Example: Table.RenameColumns(Table.FromRecords({[CustomerNum=1, Name="Bob", PhoneNum = "123-4567"]}), {{"CustomerNum",

Answer: "CustomerID"}, {"PhoneNum", "Phone"}})

Example: Table.RenameColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-4567"]}), {"NewCol",

Answer: "NewColumn"}, MissingField.Ignore)

Table.ReorderColumns

Returns a table with specific columns in an order relative to one another.

Syntax: Table.ReorderColumns(table as table, columnOrder as list, optional missingField as nullable

Example: Table.ReorderColumns(Table.FromRecords({[CustomerID=1, Phone = "123-4567", Name ="Bob"]}), {"Name","Phone"})

Answer: CUSTOMERID NAME PHONE

Example: Table.ReorderColumns(Table.FromRecords({[CustomerID=1, Name = "Bob", Phone = "123-4567"]}), {"Phone",

Answer: "Address"}, MissingField.Ignore)

Table.Repeat

Returns a table containing the rows of the table repeated the count number of times.

Syntax: Table.Repeat(table as table, count as number) as table

Example: Table.Repeat(Table.FromRecords({[a = 1, b = "hello"], [a = 3, b = "world"]}), 2)

Answer: A B

Table.ReplaceErrorValues

Replaces the error values in the specified columns with the corresponding specified value.

Syntax: Table.ReplaceErrorValues(table as table, errorReplacement as list) as table

Example: Table.ReplaceErrorValues(Table.FromRows({{1,"hello"},{3,...}}, {"A","B"}), {"B", "world"})

Answer: A B

Example: Table.ReplaceErrorValues(Table.FromRows({{..., ...},{1,2}}, {"A","B"}), {{"A", "hello"}, {"B", "world"}})

Answer: A B

Table.ReplaceKeys

Returns a new table with new key information set in the keys argument.

Syntax: Table.ReplaceKeys(table as table, keys as list) as table

Table.ReplaceMatchingRows
Replaces specific rows from a table with the new rows.
Syntax: Table.ReplaceMatchingRows(table as table, replacements as list, optional equationCriteria as any)
Example: Table.ReplaceMatchingRows(Table.FromRecords({[a = 1, b =2], [a = 2, b = 3], [a = 3, b = 4], [a = 1, b = 2]}),{
Answer: {[a = 1, b = 2], [a = -1, b = -2]}, {[a = 2, b = 3], [a = -2, b = -3]} })

Table.ReplaceRelationshipIdentity
0
Syntax: Table.ReplaceRelationshipIdentity(value as any, identity as text) as any

Table.ReplaceRows
Returns a table where the rows beginning at an offset and continuing for count are replaced with the provided rows.
Syntax: Table.ReplaceRows(table as table, offset as number, count as number, rows as list) as table
Example: Table.ReplaceRows(Table.FromRecords({[Column1=1], [Column1=2], [Column1=3], [Column1=4], [Column1=5]}), 1, 3,
Answer: {[Column1=6], [Column1=7]})

Table.ReplaceValue
Replaces oldValue with newValue in specific columns of a table, using the provided replacer function, such as text.Replace or Value.Replace.
Syntax: Table.ReplaceValue(table as table, oldValue as any, newValue as any, replacer as function,
Example: Table.ReplaceValue(Table.FromRecords({[a = 1, b = "hello"], [a = 3, b = "goodbye"]}), "goodbye", "world",
Answer: Replacer.ReplaceText, {"b"})
Example: Table.ReplaceValue(Table.FromRecords({[a = 1, b = "hello"], [a = 3, b = "wurld"]}), "ur", "or",
Answer: Replacer.ReplaceText, {"b"})

Table.ReverseRows
Returns a table with the rows in reverse order.
Syntax: Table.ReverseRows(table as table) as table
Example: Table.ReverseRows(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.RowCount
Returns the number of rows in a table.
Syntax: Table.RowCount(table as table) as number
Example: Table.RowCount(Table.FromRecords({[CustomerID =1, Name ="Bob", Phone = "123-4567"],[CustomerID =2, Name
Answer: ="Jim", Phone = "987-6543"],[CustomerID =3, Name ="Paul", Phone = "543-7890"]}))

Table.Schema
Returns a table containing a description of the columns (i.e. the schema) of the specified table.
Syntax: Table.Schema(table as table) as table

Table.SelectColumns

Returns a table that contains only specific columns.

