DAX Functions

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

DAX functions play an important role in the usage of DAX for data modeling and reporting. It is an inbuilt function provided in the DAX language that helps you perform commonly used data calculations on the Data Model.

Some of the DAX functions have the same names and functionality as that of Excel functions, however, they have been modified to use DAX data types and to work with tables and columns. DAX has additional functions that are designed to work with relational data and perform dynamic aggregation.

Audience

This tutorial has been designed for all those readers who depend heavily on MS-Excel to prepare charts, tables, and professional reports that involve complex data. It will help all those readers who use MS-Excel regularly to analyze data. Professionals who use data modeling and data analysis for reporting and decision-making purposes will benefit from this.

Prerequisites

This tutorial is an extension to Excel Power Pivot tutorial, hence it is a good idea to brush up on the Excel Power Pivot tutorial before you delve into DAX. Knowledge of Excel Functions and Excel Formulas is not necessary for this tutorial, as DAX is entirely for the Data Model in the Power Pivot window.

Disclaimer & Copyright

Table of Contents

DAX Functions – Basics

1. DAX Functions – Introduction

DAX stands for **D**ata **A**nalysis E**x**pressions. DAX is a formula language and is a collection of functions, operators, and constants that can be used in a formula or expression to calculate and return one or more values. DAX is the formula language associated with the Data Model of Microsoft Excel Power Pivot and with Microsoft Power BI.

DAX is not a programming language, however it is a formula language that allows the users to define custom calculations in calculated columns and calculated fields (also known as measures). DAX helps you create new information from the existing data in your Data Model. DAX formulas enable you to perform data modeling, data analysis, and use the results for reporting and decision making.

For an in-depth understanding of DAX, refer to the tutorial – DAX in this tutorials library.

What is a DAX Function?

A DAX function is an inbuilt function provided in the DAX language to enable you to perform various actions on the data in the tables in your Data Model.

DAX functions enable you to perform commonly used data calculations on the Data Model. Some of the DAX functions have same names and functionality as that of Excel functions but have been modified to use DAX data types and to work with tables and columns, as highlighted in the next section. DAX has additional functions that are designed to work with relational data and perform dynamic aggregation.

DAX functions play an important role in the usage of DAX for data modeling and reporting.

Excel Functions vs. DAX Functions

There are certain similarities between the Excel functions and the DAX functions and there are certain differences too. Following are the similarities and differences between Excel functions and DAX functions:

Similarities Between Excel Functions and DAX Functions

- Certain DAX functions have the same name and the same general behavior as Excel functions.

- DAX has lookup functions that are similar to the array and vector lookup functions in Excel.

Differences Between Excel Functions and DAX Functions

- DAX functions have been modified to take different types of inputs and some of the DAX functions might return a different data type. Hence, you need to understand the usage of these functions separately though they have the same name.

- You cannot use DAX functions in an Excel formula or use Excel functions in DAX formula, without the required modifications.

- Excel functions take a cell reference or a range of cells as a reference. DAX functions never take a cell reference or a range of cells as a reference, but instead take a column or table as a reference.

- Excel date and time functions return an integer that represents a date as a serial number. DAX date and time functions return a datetime data type that is in DAX but not in Excel.

- Excel has no functions that return a table, but some functions can work with arrays. Many of the DAX functions can easily reference complete tables and columns to perform calculations and return a table or a column of values. This ability of DAX adds power to the Power Pivot, Power View and Power BI, where DAX is used.

- DAX lookup functions require that a relationship is established between the respective tables.

DAX Parameter Naming Conventions

DAX has standard parameter names to facilitate the usage and understanding of the DAX functions. Further, you can use certain prefixes to the parameter names. If the prefix is clear enough, you can use the prefix itself as the parameter name.

You need to understand DAX parameter naming conventions so as to understand the syntax of the DAX functions and use the values for the required parameters correctly.

Refer to the chapter - DAX Parameter Naming Conventions for details.

Types of DAX Functions

DAX supports the following types of functions.

- DAX Table-Valued Functions
    - DAX Filter Functions
    - DAX Aggregation Functions
    - DAX Time Intelligence Functions
- DAX Date and Time Functions
- DAX Information Functions
- DAX Logical Functions
- DAX Math and Trig Functions
- DAX Other Functions
- DAX Parent and Child Functions
- DAX Statistical Functions
- DAX Text Functions

DAX Table-Valued Functions

Many DAX functions take tables as input or output tables or do both. These DAX functions are called DAX table-valued functions. Because a table can have a single column, DAX table-valued functions also take single columns as inputs. You have the following types of DAX table-valued functions:

- DAX Aggregation functions
- DAX Filter functions
- DAX Time intelligence functions

DAX Aggregation Functions

DAX Aggregation functions aggregate any expression over the rows of a table and are useful in calculations.

Refer to the chapter - DAX Aggregation functions for details.

DAX Filter Functions

DAX Filter functions return a column or a table or values related to the current row. You can use DAX Filter functions to return specific data types, look up values in related tables and filter by related values. DAX Lookup functions work by using tables and relationships between them. DAX Filter functions enable you to manipulate the data context to create dynamic calculations.

Refer to the chapter - DAX Filter functions for details.

DAX Time Intelligence Functions

DAX Time Intelligence functions return a table of dates or the use a table of dates to calculate an aggregation. These DAX functions help you create calculations that support the needs of Business Intelligence analysis by enabling you to manipulate data using time periods, including days, months, quarters, and years.

Refer to the chapter - DAX Time Intelligence functions for details.

DAX Date and Time Functions

DAX Date and Time functions are similar to the Excel date and time functions. However, DAX Date and Time functions are based on the datetime data type of DAX.

Refer to the chapter - DAX Date and Time functions for details.

DAX Information Functions

DAX Information functions look at the cell or row that is provided as an argument and tell you whether the value matches the expected type.

Refer to the chapter - DAX Information functions for details.

DAX Logical Functions

DAX Logical Functions return information about values in an expression. For example, DAX TRUE function lets you know whether an expression that you are evaluating returns a TRUE value.

Refer to the chapter - DAX Logical functions for details.

DAX Math and Trig Functions

DAX Mathematical and Trigonometric functions are very similar to the Excel mathematical and trigonometric functions.

Refer to the chapter - DAX Math and Trig functions for details.

DAX Parent and Child Functions

DAX Parent and Child functions are useful in managing data that is presented as a parent/child hierarchy in the Data Model.

Refer to the chapter - DAX Parent and Child functions for details.

DAX Statistical Functions

DAX Statistical functions are very similar to the Excel Statistical functions.

Refer to the chapter - DAX Statistical functions for details.

DAX Text Functions

DAX Text functions work with tables and columns. With DAX Text functions, you can return part of a string, search for text within a string or concatenate string values. You can also control the formats for dates, times, and numbers.

Refer to the chapter - DAX Text functions for details.

DAX Other Functions

These DAX functions perform unique actions that cannot be defined by any of the categories most other functions belong to.

Refer to the chapter - DAX Other functions for details.

DAX Function Description Structure

If you have to use a DAX function in a DAX formula, you need to understand the function in detail. You should know the syntax of the function, the parameter types, what the function returns, etc.

In this tutorial, a common function description structure is used for all the DAX functions so that you can read and interpret the DAX functions effectively.

Refer to the chapter - DAX Function Description Structure for details.

2. DAX Functions – DAX Parameter Naming Conventions

DAX has standard parameter names to facilitate the usage and understanding of the DAX functions. Further, you can use certain prefixes to the parameter names. If the prefix is clear enough, you can use the prefix itself as the parameter name.

To understand the syntax of the DAX functions and to use data values appropriately for the relevant DAX function parameters, you need to understand DAX parameter naming conventions.

Parameter Names

Following are the DAX standard parameter names –

Parameter Name	Description
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).
value	Any DAX expression that returns a single scalar value where the expression is to be evaluated exactly once before all other operations.
table	Any DAX expression that returns a table of data.
tableName	The name of an existing table using standard DAX syntax. It cannot be an expression.
columnName	The name of an existing column using standard DAX syntax, usually fully qualified. It cannot be an expression.
name	A string constant that will be used to provide the name of a new object.
order	An enumeration used to determine the sort order.
ties	An enumeration used to determine the handling of tie values.
type	An enumeration used to determine the data type for PathItem and PathItemReverse.

Prefixing Parameter Names or Using the Prefix Only

You can qualify a parameter name with a prefix –

- The prefix should be descriptive of how the argument is used.
- The prefix should be in such a way that ambiguous reading of the parameter is avoided.

For example,

- **Result_ColumnName** - Refers to an existing column used to get the result values in the DAX LOOKUPVALUE () function.
- **Search_ColumnName** - Refers to an existing column used to search for a value in the DAX LOOKUPVALUE () function.

You can omit the parameter name and use only the prefix, if the prefix is clear enough to describe the parameter. Omitting the parameter name and using only prefix can sometimes help in avoiding the clutter during reading.

For example, Consider **DATE (Year_value, Month_value, Day_value)**. You can omit the parameter name – value, that is repeated thrice and write it as DATE (Year, Month, Day). As seen, by using only the prefixes, the function is more readable. However, sometimes the parameter name and the prefix have to be present for clarity.

For example, Consider **Year_columnName**. The parameter name is ColumnName and the prefix is Year. Both are required to make the user understand that the parameter requires a reference to an existing column of years.

3. DAX Functions – Description Structure

If you have to use a DAX function in a DAX formula, you need to understand the function in detail. You should know the syntax of the function, the parameter types, what the function returns, etc.

To enable you to understand how to read and interpret the DAX functions, a uniform function description structure is used in this tutorial.

- The different types of DAX functions are grouped by the type name of the DAX functions as chapters.
- Each of these chapters provides a brief description of the utility of the respective type of DAX functions.
- The brief description will be followed by the list of DAX functions corresponding to that chapter (Type/Category of DAX functions).
- Each DAX function name is hyperlinked to DAX function details that have the following DAX function description structure:
    - Description
    - Syntax
    - Parameters
    - Return Value
    - Remarks
    - Example

The following sections explain each of these headings that appear in each DAX function explanation.

Description

In the Description section, you will learn what the DAX function is about and where it can be used.

If the DAX function is introduced in Excel 2016, the same will be mentioned here. (Rest of the DAX functions exist in Excel 2013.)

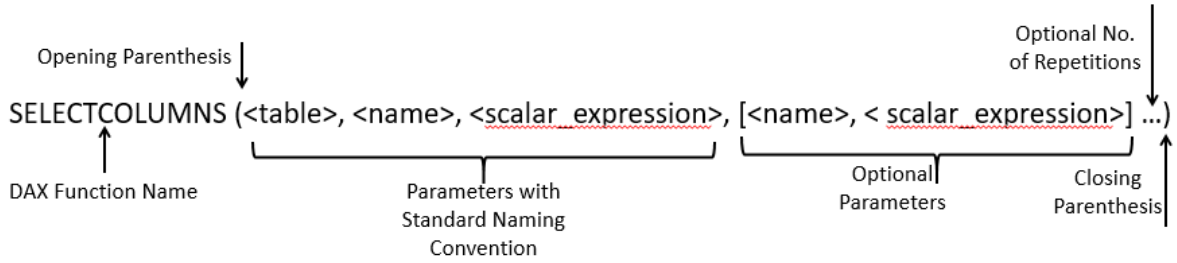
Syntax

In the Syntax section, you will learn the exact function name and the respective parameters.

- DAX function name is given in UPPERCASE letters.
- DAX function name is followed by opening parenthesis.

- Each parameter follows standard DAX parameter naming convention and is enclosed in angle brackets.
- If a parameter is optional, it is further enclosed in square brackets.
- The parameters are separated by commas.
- Ellipses … are used to show an optional number of repetitions of parameters.
- The function syntax ends with closing parenthesis.

Example



Parameters

In the Parameters section, each of the parameters of the specific DAX function is listed in a table with its description. For example, the parameters of the above example DAX function SELECTCOLUMNS is listed in the following table.

Parameter	Description
Table	Table or a DAX expression that returns a table.
Name	The name given to the column, enclosed in double quotes.
scalar_expression	DAX expression that returns a scalar value like a column reference, integer, or string value.

Return Value

In the Return Value section, you will learn about what value the DAX function will return and its data type.

Remarks

In the Remarks section, you will learn about any extra information that you need to know about the usage of the DAX function. You will also understand the potential errors and the reasons.

Example

An example of the usage of the DAX function is given in this section.

Note: When you write DAX functions with the data values for the parameters, you will follow the naming conventions as given below:

- A Table name is specified as it appears in the Data Model. E.g. Sales.
- A Column name is specified as it appears in the Data Model with square brackets enclosing it.

  For example, [Sales Amount]

  - It is recommended to use fully qualified names for columns, i.e. a column name is prefixed with the table name that contains it.

    For example, Sales[Sales Amount].

  - If the table name contains spaces, it should be enclosed in single quotes.

    For example, 'East Sales'[Sales Amount]

- A DAX function can return a column or table of values, in which case, it needs to be used as a parameter of another DAX function that requires a column or table.

DAX Aggregation Functions

4. DAX Aggregate Functions – Overview

DAX Aggregation functions aggregate any expression over the rows of a table and are useful in calculations.

Following are the DAX Aggregation functions:

- DAX ADDCOLUMNS function
- DAX AVERAGE function
- DAX AVERAGEA function
- DAX AVERAGEX function
- DAX COUNT function
- DAX COUNTA function
- DAX COUNTAX function
- DAX COUNTBLANK function
- DAX COUNTROWS function
- DAX COUNTX function
- DAX CROSSJOIN function
- DAX DISTINCTCOUNT function
- DAX GENERATE function
- DAX GENERATEALL function
- DAX MAX function
- DAX MAXA function
- DAX MAXX function
- DAX MIN function
- DAX MINA function
- DAX MINX function
- DAX PRODUCT function
- DAX PRODUCTX function
- DAX ROW function
- DAX SELECTCOLUMNS function
- DAX SUM function
- DAX SUMMARIZE function
- DAX SUMMARIZE function with Options
- DAX SUMX function
- DAX TOPN function

5. DAX Functions – ADDCOLUMNS

Description

Adds calculated columns to the given table or table expression.

Syntax

ADDCOLUMNS (<table>, <name>, <expression>, [<name>, <expression>] …)

Parameters

Parameter	Description
table	Table or a DAX expression that returns a table.
name	The name given to the column, enclosed in double quotes.
expression	DAX expression that returns a scalar expression, evaluated for each row of table.

Return Value

A table with all its original columns and the added ones.

Remarks

--

Example

```
=ADDCOLUMNS (
        Products,"East_Sales", SUMX (RELATEDTABLE(East_Sales),
                IF([Product]=East_Sales[Product],
                        East_Sales[Sales
                Amount],0)
                )
        )
```

6. DAX Functions – AVERAGE

Description

Returns the average (arithmetic mean) of all the numbers in a column.

Syntax

AVERAGE (<column>)

Parameters

Parameter	Description
Column	The column that contains the numbers for which you want the average.

Return Value

Returns a decimal number that represents the arithmetic mean of the numbers in the column.

Remarks

- If the column contains logical values or empty cells, those values are ignored and the rows are not counted.
- Cells with the value zero are included and the rows are counted for the divisor.
- Whenever there are no rows to aggregate, the function returns a blank. However, if there are rows, but none of them meet the specified criteria, the function returns 0.