Syntax: Table.SelectColumns(table as table, columns as any, optional missingField as nullable number) as

Example: Table.SelectColumns(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2,

Answer: Name = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4,

Example: Table.SelectColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-4567"]}), {"CustomerID",

Answer: "Name"})

Example: Table.SelectColumns(Table.FromRecords({[CustomerID=1, Name="Bob", Phone = "123-4567"]}), "NewColumn")

Answer: [Expression.Error] The field 'NewColumn' of the record wasn't found.

Example: Table.SelectColumns(Table.FromRecords({[CustomerID=1, Name = "Bob", Phone = "123-4567"]}), {"CustomerID",

Answer: "NewColumn"}, MissingField.UseNull)

Table.SelectRows

Returns a table containing only the rows that match a condition.

Syntax: Table.SelectRows(table as table, condition as function) as table

Example: Table.SelectRows(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name

Answer: = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4, Name =

Example: Table.SelectRows(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name

Answer: = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4, Name =

Table.SelectRowsWithErrors

Returns a table with only the rows from table that contain an error in at least one of the cells in a row.

Syntax: Table.SelectRowsWithErrors(table as table, optional columns as nullable list) as table

Example: Table.SelectRowsWithErrors(Table.FromRecords({ [CustomerID =..., Name = "Bob", Phone = "123-4567"],

Answer: [CustomerID = 2, Name = "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] ,

Table.SingleRow

Returns a single row from a table.

Syntax: Table.SingleRow(table as table) as record

Example: Table.SingleRow(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"]}))

Answer: CUSTOMERID 1

Table.Skip

Returns a table that does not contain the first row or rows of the table.

Syntax: Table.Skip(table as table, countOrCondition as any) as table

Example: Table.Skip(Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name =

Answer: "Jim", Phone = "987-6543"] , [CustomerID = 3, Name = "Paul", Phone = "543-7890"] , [CustomerID = 4, Name =
Example: Table.Skip(Table.FromRecords({[CustomerID = 1, Name = "Bob", Phone = "123-4567"],[CustomerID = 2, Name =
Answer: "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =
Example: Table.Skip(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0], [OrderID =
Answer: 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID = 3, CustomerID = 2, Item = "Fishing net",

Table.Sort

Sorts the rows in a table using a comparisonCriteria or a default ordering if one is not specified.
Syntax: Table.Sort(table as table, comparisonCriteria as any) as table
Example: Table.Sort(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0], [OrderID =
Answer: 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID = 3, CustomerID = 2, Item = "Fishing net",
Example: Table.Sort(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0], [OrderID =
Answer: 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID = 3, CustomerID = 2, Item = "Fishing net",
Example: Table.Sort(Table.FromRecords({[OrderID = 1, CustomerID = 1, Item = "Fishing rod", Price = 100.0], [OrderID =
Answer: 2, CustomerID = 1, Item = "1 lb. worms", Price = 5.0], [OrderID = 3, CustomerID = 2, Item = "Fishing net",

Table.Split

Splits the specified table into a list of tables using the specified page size.
Syntax: Table.Split(table as table, pageSize as number) as list
Example: let Customers = Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.SplitColumn

Returns a new set of columns from a single column applying a splitter function to each value.
Syntax: Table.SplitColumn(table as table, sourceColumn as text, splitter as function, optional
Example: let Customers = Table.FromRecords({ [CustomerID = 1, Name = "Bob", Phone = "123-4567"], [CustomerID = 2, Name
Answer: = "Jim", Phone = "987-6543"], [CustomerID = 3, Name = "Paul", Phone = "543-7890"], [CustomerID = 4, Name =

Table.ToColumns

Returns a list of nested lists each representing a column of values in the input table.
Syntax: Table.ToColumns(table as table) as list

Table.ToList

Returns a table into a list by applying the specified combining function to each row of values in a table.
Syntax: Table.ToList(table as table, optional combiner as nullable function) as list

Example: Table.ToList(Table.FromRows({{Number.ToText(1),"Bob", "123-4567" },
{Number.ToText(2), "Jim", "987-6543" },
Answer: {Number.ToText(3), "Paul", "543-7890" }}), Combiner.CombineTextByDelimiter(","))

Table.ToRecords
Returns a list of records from an input table.
Syntax: Table.ToRecords(table as table) as list

Table.ToRows
Returns a nested list of row values from an input table.
Syntax: Table.ToRows(table as table) as list