Example

=AVERAGE (Sales[Sales Amount])

7. DAX Functions – AVERAGEA

Description

Returns the average (arithmetic mean) of the values in a column. Handles text and non-numeric values.

Syntax

AVERAGEA (<column>)

Parameters

Parameter	Description
column	The column that contains the values for which you want the average.

Return Value

Returns a decimal number.

Remarks

The AVERAGEA function takes a column and averages the numbers in it and handles non-numeric data types according to the following rules:

- Values that evaluate to TRUE count as 1.
- Values that evaluate to FALSE count as 0 (zero).
- Values that contain non-numeric text count as 0 (zero).
- Empty text ("") counts as 0 (zero).

Example

=AVERAGEA (East_Sales[Sales Amount])

8. DAX Functions – AVERAGEX

Description

Calculates the average (arithmetic mean) of a set of expressions evaluated over a table.

Syntax

AVERAGEX (<table>, <expression>)

Parameters

Parameter	Description
table	Name of a table, or an expression that specifies the table over which the aggregation can be performed.
expression	An expression with a scalar result, which will be evaluated for each row of the table in the first argument.

Return Value

A decimal number.

Remarks

The AVERAGEX function enables you to evaluate expressions for each row of a table, and then take the resulting set of values and calculate its arithmetic mean. Therefore, the function takes a table as its first argument and an expression as the second argument.

In all other respects, AVERAGEX follows the same rules as AVERAGE. You cannot include non-numeric or null cells.

Example

=AVERAGEX (East_Sales,East_Sales[Unit Price]*East_Sales[No. of Units])

9. DAX Functions – COUNT

Description

Counts the number of cells in a column that contain numbers.

Syntax

COUNT (<column>)

Parameters

Parameter	Description
column	The column that contains the numbers to be counted.

Return Value

Returns a whole number.

Remarks

You can use columns containing any type of data, but only numbers are counted. The COUNT function counts the rows that contain the following kinds of values:

- Numbers
- Dates

If the row contains text that cannot be translated into a number, the row is not counted. When the function finds no rows to count, it returns a blank. When there are rows, but none of them meet the specified criteria, then the function returns 0.

Example

```
=COUNT (ProductInventory[UnitsBalance])
```

10. DAX Functions – COUNTA

Description

Counts the number of cells in a column that are not empty. It counts not just the rows that contain numeric values, but also the rows that contain nonblank values, including text, dates, and logical values.

Syntax

COUNTA (<column>)

Parameters

Parameter	Description
column	The column that contains the values to be counted.

Return Value

Returns a whole number.

Remarks

When the function does not find any rows to count, the function returns a blank. When there are rows, but none of them meet the specified criteria, then the function returns 0.

Example

```
=COUNTA (ProductInventory[UsageDate])
```

11. DAX Functions – COUNTAX

Description

Counts nonblank results when evaluating the result of an expression over a table. That is, it works just like the COUNTA function, however it is used to iterate through the rows in a table and count rows where the specified expressions result in a nonblank result.

Syntax

COUNTAX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.

Return Value

A whole number.

Remarks

The COUNTAX function counts the cells containing any type of information, including other expressions. For example, if the column contains an expression that evaluates to an empty string, the COUNTAX function treats that result as nonblank. Usually, the COUNTAX function does not count empty cells but in this case the cell contains a formula, so it is counted.

Whenever the function finds no rows to aggregate, the function returns a blank. However, if there are rows, but none of them meet the specified criteria, the function returns 0.

Example

```
Medal Count Summer Sports:=COUNTAX (
                    FILTER (Results, Results[Season]="Summer"),
                    Results[Medal])
```

12. DAX Functions – COUNTBLANK

Description

Counts the number of blank cells in a column.

Syntax

COUNTBLANK (<column>)

Parameters

Parameter	Description
column	The column that contains the blank cells to be counted.

Return Value

A whole number. If there are no blank rows, blank is returned.

Example

=COUNTBLANK(Results[Medal])

13. DAX Functions – COUNTROWS

Description

Counts the number of rows in the specified table, or in a table defined by an expression.

Syntax

COUNTROWS (<table>)

Parameters

Term	Definition
table	The name of the table that contains the rows to be counted, or an expression that returns a table.

Return Value

Returns a whole number.

Remarks

This function can be used to count the number of rows in a base table, but more often is used to count the number of rows that result from filtering a table, or applying a context to a table.

Example

=COUNTROWS (CALENDAR (DATE (2016,8,1), DATE (2016,10,31))) returns 92.

=COUNTROWS (Results) returns 34094.

=COUNTROWS (Events) returns 995.

Remarks

You can use columns containing any type of data, but only blank cells are counted. Cells that have the value zero (0) are not counted, as zero is considered a numeric value and not a blank.

Example

=COUNTBLANK (SalesTarget[SalesTarget])

14. DAX Functions – COUNTX

Description

Counts the number of rows that contain a number or an expression that evaluates to a number, when evaluating an expression over a table.

Syntax

COUNTX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows to be counted.
expression	An expression that returns the numbers you want to count.

Return Value

Returns a whole number.

Remarks

The COUNTX function counts only numeric values or dates. Parameters that are logical values or text that cannot be translated into numbers are not counted.

If the function finds no rows to count, it returns a blank. When there are rows, but none meets the specified criteria, then the function returns 0.

Example

```
=COUNTX (RELATEDTABLE (East_Sales), IF ([Product]=East_Sales[Product],1,0))
```

15. DAX Functions – CROSSJOIN

Description

Returns a table that contains the Cartesian product of all rows from all tables in the parameters. The columns in the new table are all the columns in all the parameter tables.

Syntax

CROSSJOIN (<table1>, <table2>, [<table3>] …)

Parameters

Parameter	Description
table1	Table or a DAX expression that returns a table.
table2	Table or a DAX expression that returns a table.
table3	Optional. Table or a DAX expression that returns a table.

Return Value

Returns a table that contains the Cartesian product of all rows from all tables in the parameters. The columns in the new table are all the columns in all the parameter tables.

Remarks

- Column names from table parameters must all be different in all tables or an error is returned.
- The total number of rows in the result table is the product of the number of rows from all tables in the parameters.
- The total number of columns in the result table is the sum of the number of columns from all tables in the parameters.

For example, if table1 has r1 rows and c1 columns, table2 has r2 rows and c2 columns, and table3 has r3 rows and c3 columns, then the resulting table will have -

$$r1 \times r2 \times r3 \text{ rows and } c1 + c2 + c3 \text{ columns}$$

Example

```
=CROSSJOIN (Salesperson,Products)
```

16. DAX Functions – DISTINCTCOUNT

Description

Counts the distinct values in a column.

Syntax

DISTINCTCOUNT (<column>)

Parameters

Parameter	Description
column	The column that contains the values to be counted.

Return Value

A whole number.

Remarks

You can use columns containing any type of data. When the function finds no rows to count, it returns a blank.

Example

```
=DISTINCTCOUNT (Sales[Account])
```

17. DAX Functions – GENERATE

Description

Returns a table with the Cartesian product between each row in table1 and the table that results from evaluating table2 in the context of the current row from table1.

Syntax

GENERATE (<table1>, <table2>)

Parameters

Parameter	Description
table1	Table or a DAX expression that returns a table.
table2	Table or a DAX expression that returns a table.

Return Value

A table that can be passed as a parameter to a DAX function.

Remarks

- If the evaluation of table2 for the current row in table1 returns an empty table, then the result table will not contain the current row from table1. This is different than GENERATEALL () where the current row from table1 will be included in the results, and columns corresponding to table2 will have null values for that row.

- All column names from table1 and table2 must be different or an error is returned.

Example

```
=GENERATE (

    SUMMARIZE(Salesperson,Salesperson[Salesperson]),
    SUMMARIZE(SalesTarget,SalesTarget[SalesTarget],"MaxTarget",MAX(Sales
    Target[SalesTarget])))
```

18. DAX Functions – GENERATEALL

Description

Returns a table with the Cartesian product between each row in table1 and the table that results from evaluating table2 in the context of the current row from table1.

Syntax

GENERATEALL (<table1>, <table2>)

Parameters

Parameter	Description
table1	Table or a DAX expression that returns a table.
table2	Table or a DAX expression that returns a table.

Return Value

Returns a table with the Cartesian product between each row in table1 and the table that results from evaluating table2 in the context of the current row from table1.

Remarks

- If the evaluation of table2 for the current row in table1 returns an empty table, then the current row from table1 will be included in the results, and columns corresponding to table2 will have null values for that row. This is different than GENERATE () where the current row from table1 will not be included in the results in such a case.

- All column names from table1 and table2 must be different or an error is returned.

Example

```
=GENERATEALL (

    SUMMARIZE(Salesperson,Salesperson[Salesperson]),
    SUMMARIZE(SalesTarget,SalesTarget[SalesTarget],"MaxTarget",MAX(Sales
    Target[SalesTarget])))
```

19. DAX Functions – MAX

Description

Returns the largest numeric value in a column.

Syntax

MAX (<column>)

Parameters

Parameter	Description
column	The column in which you want to find the largest numeric value.

Return Value

A decimal number.

Remarks

The following types of values in the column are considered:

- Numbers
- Dates

Empty cells, logical values, and text are ignored.

Example

```
=MAX (Sales[Sales Amount])
```

20. DAX Functions – MAXA

Description

Returns the largest value in a column.

Syntax

MAXA (<column>)

Parameters

Parameter	Description
column	The column in which you want to find the largest value.

Return Value

Returns a decimal number.

Remarks

The MAXA function takes as argument a column, and looks for the largest value among the following types of values:

- Numbers
- Dates
- Logical values, such as TRUE and FALSE. Rows that evaluate to TRUE count as 1 and rows that evaluate to FALSE count as 0 (zero).

Empty cells are ignored. If the column contains no values that can be used, MAXA returns 0 (zero).

Example

```
=MAXA (ProductInventory[UsageDate])
```

21. DAX Functions – MAXX

Description

Evaluates an expression for each row of a table and returns the largest numeric value.

Syntax

MAXX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table that returns a numeric value.

Return Value

Returns a decimal number.

Remarks

Of the values to evaluate, only the following are counted:

- Numbers. If the expression does not evaluate to a number, MAXX returns 0 (zero).
- Dates.

Empty cells, logical values, and text values are ignored.

Example

=MAXX (East_Sales,East_Sales[No. of Units]*East_Sales[Unit Price])

22. DAX Functions – MIN

Description

Returns the smallest numeric value in a column.

Syntax

MIN (<column>)

Parameters

Parameter	Description
column	The column in which you want to find the smallest numeric value.

Return Value

A decimal number.

Remarks

The following types of values in the column are considered:

- Numbers
- Dates

Empty cells, logical values and text are ignored.

Example

```
=MIN (Sales[Sales Amount])
```

23. DAX Functions – MINA

Description

Returns the smallest value in a column, including any logical values and numbers represented as text.

Syntax

MINA (<column>)

Parameters

Parameter	Description
column	The column in which you want to find the minimum value.

Return Value

Returns a decimal number.

Remarks

The following types of values in the column are considered:

- Numbers
- Dates
- Text that can be converted to numeric values
- Logical values, such as TRUE and FALSE are treated as 1 if TRUE and 0 (zero) if FALSE.

Empty cells are ignored. If the column contains no numeric values, MINA returns 0 (zero).

Example

=MINA (ProductInventory[InventoryDate])

24. DAX Functions – MINX

Description

Returns the smallest numeric value that results from evaluating an expression for each row of a table.

Syntax

MINX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.

Return Value

Returns a decimal number.

Remarks

The MINX function evaluates the results of the expression according to the following rules:

- Only numbers are counted. If the expression does not result in a number, MINX returns 0 (zero).
- Empty cells, logical values and text values are ignored. Numbers represented as text are treated as text.

Example

```
=MINX (East_Sales,East_Sales[No. of Units]*East_Sales[Unit Price])
```

25. DAX Functions – PRODUCT

Description

Returns the product of the numbers in a column.

DAX PRODUCT function is new in Excel 2016.

Syntax

PRODUCT (<column>)

Parameters

Parameter	Description
column	The column that contains the numbers for which the product is to be computed.

Return Value

A decimal number.

Remarks

Only the numbers in the column are considered. Blanks, logical values, and text are ignored.

Example

=PRODUCT (ProductInventory[InventoryDuration])

26. DAX Functions – PRODUCTX

Description

Returns the product of the numbers resulted from an expression evaluated for each row in a table.

DAX PRODUCTX function is new in Excel 2016.

Syntax

PRODUCTX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.

Return Value

A decimal number.

Remarks

Only the numbers in the column are considered. Blanks, logical values, and text are ignored.

Example

= [PresentValue] * PRODUCTX (AnnuityPeriods, 1 + [FixedInterestRate])

27. DAX Functions – ROW

Description

Returns a table with a single row containing values that result from the expressions given to each column.

Syntax

ROW (<name>, <expression>, [<name>, <expression>] …)

Parameters

Parameter	Description
name	The name given to the column, enclosed in double quotes.
expression	Any DAX expression that returns a single scalar value to populate the column - name.

Return Value

A single row table.

Remarks

Parameters must always come in pairs of name and expression.

Example

```
=ROW (Total Number of Products, COUNTA (Products, Products[Product_key]),
        Total Sales Value, SUM (Sales, Sales[ExtendedAmount]))
```

28. DAX Functions – SELECTCOLUMNS

Description

Adds calculated columns to the given table or table expression. DAX SELECTCOLUMNS function is new in Excel 2016.

Syntax

SELECTCOLUMNS (<table>, <name>, <scalar_expression>,

[<name>, < scalar_expression>] …)

Parameters

Parameter	Description
table	Table or a DAX expression that returns a table.
name	The name given to the column, enclosed in double quotes.
scalar_expression	DAX expression that returns a scalar value like a column reference, integer or string value.

Return Value

A table with the same number of rows as the table specified as the first parameter. The returned table has one column for each pair of name and scalar_expression parameters. Each scalar_expression is evaluated in the context of a row from the specified table parameter.

Remarks

SELECTCOLUMNS is similar to ADDCOLUMNS, and has the same behavior except that instead of starting with the table specified, SELECTCOLUMNS start with an empty table before adding columns.

Example

```
=SELECTCOLUMNS (
        Products,"Product-NoOfUnits",Products[Product]&"  -  "&Products[Units
        Sold])
```

29. DAX Functions – SUM

Description

Returns the sum of all the numbers in a column.

Syntax

SUM (<column>)

Parameters

Parameter	Description
column	The column that contains the numbers to sum.

Return Value

A decimal number.

Remarks

If any rows contain non-numeric values, blanks are returned.

Example

```
=SUM ([Sales Amount])
```

30. DAX Functions – SUMMARIZE

Description

Returns a summary table for the requested totals over a set of groups.

Syntax

SUMMARIZE (<table>, <groupBy_columnName>, [<groupBy_columnName>] …,

[<name>, <expression>] …)

Parameters

Parameter	Description
table	Any DAX expression that returns a table of data.
groupBy_columnName	The qualified name of an existing column to be used to create summary groups based on the values found in it. This parameter cannot be an expression.
name	The name given to a total or summarize column, enclosed in double quotes.
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).

Return Value

A table with the selected columns for the groupBy_columnName parameters and the summarized columns designed by the name parameters.

Remarks

- Each column for which you define a name must have a corresponding expression. Otherwise, an error is returned. The first parameter, 'name' defines the name of the column in the results. The second parameter, 'expression' defines the calculation performed to obtain the value for each row in that column.

- groupBy_columnName must be either in table or in a related table to table.

- Each name must be enclosed in double quotation marks.
- The function groups a selected set of rows into a set of summary rows by the values of one or more groupBy_columnName columns. One row is returned for each group.

Example

```
=SUMMARIZE (
        SalesTarget,SalesTarget[SalesTarget],"MaxTarget",MAX
        (SalesTarget[SalesTarget]))
```

31. DAX Functions – SUMMARIZE

Description

Read DAX SUMMARIZE function before reading this variant.

You have the following advanced options that you can use within SUMMARIZE function:

- ROLLUP function
- ROLLUPGROUP function
- ISSUBTOTAL function

When you use these functions within SUMMARIZE function, you will get different results.

- If you use ROLLUP function or ROLLUPGROUP function, the behavior of the SUMMARIZE function is modified by adding roll-up rows to the result on the groupBy_columnName columns.

- If you use ROLLUPGROUP function within ROLLUP function, you can prevent partial subtotals in roll-up rows.

- If you use ISSUBTOTAL function within expression part of SUMMARIZE function, you will create another column with logical values returned by ISSUBTOTAL in the resulting table. The value will be TRUE, if the row contains sub-total values for the column given as parameter to ISSUBTOTAL function. FALSE, otherwise.

Syntax

```
SUMMARIZE (<table>, <groupBy_columnName>, [<groupBy_columnName>] …,
                [ROLLUP (<groupBy_columnName>, [< groupBy_columnName> …])],
                [<name>, <expression>] …)
SUMMARIZE (<table>, <groupBy_columnName>, [<groupBy_columnName>] …,
        [ROLLUPGROUP (<groupBy_columnName>, [< groupBy_columnName> …])],
        [<name>, <expression>] …)
SUMMARIZE (<table>, <groupBy_columnName>, [<groupBy_columnName>] …,
[ROLLUP (ROLLUPGROUP (<groupBy_columnName>, [< groupBy_columnName>] …))],
[<name>, <expression>] …)

SUMMARIZE (<table>, <groupBy_columnName>, [<groupBy_columnName>] …,
        [ROLLUP (<groupBy_columnName>, [< groupBy_columnName> …])],
[<name>, {<expression> | ISSUBTOTAL (<columnName>)}] …)
```

Parameters (ROLLUP / ROLLUPGROUP Function)

Parameter	Description
groupBy_columnName	The qualified name of an existing column to be used to create summary groups based on the values found in it. This parameter cannot be an expression.

Parameters (ISSUBTOTAL Function)

Parameter	Description
columnName	The name of any column in the table of the SUMMARIZE function or any column in a related table to table.

The other parameters for SUMMARIZE function are as explained in DAX SUMMARIZE Function.

Return Value

A table with the selected columns for the groupBy_columnName parameters and the summarized columns designed by the name parameters and additionally, the roll-up rows to the groupBy_columnName columns. Subtotals are not displayed if ROLLUPGROUP is used within ROLLUP.

If SUBTOTAL function is used –

An additional column, with TRUE if the row contains a sub-total value for the column given as parameter, with FALSE, otherwise.

Remarks

The columns mentioned in the ROLLUP function cannot be referenced as groupBy_columnName parameters of SUMMARIZE function.

ROLLUP function can be used only as a parameter for SUMMARIZE function and nowhere else.

ROLLUPGROUP function can be used only as a parameter for the following and nowhere else.

- SUMMARIZE function, or
- ROLLUP function

ISSUBTOTAL function can only be used in the expression part of SUMMARIZE function.

ISSUBTOTAL must be preceded by a matching name column.

Example - ROLLUP

```
=SUMMARIZE (

        SalesTarget, ROLLUP (SalesTarget[SalespersonID]),
        SalesTarget[SalesTarget],"MaxTarget",MAX (SalesTarget[SalesTarget]))
```

Example – ROLLUP with ROLLUPGROUP

```
=SUMMARIZE (

        SalesTarget,ROLLUP (ROLLUPGROUP (SalesTarget[SalespersonID])),
        SalesTarget[SalesTarget],"MaxTarget",MAX(SalesTarget[SalesTarget]))
```

Example - ISSUBTOTAL

```
=SUMMARIZE (

        SalesTarget,ROLLUP (ROLLUPGROUP (SalesTarget[SalespersonID])),
        SalesTarget[SalesTarget],"MaxTarget",MAX (SalesTarget[SalesTarget]),

        "IsSubTotalSalesTarget", ISSUBTOTAL (SalesTarget[SalesTarget]))
```

32. DAX Functions – SUMX

Description

Returns the sum of an expression evaluated for each row in a table.

Syntax

SUMX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.

Return Value

A decimal number.

Remarks

Only the numbers in the column that results by evaluating the expression are counted.

Blanks, logical values, and text are ignored.

Example

USA Gold Medal
Count:=SUMX(Results,IF(AND([Country]="USA",[Medal]="Gold")=TRUE(),1,0))

33. DAX Functions – TOPN

Description

Returns the top specified number of rows of the table.

Syntax

TOPN (<n_value>, <table>, <orderBy_expression>, [<order>],

[<orderBy_expression>, [<order>]] …)

Parameters

Parameter	Description
n_value	The number of rows to return. It is any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).
table	Any DAX expression that returns a table of data from where to extract the top n_value number of rows.
orderBy_expression	Any DAX expression where the result value is used to sort the table and it is evaluated for each row of table.
order	Optional. A value that specifies how to sort orderBy_expression values, ascending or descending: 0 (zero) or FALSE: Sorts in a descending order of values of orderBy_expression. 1 or TRUE: Sorts in an ascending order of orderBy_expression. If omitted, default is 0.

Return Value

- Returns a table with the top n_value number of rows of table, if n_value > 0.
- Returns an empty table, if n_value <= 0.

Rows are not necessarily sorted in any particular order.

Remarks

- If there is a tie, in orderBy_expression values, at the Nth row of the table, then all tied rows are returned. The function might return more than n_value number of rows.

- TOPN does not guarantee any sort order for the results.

Example

```
=SUMX (TOPN (15,Sales,Sales[Salesperson],ASC),Sales[Sales Amount])
```

DAX Filter Functions

34. DAX Filter Functions – Overview

You can use DAX Filter functions to return specific data types, look up values in related tables and filter by related values. Lookup functions work by using tables and relationships between them. Filter functions enable you to manipulate data context to create dynamic calculations.

Following are the DAX Filter functions:

- DAX ADDMISSINGITEMS function
- DAX ALL function
- DAX ALLEXCEPT function
- DAX ALLNOBLANKROW function
- DAX ALLSELECTED function
- DAX CALCULATE function
- DAX CALCULATETABLE function
- DAX CROSSFILTER function
- DAX DISTINCT function
- DAX EARLIER function
- DAX EARLIEST function
- DAX FILTER function
- DAX FILTERS function
- DAX HASONEFILTER function
- DAX HASONEVALUE function
- DAX ISCROSSFILTERED function
- DAX ISFILTERED function
- DAX KEEPFILTERS function
- DAX RELATED function
- DAX RELATEDTABLE function
- DAX USERELATIONSHIP function
- DAX VALUES function

35. DAX Functions – ADDMISSINGITEMS

Description

Adds combinations of items from multiple columns to a table if they do not already exist. The determination of which item combinations to add is based on referencing source columns which contain all the possible values for the columns.

DAX ADDMISSINGITEMS function is new in Excel 2016.

Syntax

ADDMISSINGITEMS (<showAllColumn>, [<showAllColumn>] …, <table>, <groupingColumn>, [<groupingColumn>] …, [<filterTable>] …)

ADDMISSINGITEMS (<showAllColumn>, [<showAllColumn>] …, <table>, [ROLLUPISSUBTOTAL (<groupingColumn>, <isSubtotal_columnName>, [<groupingColumn>, [<isSubtotal_columnName>]] …)], [<filterTable>] …)

Parameters

Parameter	Description
showAllColumn	A column for which to return items with no data for the calculated fields used.
table	A table containing all items with data (NON EMPTY) for the calculated fields used.
groupingColumn	A column which is used to group by in the supplied table argument.
isSubtotal_columnName	A Boolean column in the supplied table argument which contains ISSUBTOTAL values for the corresponding groupingColumn column.
filterTable	A table representing filters to include in the logic for determining whether to add specific combinations of items with no data. Used to avoid having ADDMISSINGITEMS add in item combinations which are not present because they were removed by a filter.

Remarks

To determine the combinations of items from different columns to evaluate –

- AutoExist is applied for columns within the same table.
- CrossJoin is applied across different tables.

ADDMISSINGITEMS with ROLLUPGROUP

ROLLUPGROUP is used inside the ROLLUPISSUBTOTAL function to reflect ROLLUPGROUPs present in the supplied table argument.

Restrictions

- If ROLLUPISSUBTOTAL is used to define the supplied table argument or the equivalent rows and ISSUBTOTAL columns are added by some other means, ROLLUPISSUBTOTAL must be used with the same arguments within ADDMISSINGITEMS. This is also true for ROLLUPGROUP, if it is used with ROLLUPISSUBTOTAL to define the supplied table argument.

- The ADDMISSINGITEMS function requires that, if ROLLUPISSUBTOTAL is used to define the supplied table argument, ISSUBTOTAL columns corresponding to each group by column, or ROLLUPGROUP are present in the supplied table argument. Also, the names of the ISSUBTOTAL columns must be supplied in the ROLLUPISSUBTOTAL function inside ADDMISSINGITEMS and they must match the names of Boolean columns in the supplied table argument. This enables the ADDMISSINGITEMS function to identify BLANK values stemming from the fact that a row is a subtotal row from other BLANK values.

- If ROLLUPGROUP is used with ROLLUPISSUBTOTAL to define the supplied table argument, exactly one ISSUBTOTAL column name must be supplied per ROLLUPGROUP and it must match the corresponding ISSUBTOTAL column name in the supplied table argument.

The ADDMISSINGITEMS function will return BLANK values for the IsSubtotal columns of blank rows it adds.

Example

```
=ADDMISSINGITEMS (Products, FILTER (Products,Products[Product]="Air Purifier"))
```

36. DAX Functions – ALL

Description

Returns all the rows in a table or all the values in a column, ignoring any filters that might have been applied. This function is useful for clearing filters and creating calculations on all the rows in a table.

Syntax

ALL ({<table> | <column>, [<column>], [<column>] …})

Parameters

Parameter	Description
table	The table that you want to clear filters on.
column	The column that you want to clear filters on.

The argument to the ALL function must be either a reference to a base table or one or more references to base columns. You cannot use table expressions or column expressions with the ALL function.

Return Value

The table or column or columns with filters removed.

Remarks

ALL function is not used by itself, but serves as an intermediate function that can be used to change the set of results over which some other calculation is performed.

Example

```
=COUNTA (Results[Medal])/CALCULATE (COUNTA (Results[Medal], ALL (Results))
```

With this DAX formula, all the rows in the Results table are taken into account in the CALCULATE function with the filter containing the ALL function. This way, you have the total count in the denominator.

37. DAX Functions – ALLEXCEPT

Description

Removes all context filters in the table except filters that have been applied to the specified columns.

Syntax

ALLEXCEPT (<table>, <column>, [<column>] …)

Parameters

Parameter	Description
table	The table over which all context filters are removed, except filters on those columns that are specified in subsequent arguments.
column	One or more columns that are specified for which context filters must be preserved.

For the ALLEXCEPT function, the first argument must be a reference to a base table. All the subsequent arguments must be references to base columns in that table.

You cannot use table expressions or column expressions with the ALLEXCEPT function.

Return Value

A table with all filters removed except for the filters on the specified columns.

Remarks

ALLEXCEPT function is not used by itself, but serves as an intermediate function that can be used to change the set of results over which some other calculation is performed.

You can use ALLEXCEPT function if you want to remove the filters on many, but not all, columns in a table.

Example

```
=CALCULATE (COUNTA (Results[Medal]), ALLEXCEPT (Hosts, Hosts[City]))
```

The values in Medal column in the Results table are counted with all the filters removed, except for the filters on the Column City in the Hosts table.

38. DAX Functions – ALLNOBLANKROW

Description

Returns all rows but the blank row, or all distinct values of a column but the blank row from the parent table of a relationship and disregards any context filters that might exist.

Syntax

ALLNOBLANKROW (<table>|<column>)

Parameters

Parameter	Description
table	The table over which all context filters are removed.
column	The column over which all context filters are removed.

ALLNOBLANKROW takes only one argument, either table or column.

Return Value

- A table, when the argument is a table.
- A column of values, when the argument is a column.

Remarks

ALLNOBLANKROW function does not consider truly blank rows in a table, but only handles the blank row that is a special case generated in a parent table, when one or more of the child tables in the relationship contain non-matching values or blank values.

Example

```
=COUNTROWS (ALLNOBLANKROW (Salesperson))
```

This DAX formula returns 7, if the number of rows in the parent Salesperson table is 7. However, there are entries in the Sales table for an unaccounted salesperson (i.e. the salesperson is not present in the Salesperson table).

```
=COUNTROWS (ALL (Salesperson))
```

This DAX formula returns 8 though the number of rows in the parent Salesperson table is 7, as there are entries in the Sales table for an unaccounted salesperson (i.e. the salesperson is not present in the Salesperson table).

39. DAX Functions – ALLSELECTED

Description

ALLSELECTED function gets the context that represents all rows and columns in the query, while keeping explicit filters and contexts other than row and column filters.

This function can be used to obtain visual totals in queries.

Syntax

ALLSELECTED ([<tableName> | <columnName>])

Parameters

Parameter	Description
tableName	Optional. The name of a table. It cannot be an expression.
columnName	Optional. The name of a column, usually fully qualified. It cannot be an expression.

Return Value

The context of the query without any column and row filters.

Remarks

- ALLSELECTED function takes one or no arguments.
- If there is one argument, the argument is either tableName or columnName.
- This function is different from the function ALL () because it retains all filters explicitly set within the query, and it retains all context filters other than row and column filters.

Example

SumTotal:=CALCULATE (SUM (Sales[Sales Amount]),ALLSELECTED ())

40. DAX Functions – CALCULATE

Description

Evaluates an expression in a context that is modified by the specified filters.

Syntax

CALCULATE (<expression>, [<filter1>], [<filter2>] …)

Parameters

Parameter	Description
expression	The expression to be evaluated.
filter1, filter2, …	Optional. A comma separated list of Boolean expressions or a table expression that defines a filter.

Return Value

The value that is the result of the expression.

Remarks

The expression used as the first parameter is essentially the same as a calculated field.

If Boolean expressions are used as arguments, the following restrictions apply:

- An expression cannot reference a calculated field.
- An expression cannot use a nested CALCULATE function.
- An expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

If the data has been filtered, the CALCULATE function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter argument, any existing filters on that column are removed, and the filter used in the filter argument is applied instead.