Table.TransformColumnNames
Transforms column names by using the given function.
Syntax: Table.TransformColumnNames(table as table, nameGenerator as function, optional options as nullable
Example: Table.TransformColumnNames(Table.FromRecords({[#"Col#(tab)umn" = 1]}), Text.Clean)
Answer: COLUMN
Example: Table.TransformColumnNames(Table.FromRecords({[ColumnNum = 1, cOlumnnum = 2, coLumnNUM = 3]}), Text.Clean,
Answer: [MaxLength = 6, Comparer = Comparer.OrdinalIgnoreCase])

Table.TransformColumns
Transforms columns from a table using a function.
Syntax: Table.TransformColumns(table as table, transformOperations as list, optional defaultTransformation
Example: Table.TransformColumns(Table.FromRecords({[A="1", B=2], [A="5", B=10]}),{"A", Number.FromText})
Answer: A B
Example: Table.TransformColumns(Table.FromRecords({[A="1", B=2], [A="5", B=10]}), {"X", Number.FromText}, null,
Answer: MissingField.Ignore)
Example: Table.TransformColumns(Table.FromRecords({[A="1",B=2], [A="5", B=10]}), {"X", Number.FromText}, null,
Answer: MissingField.UseNull)
Example: Table.TransformColumns(Table.FromRecords({[A="1",B=2], [A="5", B=10]}), {"X", Number.FromText})
Answer: [Expression.Error] The column 'X' of the table wasn't found.

Table.TransformColumnTypes
Transforms the column types from a table using a type.
Syntax: Table.TransformColumnTypes(table as table, typeTransformations as list, optional culture as
Example: Table.TransformColumnTypes(Table.FromRecords({[a = 1, b = 2], [a = 3, b = 4]}), {"a", type text}, "en-US")
Answer: A B

Table.TransformRows
Transforms the rows from a table using a transform function.
Syntax: Table.TransformRows(table as table, transform as function) as list
Example: Table.TransformRows(Table.FromRecords({[a = 1], [a = 2], [a = 3], [a = 4], [a = 5]}), each [a])
Answer: 1

Example: Table.TransformRows(Table.FromRecords({[a = 1], [a = 2], [a = 3], [a = 4], [a = 5]}), (row) as record => [B =
Answer: Number.ToText(row[a])])

Table.Transpose

Returns a table with columns converted to rows and rows converted to columns from the input table.
Syntax: Table.Transpose(table as table, optional columns as any) as table
Example: Table.Transpose(Table.FromRecords({[Name = "Full Name", Value = "Fred"], [Name = "Age", Value = 42], [Name =
Answer: "Country", Value = "UK"]}))

Table.Unpivot

Given a list of table columns, transforms those columns into attribute-value pairs.
Syntax: Table.Unpivot(table as table, pivotColumns as list, attributeColumn as text, valueColumn as text)
Example: ({[key = "x", a = 1, b = null, c = 3], [key = "y", a = 2, b = 4, c = null]})
Answer: attribute-value pairs.

Table.UnpivotOtherColumns

Translates all columns other than a specified set into attribute-value pairs, combined with the rest of the values in each row.
Syntax: Table.UnpivotOtherColumns(table as table, pivotColumns as list, attributeColumn as text,
Example: Table.UnpivotOtherColumns(Table.FromRecords({ [key = "key1", attribute1 = 1, attribute2 = 2, attribute3 = 3
Answer:], [key = "key2", attribute1 = 4, attribute2 = 5, attribute3 = 6] }), { "key" }, "column1", "column2")

Table.View

Creates or extends a table with user-defined handlers for query and action operations.
Syntax: Table.View(table as nullable table, handlers as record) as table

Table.ViewFunction

Creates a function that can be intercepted by a handler defined on a view (via Table.View).
Syntax: Table.ViewFunction(function as function) as function

Tables.GetRelationships

Returns the relationships among a set of tables.
Syntax: Tables.GetRelationships(tables as table, optional dataColumn as nullable text) as table

Teradata.Database

Returns a table with data relating to the tables in the specified Teradata Database.
Syntax: Teradata.Database(server as text, optional options as nullable record) as table

Text.AfterDelimiter

Returns the portion of text after the specified delimiter.
Syntax: Text.AfterDelimiter(text as nullable text, delimiter as text, optional index as any) as any
Example: Text.AfterDelimiter("111-222-333", "-")
Example: Text.AfterDelimiter("111-222-333", "-", 1)
Example: Text.AfterDelimiter("111-222-333", "-", {1, RelativePosition.FromEnd})

Text.At

Returns a character starting at a zero-based offset.
Syntax: Text.At(text as nullable text, index as number) as nullable text
Example: Text.At("Hello, World", 4)