Example

=COUNTA (Results[Medal])/CALCULATE (COUNTA (Results[Medal], ALL (Results))

41. DAX Functions – CALCULATETABLE

Description

Evaluates a table expression in a context modified by the given filters.

Syntax

CALCULATETABLE (<expression>, [<filter1>], [<filter2>] …)

Parameters

Term	Definition
expression	The table expression to be evaluated.
filter1, filter2 …	A Boolean expression or a table expression that defines a filter.

Return Value

A table of values.

Remarks

The expression used as the first parameter must be a function that returns a table.

If Boolean expressions are used as arguments, the following restrictions apply:

- The expression cannot reference a calculated field.
- The expression cannot use a nested CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

CALCULATETABLE function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter argument, any existing filters on that column are removed, and the filter used in the filter argument is applied instead.

CALCULATETABLE function is a synonym for the RELATEDTABLE function.

Example

```
=SUMX (
        CALCULATETABLE (East_Sales,
                FILTER (East_Sales, East_Sales[Product]=[Product])),
        East_Sales[Sales Amount])
```

42. DAX Functions – CROSSFILTER

Description

Specifies the cross-filtering direction to be used in a calculation for a relationship that exists between two columns.

DAX CROSSFILTER function is new in Excel 2016.

Syntax

CROSSFILTER (<columnName1>, <columnName2>, <direction>)

Parameters

Parameter	Description
columnName1	The name of a column, fully qualified, that usually represents the many side or data (fact) table side of the relationship to be used. If the arguments are given in a reverse order, the function will swap them before using them. This argument cannot be an expression.
columnName2	The name of a column, fully qualified, that usually represents the one side or lookup table side of the relationship to be used. If the arguments are given in a reverse order, the function will swap them before using them. This argument cannot be an expression.
direction	The cross-filter direction to be used: • One - Filters on one or lookup table side of the relationship filter with many side. • Both - Filters on either side filter the other. • None - No cross-filtering occurs along this relationship.

Return Value

DAX CROSSFILTER function does not return any value.

DAX CROSSFILTER function only sets the cross-filtering direction for the indicated relationship, for the duration of the query.

Remarks

- In the case of a 1:1 relationship, there is no difference between one and both direction.

- CROSSFILTER can be used only in functions that takes a filter as an argument. For example, CALCULATE, CALCULATETABLE, CLOSINGBALANCEMONTH, CLOSINGBALANCEQUARTER, CLOSINGBALANCEYEAR, OPENINGBALANCEMONTH, OPENINGBALANCEQUARTER, OPENINGBALANCEYEAR, TOTALMTD, TOTALQTD and TOTALYTD functions.

- CROSSFILTER uses the existing relationships in the model, identifying relationships by their ending point columns.

- In CROSSFILTER, the cross-filtering setting of a relationship is not important. That is, whether the relationship is set to filter one, or both directions in the model does not affect the usage of the function. CROSSFILTER will override any existing cross-filtering setting.

- An error is returned if any of the columns named as an argument is not part of a relationship or the arguments belong to different relationships.

- If CALCULATE expressions are nested, and more than one CALCULATE expression contains a CROSSFILTER function, then the innermost CROSSFILTER is the one that prevails in case of a conflict or ambiguity.

Example

```
=CALCULATE (Sales[Distinct Count of Products],
                CROSSFILTER (Sales[Product],Products[Product],Both))
```

43. DAX Functions – DISTINCT

Description

Returns a one column table that contains the distinct values from the specified column. In other words, duplicate values are removed and only unique values are returned.

You need to nest the DISTINCT function within a formula, to get a list of distinct values that can be passed to another function.

Syntax

DISTINCT (<column>)

Parameters

Parameter	Description
column	The column from which unique values are to be returned. Or, an expression that returns a column.

Return Value

A column of unique values.

Remarks

The results of DISTINCT function are affected by the current filter context. For example, if you use the formula in the following example to create a calculated field, the results would change whenever the table was filtered on the column Region.

Example

```
=COUNTROWS (DISTINCT (Sales[Salesperson ID]))
```

44. DAX Functions – EARLIER Function

Description

Returns the current value of the specified column in an outer evaluation pass of the mentioned column.

Syntax

EARLIER (<column>, <number>)

Parameters

Parameter	Description
column	A column or expression that resolves to a column.
number	Optional. A positive number to the outer evaluation pass. • The next evaluation level out is represented by 1. • Two levels out is represented by 2, and so on. If omitted, default value is 1.

Return Value

The current value of row, from column, at number of outer evaluation passes.

Remarks

EARLIER is useful for nested calculations where you want to use a certain value as an input and produce calculations based on that input. In Microsoft Excel, you can do such calculations only within the context of the current row. However, in DAX you can store the value of the input and then make calculation using data from the entire table.

EARLIER is mostly used in the context of calculated columns. EARLIER succeeds if there is a row context prior to the beginning of the table scan. Otherwise, it returns an error.

Example

If you have a table Sales with sales data, you can create a calculated column with the ranks of the Sales Amount values as follows -

```
=COUNTROWS (
        FILTER (Sales,
            EARLIER (Sales[Sales Amount])<Sales[Sales Amount]))+1
```

45. DAX Functions – EARLIEST

Description

Returns the current value of the specified column in an outer evaluation pass of the specified column.

Syntax

EARLIEST (<column>)

Parameters

Parameter	Description
column	A reference to a column.

Return Value

The current value of row, from column, at the outer evaluation pass.

Remarks

The EARLIEST function is similar to EARLIER, in which you can also specify the level of recursion.

The Results of Earliest and Earlier functions will be the same, if the parameter number of Earlier function is omitted or is set to 1.

Example

If you have a table Sales with sales data, you can create a calculated column with the ranks of the Sales Amount values as follows -

```
=COUNTROWS (
                FILTER (Sales, EARLIEST (Sales[Sales Amount])<Sales[Sales Amount]))+1
```

46. DAX Functions – FILTER

Description

Returns a table that represents a subset of another table or expression.

Syntax

FILTER (<table>, <filter>)

Parameters

Parameter	Description
table	The table to be filtered. The table can also be an expression that results in a table.
filter	A Boolean expression that is to be evaluated for each row of the table.

Return Value

A table containing only the filtered rows.

Remarks

You can use DAX FILTER function to reduce the number of rows in the table that you are working with, and use only specific data in calculations.

DAX FILTER function is not used independently, but as a function that is embedded in other functions that require a table as an argument.

Example

```
Medal Count Summer Sports:=COUNTAX (
                    FILTER (Results, Results[Season]="Summer"), Results[Medal])
```

47. DAX Functions – FILTERS

Description

Returns the values that are directly applied as filters to columnName.

Syntax

FILTERS (<columnName>)

Parameters

Parameter	Description
columnName	The name of a column in a table. It cannot be an expression.

Return Value

The values that are directly applied as filters to columnName.

Example

You can find the number of direct filters that a column using the following function:

```
=COUNTROWS (FILTERS (Sales[Region]))
```

48. DAX Functions – HASONEFILTER

Description

Returns TRUE when the number of directly filtered values on columnName is one and only one. Otherwise, returns FALSE.

Syntax

HASONEFILTER (<columnName>)

Parameters

Parameter	Description
columnName	The name of an existing column, using standard DAX syntax. It cannot be an expression.

Return Value

TRUE or FALSE.

Remarks

DAX HASONEFILTER function is similar to DAX HASONEVALUE function, with the difference that HASONEFILTER works by a direct filter, while HASONEVALUE works based on cross-filters.

Example

```
=HASONEFILTER (Sales[Product])
```

49. DAX Functions – HASONEVALUE

Description

Returns TRUE when the context for columnName has been filtered down to one distinct value only. Otherwise, returns FALSE.

Syntax

HASONEVALUE (<columnName>)

Parameters

Parameter	Description
columnName	The name of a column. It cannot be an expression.

Return Value

TRUE or FALSE.

Example

=HASONEVALUE (Sales[Product])

50. DAX Functions – ISCROSSFILTERED

Description

Returns TRUE when columnName or another column in the same or related table is being filtered.

Syntax

ISCROSSFILTERED (<columnName>)

Parameters

Parameter	Description
columnName	The name of a column in a table. It cannot be an expression.

Return Value

TRUE or FALSE.

Remarks

- A column columnName is said to be cross-filtered when a filter applied to another column in the same table or in a related table affects columnName by filtering it.
- A column is said to be filtered directly when the filter or filters apply over the column.

You can use DAX ISFILTERED function to find if a column is filtered directly.

Example

=ISCROSSFILTERED (Sales)

51. DAX Functions – ISFILTERED

Description

Returns TRUE when columnName is being filtered directly.

If there is no filter on the column or if the filtering happens because a different column in the same table or in a related table is being filtered, then the DAX ISFILTERED function returns FALSE.

Syntax

ISFILTERED (<columnName>)

Parameters

Term	Definition
columnName	The name of a column in a table. It cannot be an expression.

Return Value

TRUE or FALSE.

Remarks

- A column is said to be filtered directly when the filter or filters apply over the column.
- A column columnName is said to be cross-filtered when a filter applied to another column in the same table or in a related table affects columnName by filtering it.

You can use DAX ISCROSSFILTERED function to find if a column is cross-filtered.

Example

=ISFILTERED (Sales[Product])

52. DAX Functions – KEEPFILTERS

Description

Modifies how filters are applied while evaluating a CALCULATE or CALCULATETABLE function.

Syntax

KEEPFILTERS (<expression>)

Parameters

Term	Definition
Expression	Any DAX expression.

Return Value

DAX KEEPFILTERS function does not return any value.

Remarks

You can use DAX KEEPFILTERS function within the context CALCULATE and CALCULATETABLE functions, to override the standard behavior of those functions.

When you use KEEPFILTERS, any existing filters in the current context are compared with the columns in the filter arguments, and the intersection of those arguments is used as the context for evaluating the expression.

The net effect over any one column is that both sets of arguments apply:

- The filter arguments used in CALCULATE function
- The filters in the arguments of the KEEPFILTER function.

In other words, while CALCULATE filters replace the current context, KEEPFILTERS adds filters to the current context.

Example

```
=SUMX (
        CALCULATETABLE (East_Sales,
                FILTER(East_Sales,East_Sales[Product]=[Product]),
                KEEPFILTERS(East_Sales[Product]<>"Soap")),
        East_Sales[Sales Amount])
```

53. DAX Functions – RELATED

Description

Returns a related value from another table.

Syntax

RELATED (<column>)

Parameters

Parameter	Description
column	The column that contains the values you want to retrieve.

Return Value

A single value that is related to the current row.

Remarks

DAX RELATED function requires that a relationship exists between the current table and the table with related information. When you specify the column that contains the data that you want, the function follows an existing relationship to fetch the value from the specified column in the related table.

When DAX RELATED function performs a lookup, it examines all values in the specified table regardless of any filters that may have been applied.

DAX RELATED function needs a row context. Hence, it can be used only in one of the following cases –

- A calculated column expression, where the current row context is unambiguous.
- As a nested function in an expression that uses a DAX X function, such as SUMX.

Example

```
=SUMX (FILTER (Sales, RELATED (Products[Product]) <>"Soap"), Sales[Sales Amount])
```

54. DAX Functions – RELATEDTABLE

Description

Evaluates a table expression in a context modified by the given filters.

Syntax

RELATEDTABLE (<tableName>)

Parameters

Parameter	Description
tableName	The name of an existing table. It cannot be an expression.

Return Value

A table of values.

Remarks

DAX RELATEDTETABLE function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify.

DAX RELATEDTABLE function is equivalent to DAX CALCULATETABLE function with no logical expression.

Example

```
=SUMX (RELATEDTABLE (East_Sales),East_Sales[Sales Amount])
```

55. DAX Functions – USERELATIONSHIP

Description

Specifies the relationship to be used in a specific calculation as the one that exists between columnName1 and columnName2.

Syntax

USERELATIONSHIP (<columnName1>, <columnName2>)

Parameters

Parameter	Description
columnName1	A fully qualified name of a column that represents the many side of the relationship to be used. If the parameters are given in a reverse order, the function will swap them before using them. This parameter cannot be an expression.
columnName2	A fully qualified name of a column that represents one side or lookup side of the relationship to be used. If the parameters are given in a reverse order, the function will swap them before using them. This parameter cannot be an expression.

Return Value

DAX USERELATIONSHIP function does not return any value. The function only enables the indicated relationship for the duration of the calculation.

Remarks

- USERELATIONSHIP can only be used in DAX functions that take a filter as a parameter. For example, CALCULATE, CALCULATETABLE, CLOSINGBALANCEMONTH, CLOSINGBALANCEQUARTER, CLOSINGBALANCEYEAR, OPENINGBALANCEMONTH, OPENINGBALANCEQUARTER, OPENINGBALANCEYEAR, TOTALMTD, TOTALQTD and TOTALYTD functions.

- USERELATIONSHIP uses the existing relationships in the model, identifying relationships by their ending point columns.

- In USERELATIONSHIP, the status of a relationship is not important; that is, whether the relationship is active or not does not affect the usage of the function. Even if the relationship is inactive, it will be used and overrides any other active relationships that might be present in the model but not mentioned in the function parameters.

- An error is returned if any of the columns named as a parameter is not part of a relationship or the parameters belong to different relationships.

- If multiple relationships are needed to join table A to table B in a calculation, each relationship must be indicated in a different USERELATIONSHIP function.

- If CALCULATE expressions are nested, and more than one CALCULATE expression contains a USERELATIONSHIP function, then the innermost USERELATIONSHIP is the one that prevails in case of a conflict or ambiguity.

- Up to 10 USERELATIONSHIP functions can be nested. However, your expression might have a deeper level of nesting.

Example

```
Product Sales:=CALCULATE (
                    SUM (Sales[Sales Amount]),
                        USERELATIONSHIP (Sales[Product],Products[Product])
                    )
```

56. DAX Functions – VALUES

Description

Returns a one-column table that contains the distinct values from the specified table or column. In other words, duplicate values are removed and only unique values are returned.

Syntax

VALUES (<TableNameOrColumnName>)

Parameters

Parameter	Description
TableNameOrColumnName	The table or column from which unique values are to be returned.

Return Value

A column of unique values.

Remarks

You can use DAX VALUES function as an intermediate function, nested in a formula, to get a list of distinct values that can be counted, or used to filter or sum other values.

When you use the DAX VALUES function in a context that has been filtered, such as in a PivotTable, the unique values returned by VALUES are affected by the filter.

Example

```
=COUNTROWS (VALUES (Sales[Salesperson ID]))
```

Returns the number of rows that have unique Salesperson IDs.

DAX Time Intelligence Functions

57. DAX Time Intelligence Functions – Overview

DAX Time Intelligence functions help you create calculations that support the needs of Business Intelligence analysis by enabling you to manipulate data using time periods, including days, months, quarters and years.

Following are the DAX Time Intelligence functions:

- DAX CLOSINGBALANCEMONTH function
- DAX CLOSINGBALANCEQUARTER function
- DAX CLOSINGBALANCEYEAR function
- DAX DATEADD function
- DAX DATESBETWEEN function
- DAX DATESINPERIOD function
- DAX DATESMTD function
- DAX DATESQTD function
- DAX DATESYTD function
- DAX ENDOFMONTH function
- DAX ENDOFQUARTER function
- DAX ENDOFYEAR function
- DAX FIRSTDATE function
- DAX FIRSTNONBLANK function
- DAX LASTDATE function
- DAX LASTNONBLANK function
- DAX NEXTDAY function
- DAX NEXTMONTH function
- DAX NEXTQUARTER function
- DAX NEXTYEAR function
- DAX OPENINGBALANCEMONTH function
- DAX OPENINGBALANCEQUARTER function
- DAX OPENINGBALANCEYEAR function
- DAX PARALLELPERIOD function
- DAX PREVIOUSDAY function
- DAX PREVIOUSMONTH function
- DAX PREVIOUSQUARTER function
- DAX PREVIOUSYEAR function
- DAX SAMEPERIODLASTYEAR function
- DAX STARTOFMONTH function
- DAX STARTOFQUARTER function

- DAX STARTOFYEAR function
- DAX TOTALMTD function
- DAX TOTALQTD function
- DAX TOTALYTD function

58. DAX Functions – CLOSINGBALANCEMONTH

Description

Evaluates the expression at the last date of the month in the current context.

Syntax

CLOSINGBALANCEMONTH (<expression>, <dates>, [<filter>])

Parameters

Parameter	Description
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

If the data has been filtered, the function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter parameter, any existing filters on that column are removed, and the filter used in the filter parameter is applied instead.

Example

```
Month End Inventory Value:=CLOSINGBALANCEMONTH (

        SUMX (ProductInventory,
        [UnitsBalance]*[UnitCost]),ProductInventory[InventoryDate])
```

59. DAX Functions – CLOSINGBALANCEQUARTER

Description

Evaluates the expression at the last date of the quarter in the current context.

Syntax

CLOSINGBALANCEQUARTER (<expression>, <dates>, [<filter>])

Parameters

Parameter	Description
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.

Return Value

A scalar value that represents the expression evaluated at the last date of the quarter in the current context.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value. The filter parameter can be a Boolean expression or a table expression that defines a filter.

Example

```
Quarter End Inventory Value:=CLOSINGBALANCEQUARTER (
    SUMX (ProductInventory,[UnitsBalance]*[UnitCost]),ProductInventory[InventoryDate])
```

60. DAX Functions – CLOSINGBALANCEYEAR

Description

Evaluates the expression at the last date of the year in the current context.

Syntax

CLOSINGBALANCEYEAR (<expression>, <dates>, [<filter>], [<year_end_date>])

Parameters

Parameter	Description
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.
year_end_date	Optional. A literal string with a date that defines the year end date. The default is December 31.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

The year_end_date parameter is a string literal of a date, in the locale where the workbook was created. The year portion of the date is ignored.

Example

```
Year End Inventory Value:=CLOSINGBALANCEYEAR (
    SUMX (ProductInventory,
    [UnitsBalance]*[UnitCost]),ProductInventory[InventoryDate])
```

61. DAX Functions – DATEADD

Description

Returns a table that contains a column of dates, shifted either forward or backward in time by the specified number of intervals from the dates in the current context.

Syntax

DATEADD (<dates>, <number_of_intervals>, <interval>)

Parameters

Parameter	Description
dates	A column that contains dates.
number_of_intervals	An integer that specifies the number of intervals to add to or subtract from the dates.
interval	The interval by which to shift the dates. The value for interval can be one of the following: • Year • Quarter • Month • Day

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

If the number specified for number_of_intervals parameter is positive, the dates are moved forward in time. If the number is negative, the dates are shifted back in time.

The interval parameter is an enumeration, not a set of strings. Hence, the values for interval should not be enclosed in quotation marks. Also, the values: year, quarter, month, day should be spelled in full when using them.

The result table includes only dates that are specified in the dates parameter.

Example

```
=DATEADD (ProductInventory[InventoryDate],1, YEAR)
```

62. DAX Functions – DATESBETWEEN

Description

Returns a table that contains a column of dates that begins with the start_date and continues until the end_date.

Syntax

DATESBETWEEN (<dates>, <start_date>, <end_date>)

Parameters

Parameter	Description
dates	A reference to a date/time column.
start_date	A date expression.
end_date	A date expression.

Return Value

A table containing a single column of date values.

Remarks

- If start_date is a blank date value, then start_date will be the earliest value in the dates column.
- If end_date is a blank date value, then end_date will be the latest value in the dates column.
- The dates used as the start_date and end_date are inclusive.
- If the sales occurred on October 1 and December 31 and you specify October 1 as the start date and December 31 as the end_date, then sales on October 1 and December 31 are counted.

Example

=CALCULATE (SUM (Sales[Sales Amount]), DATESBETWEEN (Sales[Date], DATE (2015,1,1), DATE (2015,3,31)))

63. DAX Functions – DATESINPERIOD

Description

Returns a table that contains a column of dates that begins with the start_date and continues for the specified number_of_intervals.

Syntax

DATESINPERIOD (<dates>, <start_date>, <number_of_intervals>, <interval>)

Parameters

Parameter	Description
dates	A column that contains dates.
start_date	A date expression.
number_of_intervals	An integer that specifies the number of intervals to add to or subtract from the dates.
interval	The interval by which to shift the dates. The value for interval can be one of the following: • year • quarter • month • day

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

If the number specified for number_of_intervals parameter is positive, the dates are moved forward in time. If the number is negative, the dates are shifted back in time.

The interval parameter is an enumeration, not a set of strings. Hence, the values for interval should not be enclosed in quotation marks. Also, the values: year, quarter, month, day should be spelled in full when using them.

The result table includes only dates that are specified in the dates parameter.

Example

```
=CALCULATE (
        SUM (Sales [Sales Amount]),
        DATESINPERIOD (Sales[Date], DATE (2015,1,1),3, MONTH))
```

64. DAX Functions – DATESMTD

Description

Returns a table that contains a column of the dates for the month to date, in the current context.

Syntax

DATESMTD (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a measure.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The year_end_date parameter is a string literal of a date, in the locale where the workbook was created. The year portion of the date is ignored.

Example

```
=CALCULATE (
            SUM (Sales [Sales Amount]), DATESMTD (Sales [Date]))
```

65. DAX Functions – DATESQTD

Description

Returns a table that contains a column of the dates for the quarter to date, in the current context.

Syntax

DATESQTD (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a measure.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
=CALCULATE (
        SUM (Sales [Sales Amount]), DATESQTD (Sales [Date]))
```

66. DAX Functions – DATESYTD

Description

Returns a table that contains a column of the dates for the year to date, in the current context.

Syntax

DATESYTD (<dates>, [<year_end_date>])

Parameters

Parameter	Description
dates	A column that contains dates.
year_end_date	Optional. A literal string with a date that defines the year-end date. If omitted, the default is December 31.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
=CALCULATE (
        SUM (Sales [Sales Amount]), DATESYTD (Sales [Date]))
```

67. DAX Functions – ENDOFMONTH

Description

Returns the last date of the month in the current context for the specified column of dates.

Syntax

ENDOFMONTH (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column and single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

=ENDOFMONTH (Sales [Date])

68. DAX ENDOFQUARTER Function

Description

Returns the last date of the quarter in the current context for the specified column of dates.

Syntax

ENDOFQUARTER (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column and single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

=ENDOFQUARTER (Sales [Date])

69. DAX Functions – ENDOFYEAR

Description

Returns the last date of the year in the current context for the specified column of dates.

Syntax

ENDOFYEAR (<dates>, [<year_end_date>])

Parameters

Parameter	Description
dates	A column that contains dates.
year_end_date	Optional. A literal string with a date that defines the year-end date. If omitted, the default is December 31.

Return Value

A table containing a single column and a single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The year_end_date parameter is a string literal of a date, in the locale where the workbook was created. The year portion of the date is ignored.

Example

```
=ENDOFYEAR (Sales [Date])
```

70. DAX Functions – FIRSTDATE

Description

Returns the first date in the current context for the specified column of dates.

Syntax

FIRSTDATE (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column and single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

When the current context is a single date, the dates returned by the DAX FIRSTDATE function and DAX LASTDATE function will be the same.

As DAX FIRSTDATE function returns a table that contains a single column and single value, it can be used as a parameter to any DAX function that requires a table in its parameters. Further, the returned value can be used wherever a date value is required.

Example

```
=FIRSTDATE (Sales [Date])
```

71. DAX Functions – FIRSTNONBLANK

Description

Returns the first value in the column filtered by the current context, where the expression is not blank.

Syntax

FIRSTNONBLANK (<column>, <expression>)

Parameters

Parameter	Description
column	A column expression.
expression	An expression evaluated for blanks for each value of column.

Return Value

A table containing a single column and single row with the computed first non-blank value.

Remarks

The column parameter can be any of the following:

- A reference to any column.
- A table with a single column.
- A Boolean expression that defines a single-column table.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

=FIRSTNONBLANK (Sales [Sales Amount], MIN (Sales [Date])>3/31/2015)

72. DAX Functions – LASTDATE Function

Description

Returns the last date in the current context for the specified column of dates.

Syntax

LASTDATE (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column and a single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

When the current context is a single date, the dates returned by the DAX FIRSTDATE function and DAX LASTDATE function will be the same.

As DAX LASTDATE function returns a table that contains a single column and single value, it can be used as a parameter to any DAX function that requires a table in its parameters. Further, the returned value can be used wherever a date value is required.

Example

```
=LASTDATE (Sales [Date])
```

73. DAX Functions – LASTNONBLANK

Description

Returns the last value in the column filtered by the current context, where the expression is not blank.

Syntax

LASTNONBLANK (<column>, <expression>)

Parameters

Parameter	Description
column	A column expression.
expression	An expression evaluated for blanks for each value of column.

Return Value

A table containing a single column and single row with the computed last non-blank value.

Remarks

The column parameter can be any of the following:

- A reference to any column.
- A table with a single column.
- A Boolean expression that defines a single-column table.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
=LASTNONBLANK (Sales [Sales Amount], MIN (Sales [Date])>3/31/2015)
```

74. DAX Functions – NEXTDAY

Description

Returns a table that contains a column of all dates from the next day, based on the first date specified in the dates column in the current context.

Syntax

NEXTDAY (<dates>)

Parameters

Parameter	Description
Dates	A column containing dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
=CALCULATE (
        SUM (Sales [Sales Amount]), NEXTDAY (Sales [Date]))
```

75. DAX Functions – NEXTMONTH

Description

Returns a table that contains a column of all dates from the next month, based on the first date in the dates column in the current context.

Syntax

NEXTMONTH (<dates>)

Parameters

Parameter	Description
dates	A column containing dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
=CALCULATE (SUM (Sales [Sales Amount]), NEXTMONTH (Sales [Date]))
```

76. DAX Functions – NEXTQUARTER

Description

Returns a table that contains a column of all dates in the next quarter, based on the first date specified in the dates column, in the current context.

Syntax

NEXTQUARTER (<dates>)

Parameters

Parameter	Description
dates	A column containing dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
=CALCULATE (SUM (Sales [Sales Amount]), NEXTQUARTER (Sales [Date]))
```

77. DAX Functions – NEXTYEAR

Description

Returns a table that contains a column of all dates in the next year, based on the first date in the dates column, in the current context.

Syntax

NEXTYEAR (<dates>, [<year_end_date>])

Parameters

Term	Definition
dates	A column containing dates.
year_end_date	Optional. A literal string with a date that defines the year-end date. If omitted, the default is December 31.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The year_end_date parameter is a string literal of a date, in the locale where the workbook was created. The year portion of the date is ignored.

Example

```
=CALCULATE (SUM (Sales [Sales Amount]), NEXTYEAR (Sales [Date]))
```

78. DAX Functions – OPENINGBALANCEMONTH

Description

Evaluates the expression at the first date of the month in the current context.

Syntax

OPENINGBALANCEMONTH (<expression>, <dates>, [<filter>])

Parameters

Parameter	Description
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

If the data has been filtered, the function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter parameter, any existing filters on that column are removed, and the filter used in the filter parameter is applied instead.

Example

```
Month Beginning Inventory Value:=OPENINGBALANCEMONTH (
        SUMX
        (ProductInventory,[UnitsBalance]*[UnitCost]),ProductInventory[InventoryDate])
```

79. DAX Functions – OPENINGBALANCEQUARTER

Description

Evaluates the expression at the first date of the quarter, in the current context.

Syntax

OPENINGBALANCEQUARTER (<expression>, <dates>, [<filter>])

Parameters

Term	Definition
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

If the data has been filtered, the function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter parameter, any existing filters on that column are removed, and the filter used in the filter parameter is applied instead.

Example

```
Quarter Beginning Inventory Value:=OPENINGBALANCEQUARTER (
        SUMX
        (ProductInventory,[UnitsBalance]*[UnitCost]),ProductInventory[InventoryDate])
```

80. DAX Functions – OPENINGBALANCEYEAR

Description

Evaluates the expression at the first date of the year in the current context.

Syntax

OPENINGBALANCEYEAR (<expression>, <dates>, [<filter>], [<year_end_date>])

Parameters

Term	Definition
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.
year_end_date	Optional. A literal string with a date that defines the year-end date. If omitted, the default is December 31.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

If the data has been filtered, the function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter parameter, any existing filters on that column are removed, and the filter used in the filter parameter is applied instead.

The year_end_date parameter is a string literal of a date, in the same locale as the locale of the client where the workbook was created. The year portion of the date is ignored.

Example

```
Year Beginning Inventory Value:=OPENINGBALANCEYEAR (

    SUMX
    (ProductInventory,[UnitsBalance]*[UnitCost]),ProductInventory[InventoryDate])
```

81. DAX Functions – PARALLELPERIOD

Description

Returns a table that contains a column of dates that represents a period parallel to the dates in the specified dates column, in the current context, with the dates shifted a number of intervals either forward in time or back in time.

Syntax

PARALLELPERIOD (<dates>, <number_of_intervals>, <interval>)

Parameters

Parameter	Description
dates	A column that contains dates.
number_of_intervals	An integer that specifies the number of intervals to add to or subtract from the dates.
interval	The interval by which to shift the dates. The value for interval can be one of the following: • Year • Quarter • Month • Day

Return Value

A table containing a single column of date values.

Remarks

DAX PARALLELPERIOD function takes the current set of dates in the column specified by dates, shifts the first date and the last date the specified number of intervals, and then returns all contiguous dates between the two shifted dates.

If the interval is a partial range of month, quarter, or year, then any partial months in the result are also filled out to complete the entire interval.

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

If the number specified for number_of_intervals parameter is positive, the dates are moved forward in time. If the number is negative, the dates are shifted back in time.

The interval parameter is an enumeration, not a set of strings. Hence, the values for interval should not be enclosed in quotation marks. Also, the values: year, quarter, month, day should be spelled in full when using them.

The result table includes only dates that are specified in the dates parameter.

If the dates in the current context do not form a contiguous interval, the function returns an error.

Example

```
Previous Year Sales:=CALCULATE (
                                    SUM (Sales[Sales Amount]),
                                    PARALLELPERIOD (Sales[Date], -1,YEAR))
```

82. DAX Functions – PREVIOUSDAY

Description

Returns a table that contains a column of all dates representing the day that is previous to the first date in the dates column, in the current context.