Text.BeforeDelimiter

Returns the portion of text before the specified delimiter.
Syntax: Text.BeforeDelimiter(text as nullable text, delimiter as text, optional index as any) as any
Example: Text.BeforeDelimiter("111-222-333", "-")
Example: Text.BeforeDelimiter("111-222-333", "-", 1)
Example: Text.BeforeDelimiter("111-222-333", "-", {1, RelativePosition.FromEnd})

Text.BetweenDelimiters

Returns the portion of text between the specified startDelimiter and endDelimiter.
Syntax: Text.BetweenDelimiters(text as nullable text, startDelimiter as text, endDelimiter as text,
Example: Text.BetweenDelimiters("111 (222) 333 (444)", "(", ")")
Answer: 222
Example: Text.BetweenDelimiters("111 (222) 333 (444)", "(", ")", 1, 0)
Answer: 444
Example: Text.BetweenDelimiters("111 (222) 333 (444)", "(", ")", {1, RelativePosition.FromEnd}, {1,
Answer: RelativePosition.FromStart})

Text.Clean

Returns the original text value with non-printable characters removed.
Syntax: Text.Clean(text as nullable text) as nullable text
Example: Text.Clean("ABC#(lf)D")

Text.Combine

Returns a text value that is the result of joining all text values with each value separated by a separator.
Syntax: Text.Combine(texts as list, optional separator as nullable text) as text
Example: Text.Combine({"Seattle", "WA"})
Example: Text.Combine({"Seattle", "WA"}, ", ")
Answer: Seattle, WA

Text.Contains

Returns true if a text value substring was found within a text value string
Example: otherwise, false.
Syntax: Text.Contains(text as nullable text, substring as text, optional comparer as nullable function) as
Example: Text.Contains("Hello World", "Hello")
Example: Text.Contains("Hello World", "hello")

Text.End

Returns the number of characters from the end of a text value.
Syntax: Text.End(text as nullable text, count as number) as nullable text
Example: Text.End("Hello, World", 5)

Text.EndsWith

Returns a logical value indicating whether a text value substring was found at the end of a string.

Syntax: Text.EndsWith(text as nullable text, substring as text, optional comparer as nullable function) as
Example: Text.EndsWith("Hello, World", "world")
Example: Text.EndsWith("Hello, World", "World")

Text.Format
Syntax: Text.Format(formatString as text, arguments as any, optional culture as nullable text) as text
Example: Text.Format("#{0}, #{1}, and #{2}.", { 17, 7, 22 })
Answer: 17, 7, and 22.
Example: Text.Format("The time for the #[distance] km run held in #[city] on #[date] was #[duration].", [city =
Answer: "Seattle", date = #date(2015, 3, 10), duration = #duration(0,0,54,40), distance = 10], "en-US")

Text.From
Returns the text representation of a number, date, time, datetime, datetimezone, logical, duration or binary value. If a value is null, Text.From returns null. The optional culture parameter is used to format the text value according to the given culture.
Syntax: Text.From(value as any, optional culture as nullable text) as nullable text
Example: Text.From(3)

Text.FromBinary
Decodes data from a binary value in to a text value using an encoding.
Syntax: Text.FromBinary(binary as nullable binary, optional encoding as nullable number) as nullable text

Text.InferNumberType
Infers granular number type (Int64.Type, Double.Type, etc.) of text using culture.
Syntax: Text.InferNumberType(text as text, optional culture as nullable text) as type

Text.Insert
Returns a text value with newValue inserted into a text value starting at a zero-based offset.
Syntax: Text.Insert(text as nullable text, offset as number, newText as text) as nullable text
Example: Text.Insert("ABD", 2, "C")

Text.Length
Returns the number of characters in a text value.
Syntax: Text.Length(text as nullable text) as nullable number
Example: Text.Length("Hello World")

Text.Lower
Returns the lowercase of a text value.
Syntax: Text.Lower(text as nullable text, optional culture as nullable text) as nullable text
Example: Text.Lower("AbCd")

Text.Middle
Returns the substring up to a specific length.
Syntax: Text.Middle(text as nullable text, start as number, optional count as nullable number) as nullable
Example: Text.Middle("Hello World", 6, 5)
Example: Text.Middle("Hello World", 6, 20)

Text.NewGuid
Returns a Guid value as a text value.
Syntax: Text.NewGuid() as text

Text.PadEnd
Returns a text value padded at the end with pad to make it at least length characters.
Syntax: Text.PadEnd(text as nullable text, count as number, optional character as nullable text) as
Example: Text.PadEnd("Name", 10)
Example: Text.PadEnd("Name", 10, "|")