Syntax

PREVIOUSDAY (<dates>)

Parameters

Parameter	Description
dates	A column containing dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
Previous Day Sales:=CALCULATE (
                    SUM (Sales[Sales Amount]),PREVIOUSDAY (Sales[Date]))
```

83. DAX Functions – PREVIOUSMONTH

Description

Returns a table that contains a column of all dates from the previous month, based on the first date in the dates column, in the current context.

Syntax

PREVIOUSMONTH (<dates>)

Parameters

Parameter	Description
dates	A column containing dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a measure.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
Previous Month Sales:=CALCULATE (
        SUM (Sales[Sales Amount]), PREVIOUSMONTH (Sales[Date]))
```

84. DAX Functions – PREVIOUSQUARTER

Description

Returns a table that contains a column of all dates from the previous quarter, based on the first date in the dates column, in the current context.

Syntax

PREVIOUSQUARTER (<dates>)

Parameters

Parameter	Description
dates	A column containing dates.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a measure.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
Previous Quarter Sales:= =CALCULATE (
        SUM (Sales [Sales Amount]),PREVIOUSQUARTER (Sales [Date]))
```

85. DAX Functions – PREVIOUSYEAR

Description

Returns a table that contains a column of all dates from the previous year, given the last date in the dates column, in the current context.

Syntax

PREVIOUSYEAR (<dates>, [<year_end_date>])

Parameters

Parameter	Description
dates	A column containing dates.
year_end_date	Optional. A literal string with a date that defines the year-end date. If omitted, the default is December 31.

Return Value

A table containing a single column of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a measure.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The year_end_date parameter is a string literal of a date, in the locale where the workbook was created. The year portion of the date is ignored.

Example

```
Previous Year Sales:=CALCULATE (
        SUM (Sales [Sales Amount]),PREVIOUSYEAR (Sales [Date]))
```

86. DAX Functions – SAMEPERIODLASTYEAR

Description

Returns a table that contains a column of dates shifted one year back in time from the dates in the specified dates column, in the current context.

Syntax

SAMEPERIODLASTYEAR (<dates>)

Parameters

Parameter	Description
dates	A column containing dates.

Return Value

A single column table of date values.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
Previous Year Sales:=CALCULATE (
        SUM (Sales [Sales Amount]),SAMEPERIODLASTYEAR (Sales [Date]))
```

87. DAX Functions – STARTOFMONTH

Description

Returns the first date of the month in the current context for the specified column of dates.

Syntax

STARTOFMONTH (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column and single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

```
=STARTOFMONTH (Sales[Date])
```

88. DAX Functions – STARTOFQUARTER

Description

Returns the first date of the quarter in the current context for the specified column of dates.

Syntax

STARTOFQUARTER (<dates>)

Parameters

Parameter	Description
dates	A column that contains dates.

Return Value

A table containing a single column and single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

Example

=STARTOFQUARTER (Sales [Date])

89. DAX Functions – STARTOFYEAR

Description

Returns the first date of the year in the current context for the specified column of dates.

Syntax

STARTOFYEAR (<dates>, [<Year_end_date>])

Parameters

Parameter	Description
dates	A column that contains dates.
Year_end_date	Optional. A year end date value. If omitted, default is 31st December.

Return Value

A table containing a single column and a single row with a date value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The year_end_date parameter is a string literal of a date, in the locale where the workbook was created. The year portion of the date is ignored.

Example

=STARTOFYEAR (Sales [Date])

90. DAX Functions – TOTALMTD

Description

Evaluates the value of the expression for the month to date, in the current context.

Syntax

TOTALMTD (<expression>, <dates>, [<filter>])

Parameters

Term	Definition
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

If the data has been filtered, the function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter parameter, any existing filters on that column are removed, and the filter used in the filter parameter is applied instead.

Example

Month Running Sum:=TOTALMTD (SUM (Sales[Sales Amount]),Sales[Date])

91. DAX Functions – TOTALQTD

Description

Evaluates the value of the expression for the dates in the quarter to date, in the current context.

Syntax

TOTALQTD (<expression>, <dates>, [<filter>])

Parameters

Term	Definition
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

If the data has been filtered, the function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter parameter, any existing filters on that column are removed, and the filter used in the filter parameter is applied instead.

Example

Quarter Running Sum:=TOTALQTD (SUM (Sales[Sales Amount]),Sales[Date])

92. DAX Functions – TOTALYTD

Description

Evaluates the year-to-date value of the expression in the current context.

Syntax

TOTALYTD (<expression>, <dates>, [<filter>], [<year_end_date>])

Parameters

Parameter	Description
expression	An expression that returns a scalar value.
dates	A column that contains dates.
filter	Optional. An expression that specifies a filter to apply to the current context.
year_end_date	Optional. A literal string with a date that defines the year-end date. If omitted, the default is December 31.

Return Value

A scalar value.

Remarks

The dates parameter can be any of the following:

- A reference to a date/time column.
- A table expression that returns a single column of date/time values.
- A Boolean expression that defines a single-column table of date/time values.

Constraints on Boolean expressions:

- The expression cannot reference a calculated field.
- The expression cannot use CALCULATE function.
- The expression cannot use any function that scans a table or returns a table, including aggregation functions.

However, a Boolean expression can use any function that looks up a single value, or that calculates a scalar value.

The filter parameter can be a Boolean expression or a table expression that defines a filter.

If the data has been filtered, the function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter parameter, any existing filters on that column are removed, and the filter used in the filter parameter is applied instead.

The year_end_date parameter is a string literal of a date, in the locale where the workbook was created. The year portion of the date is ignored.

Example

Year Running Sum:=TOTALYTD (SUM (Sales[Sales Amount]),Sales[Date])

DAX Date and Time Functions

93. DAX Date and Time Functions – Overview

DAX Date and Time functions are similar to the Excel date and time functions. However, DAX date and time functions are based on the DAX datetime data type.

Following are the DAX Date and Time functions:

- DAX CALENDAR function
- DAX CALENDARAUTO function
- DAX DATE function
- DAX DATEDIFF function
- DAX DATEVALUE function
- DAX DAY function
- DAX EDATE function
- DAX EOMONTH function
- DAX HOUR function
- DAX MINUTE function
- DAX MONTH function
- DAX NOW function
- DAX SECOND function
- DAX TIME function
- DAX TIMEVALUE function
- DAX TODAY function
- DAX WEEKDAY function
- DAX WEEKNUM function
- DAX YEAR function
- DAX YEARFRAC function

94. DAX Functions – CALENDAR

Description

Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is from the specified start date to the specified end date, inclusive of those two dates.

DAX CALENDAR function is new in Excel 2016.

Syntax

CALENDAR (<start_date>, <end_date>)

Parameters

Parameter	Description
start_date	Any DAX expression that returns a datetime value.
end_date	Any DAX expression that returns a datetime value.

Return Value

Returns a table with a single column named "Date" containing a contiguous set of dates.

Remarks

An error is returned if start_date is greater than end_date.

Example

=COUNTROWS (CALENDAR (DATE (2016,8,1), DATE (2016,10,31))) returns 92.

95. DAX Functions – CALENDARAUTO

Description

Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is calculated automatically based on the data in the model.

DAX CALENDARAUTO function is new in Excel 2016.

Syntax

CALENDARAUTO ([<fiscal_year_end_month>])

Parameters

Parameter	Description
fiscal_year_end_month	Any DAX expression that returns an integer from 1 to 12. If omitted, defaults to the value specified in the calendar table template for the current user, if present. Otherwise, defaults to 12.

Return Value

Returns a table with a single column named "Date" that contains a contiguous set of dates. The range of dates is calculated automatically based on data in the model.

Remarks

The date range is calculated as follows:

- The earliest date in the model which is not in a calculated column or calculated table is taken as the MinDate.
- The latest date in the model which is not in a calculated column or calculated table is taken as the MaxDate.
- The date range returned is dates between the beginning of the fiscal year associated with MinDate and the end of the fiscal year associated with MaxDate.

An error is returned if the model does not contain any datetime values which are not in calculated columns or calculated tables.

Example

```
=COUNTROWS (CALENDARAUTO ())
```

96. DAX Functions – DATE

Description

Returns the specified date in datetime format.

Syntax

DATE (<year>, <month>, <day>)

Parameters

Parameter	Description
year	A number representing the year. The value of the year argument can include one to four digits. The year argument is interpreted according to the date system used by your computer. Dates beginning with March 1, 1900 are supported. If you enter a number that has decimal places, the number is rounded. For values greater than 9999 or less than zero (negative values), the function returns a #VALUE! error. If the year value is between 0 and 1899, the value is added to 1900 to produce the final value. **Note**: You should use four digits for the year argument whenever possible to prevent unwanted results. For example, using 15 for 2015 returns 1915 as the year value, which is not the case.
month	A number representing the month or a calculation according to the following rules: If month is a number from 1 to 12, then it represents a month of the year. 1 represents January, 2 represents February, and so on until 12 that represents December. If you enter an integer larger than 12, the following computation occurs: The date is calculated by adding the value of month to the year. For example, if you have DATE (2015, 19, 1), the function returns a

	datetime value equivalent to July 1st of 2016, because 19 months are added to the beginning of 2015, yielding a value of July 2016. If you enter a negative integer, the following computation occurs: The date is calculated subtracting the value of month from the year. For example, if you have DATE(2015, -6, 15), the function returns a datetime value equivalent to June 15th of 2014, because when 6 months are subtracted from the beginning of 2015 it yields a value of June 2014.
day	A number representing the day or a calculation according to the following rules: If day is a number from 1 to the last day of the given month then it represents a day of the month. If you enter a number larger than the last day of the given month, the following computation occurs: The date is calculated by adding the value of day to month. For example, in the formula DATE(2016, 8, 45), the DATE function returns a datetime value equivalent to September 15th of 2016, because 45 days are added to the beginning of August yielding a value of September 15th. If you enter a negative number, the following computation occurs: The date is calculated subtracting the value of day from month. For example, in the formula DATE(2016, 5, -15), the DATE function returns a datetime value equivalent to April 15th of 2016, because 15 days are subtracted from the beginning of May 2016 yielding a value of April 2016. If day contains a decimal portion, it is rounded to the nearest integer value.

Return Value

Specified date in datetime format.

Remarks

The DATE function takes the numbers that are input as arguments and generates the corresponding date. The DATE function is most useful in situations where the year, month, and day are supplied by DAX formulas.

For e.g. the underlying data might contain dates in a format that is not recognized by DAX as a date, such as YYYYMMDD. You can use the DATE function in conjunction with other DAX functions to convert the dates to datetime format that can be recognized as a date by DAX.

DAX date functions always return a datetime data type. However, you can use formatting to display dates as serial numbers if you want.

Example

=DATE (2016,8,5) returns 8/5/2016 12:00:00 AM

=DATE (2016,8,45) returns 9/14/2016 12:00:00 AM

=DATE (2016,8, -5) returns 7/26/2016 12:00:00 AM

=DATE (2016,15,15) returns 3/15/2017 12:00:00 AM

97. DAX Functions – DATEDIFF

Description

Returns the count of interval boundaries crossed between two dates.

DAX DATEDIFF function is new in Excel 2016.

Syntax

DATEDIFF (<start_date>, <end_date>, <interval>)

Parameters

Parameter	Description
start_date	A scalar datetime value.
end_date	A scalar datetime value.
interval	The interval to use when comparing the dates. The value can be one of the following: • SECOND • MINUTE • HOUR • DAY • WEEK • MONTH • QUARTER • YEAR

Return Value

A whole number.

Remarks

If start_date is larger than end_date, an error value is returned.

The values given to the parameter interval are constants and not strings. Hence, they should not be enclosed in double quotation marks.

Example

=DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), MONTH) returns 2.

=DATEDIFF (DATE (2016,1,1), DATE (2016,4,1), MONTH) returns 3.

=DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), DAY) returns 90.

=DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), HOUR) returns 2160.

=DATEDIFF (DATE (2016,1,1), DATE (2016,3,31), SECOND) returns 7776000.

98. DAX Functions – DATEVALUE

Description

Converts a date in the form of text to a date in datetime format.

Syntax

DATEVALUE (<date_text>)

Parameters

Parameter	Description
date_text	Text that represents a date.

Return Value

A date in datetime format.

Remarks

The DATEVALUE function uses the locale and date/time settings of the client computer to understand the text value when performing the conversion.

- If the current date/time settings represent dates in the format of Month/Day/Year, then the string, "1/8/2015", would be converted to a datetime value equivalent to January 8th of 2015.
- If the current date and time settings represent dates in the format of Day/Month/Year, the same string would be converted as a datetime value equivalent to August 1st of 2015.
- If the year portion of the date_text argument is omitted, the DATEVALUE function uses the current year from your computer's built-in clock.

Time information in the date_text argument is ignored.

Example

=DATEVALUE ("1/8/2016") returns 1/8/2016 12:00:00 AM.

=DATEVALUE("8-Jan-2016") returns 1/8/2016 12:00:00 AM.

=DATEVALUE("8-Jan") returns 1/8/2016 12:00:00 AM.

99. DAX Functions – DAY

Description

Returns the day of the month, a number from 1 to 31.

Syntax

DAY (<date>)

Parameters

Parameter	Description
date	A date in datetime format or a text representation of a date.

Return Value

An integer indicating the day of the month.

Remarks

The argument to the DAY function is the Date of the day. DAX handles the date values in datetime format.

You can specify Date as one of the following –

- An output of another date function.
- An expression that returns a date.
- A date in a datetime format.
- A date as text representation in one of the accepted string formats for dates.

The DAY function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. For example,

- If the current date/time settings represent dates in the format of Month/Day/Year, then the string, "1/8/2016" is understood as a datetime value equivalent to 8th January, 2016 and the function returns 8.

- If the current date/time settings represent dates in the format of Day/Month/Year, the same string would be understood as a datetime value equivalent to 1st August, 2016, and the function returns 1.

Example

=DAY ("8-Jan") returns 8.

=DAY ("3/5/2016") returns 5.

=DAY ("March 5, 2016") returns 5.

=DAY (TODAY ()) returns 16 if TODAY () returns 12/16/2016 12:00:00 AM.

=DAY ([Date]) returns a calculated column with day values.

100. DAX Functions – EDATE

Description

Returns the date that is the indicated number of months before or after the start date.

Syntax

EDATE (<start_date>, <months>)

Parameters

Parameter	Description
start_date	A date that represents the start date. It can be in datetime or text format.
months	An integer that represents the number of months before or after start_date. If months is not an integer, it is truncated.

Return Value

A date in datetime format.

Remarks

You can use EDATE to calculate maturity dates or due dates that fall on the same day of the month as the date of issue.

DAX works with dates in datetime format. Dates stored in other formats are converted implicitly.

- If start_date is not a valid date, EDATE returns an error value.

  Make sure that the column reference or date that you supply as the first parameter is a date.

- DAX EDATE function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. For example,

  - If the current date/time settings represent dates in the format of Month/Day/Year, then the string, "1/8/2016" is understood as a datetime value equivalent to 8th January, 2016.

- If the current date/time settings represent dates in the format of Day/Month/Year, the same string would be understood as a datetime value equivalent to 1st August, 2016.