Text.PadStart
Returns a text value padded at the beginning with pad to make it at least length characters. If pad is not specified, whitespace is used as pad.
Syntax: Text.PadStart(text as nullable text, count as number, optional character as nullable text) as
Example: Text.PadStart("Name", 10)
Example: Text.PadStart("Name", 10, "|")

Text.PositionOf
Returns the first occurrence of substring in a string and returns its position starting at startOffset.
Syntax: Text.PositionOf(text as text, substring as text, optional occurrence as nullable number, optional
Example: Text.PositionOf("Hello, World! Hello, World!", "World")
Example: Text.PositionOf("Hello, World! Hello, World!", "World", Occurrence.Last)

Text.PositionOfAny
Returns the first occurrence of a text value in list and returns its position starting at startOffset.
Syntax: Text.PositionOfAny(text as text, characters as list, optional occurrence as nullable number) as
Example: Text.PositionOfAny("Hello, World!", {"W"})
Example: Text.PositionOfAny("Hello, World!", {"H","W"})

Text.Proper
Returns a text value with first letters of all words converted to uppercase.
Syntax: Text.Proper(text as nullable text, optional culture as nullable text) as nullable text
Example: Text.Proper
Answer: Text.Proper("the QUICK BrOWn fOx jUmPs oVER tHe LAzy DoG")

Text.Range
Returns a number of characters from a text value starting at a zero-based offset and for count number of characters.
Syntax: Text.Range(text as nullable text, offset as number, optional count as nullable number) as nullable
Example: Text.Range("Hello World", 6)
Example: Text.Range("Hello World Hello", 6, 5)

Text.Remove
Removes all occurrences of a character or list of characters from a text value. The removeChars parameter can be a character value or a list of character values.
Syntax: Text.Remove(text as nullable text, removeChars as any) as nullable text
Example: Text.Remove("a,b;c",{",",";"})

Text.RemoveRange
Removes count characters at a zero-based offset from a text value.
Syntax: Text.RemoveRange(text as nullable text, offset as number, optional count as nullable number) as
Example: Text.RemoveRange("ABEFC", 2)
Example: Text.RemoveRange("ABEFC", 2, 2)

Text.Repeat
Returns a text value composed of the input text value repeated a number of times.
Syntax: Text.Repeat(text as nullable text, count as number) as nullable text
Example: Text.Repeat("a", 5)
Example: Text.Repeat("helloworld.", 3)
Answer: helloworld.helloworld.helloworld.

Text.Replace
Replaces all occurrences of a substring with a new text value.
Syntax: Text.Replace(text as nullable text, old as text, new as text) as nullable text
Example: Text.Replace("the quick brown fox jumps over the lazy dog", "the", "a")
Answer: a quick brown fox jumps over a lazy dog

Text.ReplaceRange
Replaces length characters in a text value starting at a zero-based offset with the new text value.
Syntax: Text.ReplaceRange(text as nullable text, offset as number, count as number, newText as text) as
Example: Text.ReplaceRange("ABGF", 2, 1, "CDE")

Text.Reverse
Reverses the provided text.
Syntax: Text.Reverse(text as nullable text) as nullable text
Example: Text.Reverse("123")

Text.Select
Selects all occurrences of the given character or list of characters from the input text value.
Syntax: Text.Select(text as nullable text, selectChars as any) as nullable text
Example: Text.Select("a,b;c", {"a".."z"})

Text.Split
Returns a list containing parts of a text value that are delimited by a separator text value.
Syntax: Text.Split(text as text, separator as text) as list
Example: Text.Split("Name|Address|PhoneNumber", "|")

Text.SplitAny
Returns a list containing parts of a text value that are delimited by any separator text values.
Syntax: Text.SplitAny(text as text, separators as text) as list
Example: Text.SplitAny("Jamie|Campbell|Admin|Adventure Works|www.adventure-works.com", "|")

Text.Start
Returns the count of characters from the start of a text value.
Syntax: Text.Start(text as nullable text, count as number) as nullable text
Example: Text.Start("Hello, World", 5)

Text.StartsWith

Returns a logical value indicating whether a text value substring was found at the beginning of a string.

Syntax: Text.StartsWith(text as nullable text, substring as text, optional comparer as nullable function)

Example: Text.StartsWith("Hello, World", "hello")

Example: Text.StartsWith("Hello, World", "Hello")

Text.ToBinary

Encodes a text value into binary value using an encoding.

Syntax: Text.ToBinary(text as nullable text, optional encoding as nullable number, optional

Text.ToList

Returns a list of characters from a text value.

Syntax: Text.ToList(text as text) as list

Example: Text.ToList("Hello World")

Text.Trim

Removes any occurrences of characters in trimChars from text.

Syntax: Text.Trim(text as nullable text, optional trim as any) as nullable text

Example: Text.Trim(" a b c d ")

Text.TrimEnd

Removes any occurrences of the characters specified in trimChars from the end of the original text value.

Syntax: Text.TrimEnd(text as nullable text, optional trim as any) as nullable text

Example: Text.TrimEnd(" a b c d ")

Text.TrimStart

Removes any occurrences of the characters in trimChars from the start of the original text value.

Syntax: Text.TrimStart(text as nullable text, optional trim as any) as nullable text

Example: Text.TrimStart(" a b c d ")

Text.Upper

Returns the uppercase of a text value.

Syntax: Text.Upper(text as nullable text, optional culture as nullable text) as nullable text

Example: Text.Upper("aBcD")

TextEncoding.Ascii

Use to choose the ASCII binary form.

Value: 20127

TextEncoding.BigEndianUnicode

Use to choose the UTF16 big endian binary form.

Value: 1201

TextEncoding.Unicode

Use to choose the UTF16 little endian binary form.

Value: 1200

TextEncoding.Utf16
Use to choose the UTF16 little endian binary form.
Value: 1200

TextEncoding.Utf8
Use to choose the UTF8 binary form.
Value: 65001

TextEncoding.Windows
Use to choose the Windows binary form.
Value: 1252

Time.EndOfHour
Returns a DateTime value from the end of the hour.
Syntax: Time.EndOfHour(dateTime as any) as any
Example: Time.EndOfHour(#datetime(2011, 5, 14, 17, 0, 0))
Answer: #datetime(2011, 5, 14, 17, 59, 59.9999999)
Example: Time.EndOfHour(#datetimezone(2011, 5, 17, 5, 0, 0, -7, 0))
Answer: #datetimezone(2011, 5, 17, 5, 59, 59.9999999, -7, 0)

Time.From
Returns a time value from a value.
Syntax: Time.From(value as any, optional culture as nullable text) as nullable time
Example: Time.From(0.7575)
Answer: #time(18,10,48)
Example: Time.From(#datetime(1899, 12, 30, 06, 45, 12))
Answer: #time(06, 45, 12)

Time.FromText
Returns a Time value from a set of date formats.
Syntax: Time.FromText(text as nullable text, optional culture as nullable text) as nullable time
Example: Time.FromText("10:12:31am")
Answer: #time(10, 12, 31)
Example: Time.FromText("1012")
Answer: #time(10, 12, 00)
Example: Time.FromText("10")
Answer: #time(10, 00, 00)

Time.Hour
Returns an hour value from a DateTime value.
Syntax: Time.Hour(dateTime as any) as nullable number
Example: Time.Hour(#datetime(2011, 12, 31, 9, 15, 36))

Time.Minute
Returns a minute value from a DateTime value.
Syntax: Time.Minute(dateTime as any) as nullable number
Example: Time.Minute(#datetime(2011, 12, 31, 9, 15, 36))

Time.Second
Returns a second value from a DateTime value
Syntax: Time.Second(dateTime as any) as nullable number`

Example: Time.Second(#datetime(2011, 12, 31, 9, 15, 36.5))

Time.StartOfHour
Returns the first value of the hour from a time value.
Syntax: Time.StartOfHour(dateTime as any) as any
Example: Time.StartOfHour(#datetime(2011, 10, 10, 8, 10, 32))
Answer: #datetime(2011, 10, 10, 8, 0, 0)

Time.ToRecord
Returns a record containing parts of a Date value.
Syntax: Time.ToRecord(time as time) as record
Example: Time.ToRecord(#time(11, 56, 2))
Answer: HOUR 11

Time.ToText
Returns a text value from a Time value.
Syntax: Time.ToText(time as nullable time, optional format as nullable text, optional culture as nullable
Example: Time.ToText(#time(11, 56, 2))
Answer: 0.497222222222222
Example: Time.ToText(#time(11, 56, 2), "hh:mm")
Answer: 0.497222222222222

TraceLevel.Critical
Returns 1, the value for Critical trace level.
Value: 1

TraceLevel.Error
Returns 2, the value for Error trace level.
Value: 2

TraceLevel.Information
Returns 4, the value for Information trace level.
Value: 8