Example

=EDATE (DATE (2015,1,1),9) returns 10/1/2015 12:00:00 AM

=EDATE (DATE (2015,1,30),1) returns 2/28/2015 12:00:00 AM

=EDATE (DATE (2015,1,29),1) returns 2/28/2015 12:00:00 AM

101. DAX Functions – EOMONTH

Description

Returns the date in datetime format of the last day of the month, before or after a specified number of months.

Syntax

EOMONTH (<start_date>, <months>)

Parameters

Parameter	Description
start_date	A date that represents the start date. It can be in datetime or text format.
months	A whole number that represents the number of months before or after start_date. If months is not an integer, rounded up or down to the nearest integer.

Return Value

A date in datetime format.

Remarks

You can use EOMONTH to calculate the maturity dates or due dates that fall on the last day of the month.

DAX works with dates in datetime format. Dates stored in other formats are converted implicitly.

- If start_date is not a valid date, EOMONTH returns an error.
- If start_date plus months yields an invalid date, EOMONTH returns an error. Dates before March 1st of 1900 and after December 31st of 9999 are invalid.

- DAX EOMONTH function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. For example,
    - If the current date/time settings represent dates in the format of Month/Day/Year, then the string, "1/8/2016" is understood as a datetime value equivalent to 8th January, 2016.
    - If the current date/time settings represent dates in the format of Day/Month/Year, the same string would be understood as a datetime value equivalent to 1st August, 2016.

If the text representation of the date cannot be correctly converted to a datetime value, the function returns an error.

Example

=EOMONTH (DATE (2016,4,5),5) returns 9/30/2016 12:00:00 AM

=EOMONTH (DATE (2016,4,5),4.5) also returns 9/30/2016 12:00:00 AM, as 4.5 will be rounded up to 5.

102. DAX Functions – HOUR

Description

Returns the hour as an integer from 0 (12:00 A.M.) to 23 (11:00 P.M.).

Syntax

HOUR (<datetime>)

Parameters

Parameter	Description
datetime	A datetime value representing time. E.g. 18:15:00 or 6:48 P.M.

Return Value

An integer from 0 to 23.

Remarks

The parameter to the DAX HOUR function is the time that contains the hour you want to find.

You can specify the time as one of the following –

- An output of a date/time function.
- An expression that returns a datetime value.
- A value in one of the accepted time formats.
- An accepted text representation of a time.

The HOUR function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. Most locales use the colon (:) as the time separator and any input text using colons as time separators will parse correctly. Review your locale settings to understand your results.

Example

=HOUR ("18:15:15") returns 18.

=HOUR ("6:15:15") returns 6.

=HOUR (NOW ()) returns 9, if NOW () returns 12/16/2016 9:02:12 AM.

103. DAX Functions – MINUTE

Description

Returns the minute as an integer from 0 to 59.

Syntax

MINUTE (<datetime>)

Parameters

Parameter	Description
datetime	A datetime value representing time. E.g. 18:15:00 or 6:48 P.M.

Return Value

An integer from 0 to 59.

Remarks

The argument to the MINUTE function is the time that contains the minute you want to find.

You can specify the time as one of the following –

- Output of a date/time function.
- An expression that returns a datetime.
- A value in one of the accepted time formats.
- An accepted text representation of a time.

The MINUTE function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. Most locales use the colon (:) as the time separator and any input text using colons as time separators will parse correctly. Review your locale settings to understand your results.

Example

=MINUTE ("18:15:15") returns 15.

=MINUTE ("6:25:15")" returns 25.

=MINUTE (Now ()) returns 11 if Now () returns 12/16/2016 9:11:00 AM.

104. DAX Functions – MONTH

Description

Returns the month as a number from 1 (January) to 12 (December).

Syntax

MONTH (<datetime>)

Parameters

Parameter	Description
datetime	A date in datetime or text format.

Return Value

An integer number from 1 to 12.

Remarks

DAX uses datetime format when working with dates. Dates stored in other formats are converted implicitly.

You can enter the date used as a parameter to the MONTH function in any of the following ways -

- By typing an accepted datetime format.
- By providing a reference to a column that contains dates.
- By using an expression that returns a date.
- By using a text representation for a date.

DAX MONTH function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. For example,

- If the current date/time settings represent dates in the format of Month/Day/Year, then the string "1/8/2016" is understood as a datetime value equivalent to 8th January, 2016 and the function returns 1.

- If the current date/time settings represent dates in the format of Day/Month/Year, the same string would be understood as a datetime value equivalent to 1st August, 2016, and the function returns 8.

  If the text representation of the date cannot be correctly converted to a datetime value, the function returns an error.

Example

=MONTH ("April 5, 2016") returns 4.

=MONTH ("March 2, 2016 3:45 PM”) returns 3.

=MONTH (TODAY ()) returns 12 if TODAY () returns 12/16/2016 12:00:00 AM.

105. DAX Functions – NOW

Description

Returns the current date and time in datetime format.

Syntax

NOW ()

Parameters

No parameters for this function.

Return Value

A date in datetime format.

Remarks

DAX NOW function is useful when you need to display the current date and time on a worksheet or calculate a value based on the current date and time, and have that value updated each time you open the workbook.

DAX uses datetime format when working with dates. Dates stored in other formats are converted implicitly.

The result of NOW function changes only when the column that contains the DAX formula is refreshed. It is not updated continuously.

DAX TODAY function also returns the current date but is not precise with regard to time. The time returned is always 12:00:00 AM and only the date is updated.

Example

```
=NOW ()

=HOUR (NOW ())
```

106. DAX Functions – SECOND

Description

Returns the seconds of a time value, as a number from 0 to 59.

Syntax

SECOND (<datetime>)

Parameters

Parameter	Description
datetime	A datetime value representing time. E.g. 18:15:15 or 5:45:15 P.M.

Return Value

An integer number from 0 to 59.

Remarks

The parameter to the SECOND function is the time that contains the second you want to find.

You can specify the time as one of the following –

- Output of a date/time function.
- An expression that returns a datetime value.
- A value in one of the accepted time formats.
- An accepted text representation of a time.

DAX SECOND function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. Most locales use the colon (:) as the time separator and any input text using colons as time separators will parse correctly. Review your locale settings to understand your results.

Example

=SECOND ("18:15:45") returns 45.

=SECOND ("2:15:05") returns 5.

=SECOND (NOW ()) returns 57 if NOW () returns 12/16/2016 10:07:57 AM.

107. DAX Functions – TIME

Description

Converts hours, minutes, and seconds given as numbers to a time in datetime format.

Syntax

TIME (<hour>, <minute>, <second>)

Parameters

Parameter	Description
hour	A number from 0 to 23 representing the hour. Any value greater than 23 will be divided by 24 and the remainder will be treated as the hour value.
minute	A number from 0 to 59 representing the minute. Any value greater than 59 will be converted to hours and minutes.
second	A number from 0 to 59 representing the second. Any value greater than 59 will be converted to hours, minutes, and seconds.

Return Value

Returns the specified time in datetime format.

Remarks

DAX works with date and time values in datetime format. Numbers in other formats are implicitly converted when you use a date/time value in a DAX function.

DAX TIME function takes the integers that are input as parameters and generates the corresponding time. The TIME function is most useful in situations where the hour, minute, and second are supplied by DAX formulas.

Time values are a portion of a date value, and in the serial number system are represented by a decimal number. Hence, the datetime value 12:00 PM is equivalent to 0.5, because it is half of a day. You can use DAX TIME function in conjunction with other DAX functions to convert the numbers to a format that can be recognized as a time.

Example

=TIME (2,90,30) returns 12/30/1899 3:30:30 AM.

=TIME (12, 30, 0) returns 12/30/1899 12:30:00 PM

108. DAX Functions – TIMEVALUE

Description

Converts time in text format to time in datetime format.

Syntax

TIMEVALUE (<time_text>)

Parameters

Parameter	Description
time_text	A text string that represents a certain time of the day. Any date information included in the time_text parameter is ignored.

Return Value

A date in datetime format.

Remarks

When the time_text parameter is a text representation of the date and time, DAX TIMEVALUE function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. Most locales use the colon (:) as the time separator, and any input text using colons as time separators will parse correctly.

Review your locale settings to understand your results.

Example

=TIMEVALUE ("14:45:35") returns 12/30/1899 2:45:35 PM.

=TIMEVALUE ("2:35:55") returns 12/30/1899 2:35:55 AM.

109. DAX Functions – TODAY

Description

Returns the current date.

Syntax

TODAY ()

Parameters

No parameters for this function.

Return Value

Current date in datetime format.

Remarks

DAX TODAY function is useful when you need to have the current date displayed on a workbook, regardless of when you open the workbook. It is also useful for calculating intervals.

DAX functions - TODAY and NOW both return the current date. However,

- TODAY returns the time as 12:00:00 always.
- NOW returns the time precisely.

Example

=YEAR (TODAY () – [JoiningDate]) – 1900 returns the number of years of service for each employee.

110. DAX Functions – WEEKDAY

Description

Returns a number identifying the day of the week of a date.

Syntax

WEEKDAY (<date>, [<return_type>])

Parameters

Parameter	Description
date	A date in datetime format.
return_type	Optional. A number that determines the return value: 1 - Week begins on Sunday (1) and ends on Saturday (7), numbered 1 through 7. 2 - Week begins on Monday (1) and ends on Sunday (7), numbered 1 through 7. 3 - Week begins on Monday (0) and ends on Sunday (6), numbered 0 through 6. If omitted, default is 1.

Return Value

An integer from 0 to 7, based on the return type.

Remarks

The first parameter to the WEEKDAY function is the date of the day. DAX handles the date values in datetime format.

You can specify the date as one of the following –

- Output of another date function.
- An expression that returns a date value.
- A date in a datetime format.
- A date as text representation in one of the accepted string formats for dates.

DAX WEEKDAY function uses the locale and date/time settings of the client computer to understand the text value in order to perform the conversion. For example,

- If the current date/time settings represent dates in the format of Month/Day/Year, then the string "1/8/2016" is understood as a datetime value equivalent to 8th January, 2016.

- If the current date/time settings represent dates in the format of Day/Month/Year, the same string would be understood as a datetime value equivalent to 1st August, 2016.

Example

=WEEKDAY ("1-5-2016") returns 3.

=WEEKDAY (TODAY (),2) returns 5. (If today is Friday).

111. DAX Functions – WEEKNUM

Description

Returns the week number for the given date and year according to the return_type value. The week number indicates where the week falls numerically within a year.

Syntax

WEEKNUM (<date>, [<return_type>])

Parameters

Parameter	Description
date	Date in datetime format.
return_type	A number that determines the return value: 1 - Week begins on Sunday. Weekdays are numbered 1 through 7. 2 - Week begins on Monday. Weekdays are numbered 1 through 7. If omitted, the default value is 1.

Return Value

An integer, in the range 1 to 53.

Remarks

DAX uses datetime data type to work with dates and times.

If the source data is in a different format, DAX implicitly converts the data to datetime to perform calculations.

By default, the WEEKNUM function uses a calendar convention in which the week containing January 1 is considered to be the first week of the year.

Note: The ISO 8601 calendar standard, widely used in Europe, defines the first week as the one with the majority of days (four or more) falling in the New Year.

This means that for years in which there are three days or less in the first week of January, the WEEKNUM function returns week numbers that are different from the ISO 8601 definition.

Example

=WEEKNUM ("Oct 2, 2016", 1) returns 41.

=WEEKNUM ("Dec 31, 2016", 1) returns 53.

112. DAX Functions – YEAR

Description

Returns the year of a date as a four-digit integer in the range 1900-9999.

Syntax

YEAR (<date>)

Parameters

Parameter	Description
date	A date in datetime or text format, containing the year you want to find.

Return Value

An integer in the range 1900-9999.

Remarks

DAX uses datetime data type to work with dates and times.

YEAR function takes the parameter date in one of the following ways -

- By using the DATE function.
- As a result of other DAX formulas or DAX functions.
- As an accepted text representation of date.

The function uses the locale and date time settings of the client computer to understand the text value in order to perform the conversion. For example,

- If the current date/time settings represent dates in the format of Month/Day/Year, then the string "1/8/2016" is understood as a datetime value equivalent to 8th January, 2016.

- If the current date/time settings represent dates in the format of Day/Month/Year, the same string would be understood as a datetime value equivalent to 1st August, 2016.

If the format of the string is incompatible with the current locale settings, YEAR function might return an error. For example, if your locale defines dates to be formatted as month/day/year, and the date is provided as day/month/year, then 25/1/2009 will not be interpreted as January 25th of 2009 but as an invalid date.

Example

=YEAR (DATE (2016,9,15)) returns 2016.

=YEAR (TODAY ()) returns 2016 if TODAY () returns 12/16/2016 12:00:00 AM.

113. DAX Functions – YEARFRAC

Description

Calculates the fraction of the year represented by the number of whole days between two dates.

Syntax

YEARFRAC (<start_date>, <end_date>, [<basis>])

Parameters

Parameter	Description
start_date	The start date in datetime format.
end_date	The end date in datetime format.
basis	Optional. The type of day count basis to use. An integer between 0 and 4. If not an integer, the parameter will be truncated. 0 - US (NASD) 30/360. 1 - Actual/actual. 2 - Actual/360. 3 - Actual/365. 4 - European 30/360. If omitted, default is 0.

Return Value

A decimal number. The internal data type is a signed IEEE 64-bit (8-byte) double-precision floating-point number.

Remarks

You can use the YEARFRAC function to identify the proportion of a whole year's benefits or obligations to assign to a specific term.

DAX uses a datetime format to work with dates and times.

- If start_date or end_date are not valid dates, YEARFRAC returns an error.
- If basis < 0 or if basis > 4, YEARFRAC returns an error.

Example

=YEARFRAC ([InventoryDate], [UsageDate])

This formula returns a calculated column with fraction values representing InventoryDuration.

DAX Information Functions

114. DAX Information Functions – Overview

DAX Information functions look at the value or column that is provided as an argument and tell you whether the value matches the expected type.

Following are the DAX Information functions:

- DAX CONTAINS function
- DAX CUSTOMDATA function
- DAX ISBLANK function
- DAX ISERROR function
- DAX ISEMPTY function
- DAX ISEVEN function
- DAX ISLOGICAL function
- DAX ISNONTEXT function
- DAX ISNUMBER function
- DAX ISODD function
- DAX ISONORAFTER function
- DAX ISTEXT function
- DAX LOOKUPVALUE function
- DAX USERNAME function

115. DAX Functions – CONTAINS

Description

Returns true if values for all referred columns exist, or are contained, in those columns. Otherwise, returns false.

Syntax

CONTAINS (<table>, <columnName>, <value>, [<columnName>, <value>] …)

Parameters

Parameter	Description
table	Any DAX expression that returns a table of data.
columnName	The name of a column, in a table. It cannot be an expression.
value	Any DAX expression that returns a single scalar value. The expression is to be evaluated exactly once and before it is passed as a parameter. Value is what you find, if it exists or if contained in columnName.

Return Value

TRUE or FALSE.

Remarks

- The parameters columnName and value must come in pairs. Otherwise, the function returns an error.
- columnName must belong to the specified table, or to a table that is related to table.
- If columnName refers to a column in a related table, then it must be fully qualified. Otherwise, the function returns an error.