TraceLevel.Verbose
Returns 5, the value for Verbose trace level.
Value: 16

TraceLevel.Warning
Returns 3, the value for Warning trace level.
Value: 4

Type.AddTableKey
Add a key to a table type.
Syntax: Type.AddTableKey(table as type, columns as list, isPrimary as logical) as type

Type.ClosedRecord
The given type must be a record type returns a closed version of the given record type (or the same type, if it is already closed)
Syntax: Type.ClosedRecord(type as type) as type

Example: Type.ClosedRecord(type [A = number,...])
Answer: type [A = number]

Type.Facets

Returns the facets of a type.
Syntax: Type.Facets(type as type) as record

Type.ForFunction

Creates a function type from the given .
Syntax: Type.ForFunction(signature as record, min as number) as type
Example: Type.ForFunction([ReturnType = type number, Parameters = [X = type number]], 1)
Answer: type function (X as number) as number

Type.ForRecord

Returns a Record type from a fields record.
Syntax: Type.ForRecord(fields as record, open as logical) as type

Type.FunctionParameters

Returns a record with field values set to the name of the parameters of a function type, and their values set to their corresponding types.
Syntax: Type.FunctionParameters(type as type) as record

Type.FunctionRequiredParameters

Returns a number indicating the minimum number of parameters required to invoke the a type of function.
Syntax: Type.FunctionRequiredParameters(type as type) as number
Example: Type.FunctionRequiredParameters(type function (x as number, optional y as text) as any)
Answer: 1

Type.FunctionReturn

Returns a type returned by a function type.
Syntax: Type.FunctionReturn(type as type) as type
Example: Type.FunctionReturn(type function () as any)
Answer: type any

Type.Is

Type.Is
Syntax: Type.Is(type1 as type, type2 as type) as logical

Type.IsNullable

Returns true if a type is a nullable type
Example: otherwise, false.
Syntax: Type.IsNullable(type as type) as logical
Example: Type.IsNullable(type number)
Answer: FALSE
Example: Type.IsNullable(type nullable number)
Answer: TRUE

Type.IsOpenRecord

Returns whether a record type is open.
Syntax: Type.IsOpenRecord(type as type) as logical

Example: Type.IsOpenRecord(type [A = number,...])
Answer: TRUE

Type.ListItem
Returns an item type from a list type.
Syntax: Type.ListItem(type as type) as type
Example: Type.ListItem(type {number})
Answer: type number

Type.NonNullable
Returns the non nullable type from a type.
Syntax: Type.NonNullable(type as type) as type
Example: Type.NonNullable(type nullable number)
Answer: type number

Type.OpenRecord
Returns an opened version of a record type, or the same type, if it is already open.
Syntax: Type.OpenRecord(type as type) as type
Example: Type.OpenRecord(type [A = number])
Answer: type [A = number, ...]

Type.RecordFields
Returns a record describing the fields of a record type with each field of the returned record type having a corresponding name and a value that is a record of the form [Type = type, Opional = logical].
Syntax: Type.RecordFields(type as type) as record

Type.ReplaceFacets
Replaces the facets of a type.
Syntax: Type.ReplaceFacets(type as type, facets as record) as type

Type.ReplaceTableKeys
Replaces the keys in a table type.
Syntax: Type.ReplaceTableKeys(tableType as type, keys as list) as type

Type.TableColumn
Returns the type of a column in a table.
Syntax: Type.TableColumn(tableType as type, column as text) as type

Type.TableKeys
Returns keys from a table type.
Syntax: Type.TableKeys(tableType as type) as list

Type.TableRow
Returns a row type from a table type.
Syntax: Type.TableRow(table as type) as type

Type.TableSchema
Returns a table containing a description of the columns (i.e. the schema) of the specified table type.
Syntax: Type.TableSchema(tableType as type) as table

Type.Union

Returns the union of a list of types.

Syntax: Type.Union(types as list) as type

Uri.BuildQueryString

Assemble a record into a URI query string.

Syntax: Uri.BuildQueryString(query as record) as text

Uri.Combine

Returns a Uri based on the combination of the base and relative parts.

Syntax: Uri.Combine(baseUri as text, relativeUri as text) as text

Uri.EscapeDataString

Encodes special characters in accordance with RFC 3986.

Syntax: Uri.EscapeDataString(data as text) as text

Uri.Parts

Returns a record value with the fields set to the parts of a Uri text value.

Syntax: Uri.Parts(absoluteUri as text) as record

Example: Uri.Parts("www.adventure-works.com")

Answer: SCHEME http

Example: let UriUnescapeDataString = (data as text) as text => Uri.Parts("http://contoso?a=" & data)[Query][a] in

Answer: UriUnescapeDataString("%2Bmoney%24")

Value.Add

Returns the sum of the two values.

Syntax: Value.Add(value1 as any, value2 as any, optional precision as nullable number) as any

Value.As

Value.As is the function corresponding to the as operator in the formula language. The expression value as type asserts that the value of a value argument is compatible with type as per the is operator. If it is not compatible, an error is raised.

Syntax: Value.As(value as any, type as type) as any

Value.Compare

Returns 1, 0, or -1 based on value1 being greater than, equal to, or less than the value2. An optional comparer function can be provided.

Syntax: Value.Compare(value1 as any, value2 as any, optional precision as nullable number) as number

Value.Divide

Returns the result of dividing the first value by the second.

Syntax: Value.Divide(value1 as any, value2 as any, optional precision as nullable number) as any

Value.Equals

Returns whether two values are equal.

Syntax: Value.Equals(value1 as any, value2 as any, optional precision as nullable number) as logical

Value.Firewall

Value.Firewall

Syntax: Value.Firewall(key as text) as any

Value.FromText
Decodes a value from a textual representation, value, and interprets it as a value with an appropriate type. Value.FromText takes a text value and returns a number, a logical value, a null value, a DateTime value, a Duration value, or a text value. The empty text value is interpreted as a null value.
Syntax: Value.FromText(text as any, optional culture as nullable text) as any

Value.Is
Value.Is is the function corresponding to the is operator in the formula language. The expression value is type returns true if the ascribed type of vlaue is compatible with type, and returns false if the ascribed type of value is incompatible with type.
Syntax: Value.Is(value as any, type as type) as logical

Value.Metadata
Returns a record containing the input’s metadata.
Syntax: Value.Metadata(value as any) as any

Value.Multiply
Returns the product of the two values.
Syntax: Value.Multiply(value1 as any, value2 as any, optional precision as nullable number) as any

Value.NativeQuery
Evaluates a query against a target.
Syntax: Value.NativeQuery(target as any, query as text, optional parameters as any, optional options as

Value.NullableEquals
Returns a logical value or null based on two values .
Syntax: Value.NullableEquals(value1 as any, value2 as any, optional precision as nullable number) as

Value.RemoveMetadata
Removes the metadata on the value and returns the original value.
Syntax: Value.RemoveMetadata(value as any, optional metaValue as any) as any

Value.ReplaceMetadata
Replaces the metadata on a value with the new metadata record provided and returns the original value with the new metadata attached.
Syntax: Value.ReplaceMetadata(value as any, metaValue as any) as any

Value.ReplaceType
A value may be ascribed a type using Value.ReplaceType. Value.ReplaceType either returns a new value with the type ascribed or raises an error if the new type is incompatible with the value’s native primitive type. In particular, the function raises an error when an attempt is made to ascribe an abstract type, such as any. When replacing a the type of a record, the new type must have the same number of fields, and the new fields replace the old fields by ordinal position, not by name. Similarly, when replacing the type of a table, the new type must have the same number of columns, and the new columns replace the old columns by ordinal position.
Syntax: Value.ReplaceType(value as any, type as type) as any

Value.ResourceExpression

Value.Subtract
Returns the difference of the two values.
Syntax: Value.Subtract(value1 as any, value2 as any, optional precision as nullable number) as any

Value.Type
Returns the type of the given value.
Syntax: Value.Type(value as any) as type

Variable.Value
Variable.Value
Syntax: Variable.Value(identifier as text) as any

Web.Contents
Returns the contents downloaded from a web url as a binary value.
Syntax: Web.Contents(url as text, optional options as nullable record) as binary

Web.Page
Returns the contents of an HTML webpage as a table.
Syntax: Web.Page(html as any) as table

WebAction.Request
Creates an action that, when executed, will return the results of performing a method request against url using HTTP as a binary value.
Syntax: WebAction.Request(method as text, url as text, optional options as nullable record) as action

WebMethod.Delete
Specifies the DELETE method for HTTP.

WebMethod.Get
Specifies the GET method for HTTP.

WebMethod.Head
Specifies the HEAD method for HTTP.

WebMethod.Patch
Specifies the PATCH method for HTTP.

WebMethod.Post
Specifies the POST method for HTTP.

WebMethod.Put
Specifies the PUT method for HTTP.

Xml.Document
Returns the contents of an XML document as a hierarchical table (list of records).

Xml.Tables

Returns the contents of an XML document as a nested collection of flattened tables

Syntax: Xml.Tables(contents as any, optional options as nullable record, optional encoding as nullable