Example

```
=CONTAINS (Results, [Country],"IND", [Medal],"Gold")
```

This DAX formula returns TRUE, if there exists a Gold Medal for the Country India in the Results table. Otherwise, returns FALSE.

116. DAX Functions – CustomData

Description

Returns the content of the CustomData property in the connection string.

Syntax

CUSTOMDATA ()

Parameters

No parameters for this function.

Return Value

The content of the CustomData property in the connection string.

Blank, if CustomData property was not defined at the connection time.

Example

=CUSTOMDATA ()

117. DAX Functions – ISBLANK

Description

Checks whether a value is blank, and returns TRUE or FALSE.

Syntax

ISBLANK (<value>)

Parameters

Parameter	Description
value	The value or expression you want to test.

Return Value

TRUE or FLASE.

Example

```
=if (ISBLANK([Athlete]), "Name Missing", [Athlete])
```

This DAX formula returns a column of values – with Athlete name, if present and with the text string Name Missing, otherwise.

118. DAX Functions – ISERROR

Description

Checks whether a value is an error, and returns TRUE or FALSE.

Syntax

ISERROR (<value>)

Parameters

Parameter	Description
value	The value or expression you want to test.

Return Value

TRUE or FALSE.

Example

=ISERROR (5/1): This DAX formula returns FALSE.

=ISERROR (5/0): This DAX formula returns TRUE.

119. DAX Functions – ISEMPTY

Description

Checks if a table is empty.

DAX ISEMPTY function is new in Excel 2016.

Syntax

ISEMPTY (<table_expression>)

Parameters

Parameter	Description
table_expression	A table reference or a DAX expression that returns a table.

Return Value

TRUE or FALSE.

Example

= ISEMPTY (Events)

This DAX formula returns TRUE, if there is at least one row and one column of data in the table - Events. Otherwise, it returns FALSE.

120. DAX Functions – ISEVEN

Description

Returns TRUE if the number is even, or FALSE if the number is odd.

DAX ISEVEN function is new in Excel 2016.

Syntax

ISEVEN (<number>)

Parameters

Parameter	Description
number	The numeric value to test. If the number is not an integer, it is truncated.

Return Value

TRUE or FALSE.

Remarks

If the number is non-numeric, DAX ISEVEN function returns the #VALUE! error.

Example

=ISEVEN (8) returns TRUE.

=ISEVEN (7) returns FALSE.

121. DAX Functions – ISLOGICAL

Description

Checks whether a value is a logical value (TRUE or FALSE), and returns TRUE or FALSE.

Syntax

ISLOGICAL (<value>)

Parameters

Parameter	Description
value	The value that you want to test.

Return Value

TRUE or FALSE.

Example

=ISLOGICAL (5>4) returns TRUE.

122. DAX Functions – ISNONTEXT

Description

Checks if a value is not text, and returns TRUE or FALSE.

Syntax

ISNONTEXT (<value>)

Parameters

Parameter	Description
value	The value you want to check.

Return Value

TRUE or FALSE.

Remarks

- An empty string is considered as text.
- A blank cell is considered as non-text.

Example

=ISNONTEXT([Athlete])

This DAX formula returns a column of values with TRUE, if Athlete row value is Blank and FALSE if Athlete name exists.

123. DAX Functions – ISNUMBER

Description

Checks whether a value is a number, and returns TRUE or FALSE.

Syntax

ISNUMBER (<value>)

Parameters

Parameter	Description
value	The value you want to test.

Return Value

TRUE or FALSE.

Example

=ISNUMBER (5.0): The DAX formula returns TRUE.

=ISNUMBER("AB"): This DAX formula returns FALSE.

124. DAX Functions – ISODD

Description

Returns TRUE if the given number is odd. Otherwise, FALSE.

DAX ISODD function is new in Excel 2016.

Syntax

ISODD (<number>)

Parameters

Parameter	Description
number	The numeric value to test. If the number is not an integer, it is truncated.

Return Value

TRUE or FALSE.

Remarks

If the number is non-numeric, DAX ISODD function returns the #VALUE! error.

Example

=ISODD (5) returns TRUE.

=ISODD (2) returns FALSE.

125. DAX Functions – ISONORAFTER

Description

Takes a variable number of triples, the first two values in a triple are the expressions to be compared, and the third parameter indicates the sort order. The sort order can be ascending (default) or descending.

The first parameter in a triple is compared with the second parameter, based on the sort order. If the sort order is ascending, the comparison to be done is, the first parameter greater than or equal to the second parameter. If the sort order is descending, the comparison to be done is, the second parameter less than or equal to the first parameter.

ISONORAFTER function returns TRUE if any of the comparisons is TRUE, otherwise returns FALSE.

DAX ISONORAFTER function is new in Excel 2016.

Syntax

ISONORAFTER (<scalar_expression>, <scalar_expression>, [sort_order],

[<scalar_expression>, <scalar_expression>, [sort_order]] …)

Parameters

Parameter	Description
scalar_expression	Any expression that returns a scalar value like a column reference or integer or string value. Typically, the first parameter is a column reference and the second parameter is a scalar value.
sort_order	Optional. The order in which the comparison is done. ASC: Ascending, or DEC: Descending. If omitted, default is ascending.

Return Value

TRUE or FALSE.

Example

=ISONORAFTER(2,0,DESC,2,1,DESC): This DAX formula returns FALSE, because both 0 and 1 are not greater than 2.

=ISONORAFTER(2,5,DESC,2,1,DESC): This DAX formula returns TRUE, because 5 is greater than 2.

126. DAX Functions – ISTEXT

Description

Checks if a value is text, and returns TRUE or FALSE.

Syntax

ISTEXT (<value>)

Parameters

Parameter	Description
value	The value you want to check.

Return Value

TRUE or FALSE.

Remarks

Empty strings are considered as text.

Blank cells, BLANK () are considered as non-text.

Example

=ISTEXT("") returns TRUE.

=ISTEXT("AB") returns TRUE.

=ISTEXT(4) returns FALSE.

=ISTEXT(TRUE()) returns FALSE.

127. DAX Functions – LOOKUPVALUE

Description

Returns the value in result_columnName for the row that meets all criteria specified by search_columnName and search_value.

Syntax

LOOKUPVALUE (<result_columnName>, <search_columnName>, <search_value>, [<search_columnName>, <search_value>] …)

Parameters

Parameter	Description
result_columnName	The fully qualified name of a column that contains the value you want to return. It cannot be an expression.
search_columnName	The fully qualified name of a column, in the same table as result_columnName, or in a related table, over which the look-up is performed. It cannot be an expression.
search_value	A scalar expression that does not refer to any column in the same table being searched.

Return Value

- The value of result_column at the row where all pairs of search_column and search_value have a match.
- If there is no match that satisfies all the search values, a BLANK is returned. In other words, the function will not return a lookup value if only some of the criteria match.
- If multiple rows match the search values and in all cases result_column values are identical then that value is returned. However, if result_column returns different values an error is returned.

Example

```
=LOOKUPVALUE([Sport], [EventID],"E962")
```

This DAX formula returns the Sport corresponding to the EventID – E962.

128. DAX Functions – USERNAME

Description

Returns the domain name and the username from the credentials given to the system at the connection time.

Syntax

USERNAME ()

Parameters

No parameters to this function.

Return Value

The username from the credentials given to the system at connection time.

Example

=USERNAME ()

DAX Logical Functions

129. DAX Logical Functions – Overview

DAX Logical functions return logical values (TRUE/FALSE) based on the logical operations performed on the relevant parameters.

Following are the DAX Logical functions:

- DAX AND function
- DAX FALSE function
- DAX IF function
- DAX IFERROR function
- DAX NOT function
- DAX OR function
- DAX SWITCH function
- DAX TRUE function

130. DAX Functions – AND

Description

Checks the two arguments if they are TRUE or FALSE, and returns TRUE only when both are TRUE. Otherwise, returns FALSE.

Syntax

AND (<logical1>, <logical2>)

Parameters

Parameter	Description
logical1	Logical values that you want to test.
logical2	

Return Value

- TRUE, if both the arguments are TRUE.
- FALSE, otherwise.

Remarks

AND function takes only two arguments. If you have more than two arguments, either nest the AND functions or use the DAX logical operator &&.

Example

```
= AND([Country]="USA",[Medal]="Gold")
```

This DAX formula returns a calculated column with TRUE for Country – USA and Medal – Gold values, and FALSE otherwise.

USA Gold Medal

```
Count:=SUMX(Results,IF(AND([Country]="USA",[Medal]="Gold")=TRUE(),1,0))
```

This DAX formula returns a calculated field with total number of Gold Medals for USA.

131. DAX Functions – FALSE

Description

Returns the logical value FALSE.

Syntax

FALSE ()

Parameters

No parameters for this function.

Return Value

FALSE always.

Remarks

Word FALSE is also considered as the logical value FALSE.

Example

=SUMX (Results, IF ([Country]="USA", FALSE (),1))

Returns the number of Medals won by the Country other than USA.

132. DAX Functions – IF Function

Description

Checks a condition given as the first argument of the function and returns one value if the condition is TRUE and returns another value if the condition is FALSE.

Syntax

IF (<logical_test>, <value_if_true>, [<value_if_false>])

Parameters

Parameter	Description
logical_test	Any value or expression that can be evaluated to TRUE or FALSE.
value_if_true	The value that is returned if the logical test is TRUE.
value_if_false	Optional. The value that is returned if the logical test is FALSE. If omitted, FALSE is returned.

Return Value

Any type of value that can be returned by an expression.

Remarks

- If value_if_false is omitted, IF treats it as an empty string value ("").
- If the value referenced in the logical_test is a column, IF returns the value that corresponds to the current row. Thus, the IF function returns a column of all the values resulting from the logical test corresponding to each of the rows.
- If you have 3 values to return, then you can nest the IF functions.

Example

```
=IF([Country]="USA",1,0)
```

Returns a calculated column of 1's and 0's. These values can be summed up.

If you name the column as USA Medals, then you can write the following -

```
USA Medal Count:=SUM([USA Medals])
```

133. DAX Functions – IFERROR

Description

Evaluates an expression and returns a specified value if the expression returns an error. Otherwise, returns the value of the expression itself.

Syntax

IFERROR (<value>, <value_if_error>)

Parameters

Parameter	Description
value	Any value or expression.
value_if_error	Any value or expression.

Return Value

- The value returned by the expression, if error is not returned.
- The alternative value provided, if error is returned.

Remarks

You can use the function IFERROR to trap the errors returned by expressions and have meaningful values returned without abruptly stopping the evaluation.

- Both value and value_if_error must be of the same data type. Hence, the column or expression used for value and the value returned for value_if_error must be of the same data type.

- If either value or value_if_error is an empty cell, IFERROR treats it as an empty string value ("").

Example

=IFERROR (5/0,"Div by zero") returns Div by zero.

=IFERROR (5/1,"Div by zero") returns 5.

134. DAX Functions – NOT Function

Description

Converts FALSE to TRUE and TRUE to FALSE.

Syntax

NOT (<logical>)

Parameters

Parameter	Description
logical	A value or expression that can be evaluated to TRUE or FALSE.

Return Value

Either TRUE or FALSE.

Example

=NOT (AND ([Medal Count]<500, [Count of Sport]>100)) returns True.

=NOT (AND (Results[Medal Count]>150,Results[Count of Sport]>150)) returns False.

135. DAX Functions – OR Function

Description

Checks whether one of the arguments is TRUE. Returns TRUE if at least one of the arguments is TRUE and returns FALSE if both the arguments are FALSE.

Syntax

OR (<logical1>, <logical2>)

Parameters

Parameter	Description
logical1	The logical values you want to test.
logical2	

Return Value

- TRUE if any of the two arguments is TRUE.
- FALSE if both the arguments are FALSE.

Remarks

DAX OR function takes only two arguments. If you have more than two arguments, either nest the OR functions or use the logical OR operator ||.

Example

=OR (Results[Medal Count]>150,Results[Count of Sport]>150)

=OR ([Medal Count]<500, [Count of Sport]>100)

Both the above DAX formulas return True as [Count of Sport] > 100 is True.

OR ([Medal Count]<500, [Count of Sport]<100)

Returns False as both arguments are False.

136. DAX Functions – SWITCH

Description

Evaluates an expression against a list of values and returns one of multiple possible result expressions.

Syntax

SWITCH (<expression>, <value>, <result>, [<value>, <result>] …, [<else>])

Parameters

Parameter	Description
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times for each row/context.
value	A constant value to be matched with the results of expression.
result	Any scalar expression to be evaluated, if the results of expression match the corresponding value.
else	Optional. Any scalar expression to be evaluated, if the result of expression doesn't match any of the value arguments.

Return Value

A scalar value coming from one of the result expressions, if there was a match with value, or from the else expression, if there was no match with any value.

Remarks

All the result expressions and the else expression must be of the same data type.

Example

```
=SWITCH ([Week Day], 1, "Sunday", 2, "Monday", 3, "Tuesday", 4, "Wednesday", 5, "Thursday", 6, "Friday", 7, "Saturday", "Unknown")
```

This DAX formula returns a calculated column with the names of the Week Day values.

137. DAX Functions – TRUE Function

Description

Returns the logical value TRUE.

Syntax

TRUE ()

Parameters

No parameters for this function.

Return Value

TRUE always.

Remarks

The word TRUE is also interpreted as the logical value TRUE.

Example

```
=IF(AND([Country]="USA",[Medal]="Gold")=TRUE(),1,0))
```

Returns a calculated column with 1s for the rows with values USA in the column Country and Gold in the column Medal. Returns 0s for the rest.

DAX Mathematical & Trigonometric Functions

138. DAX Math & Trig Functions – Overview

DAX Mathematical and Trigonometric functions are very similar to the Excel mathematical and trigonometric functions.

Following are the DAX Math and Trig functions:

- DAX ABS function
- DAX ACOS function
- DAX ACOSH function
- DAX ASIN function
- DAX ASINH function
- DAX ATAN function
- DAX ATANH function
- DAX CEILING function
- DAX COMBIN function
- DAX COMBINA function
- DAX COS function
- DAX COSH function
- DAX CURRENCY function
- DAX DEGREES function
- DAX DIVIDE function
- DAX EVEN function
- DAX EXP function
- DAX FACT function
- DAX FLOOR function
- DAX GCD function
- DAX INT function
- DAX ISO.CEILING function
- DAX LCM function
- DAX LN function
- DAX LOG function
- DAX LOG10 function
- DAX MROUND function
- DAX MOD function
- DAX ODD function
- DAX PERMUT function
- DAX PI function

- DAX POWER function
- DAX QUOTIENT function
- DAX RADIANS function
- DAX RAND function
- DAX RANDBETWEEN function
- DAX ROUND function
- DAX ROUNDDOWN function
- DAX ROUNDUP function
- DAX SIGN function
- DAX SIN function
- DAX SINH function
- DAX SQRT function
- DAX SQRTPI function
- DAX TAN function
- DAX TANH function
- DAX TRUNC function

139. DAX Functions – ABS Function

Description

Returns the absolute value of a number.

Syntax

ABS (<number>)

Parameters

Parameter	Description
number	The number for which you want the absolute value.

Return Value

A decimal number.

Remarks

The absolute value of a number is a decimal number, whole or decimal, without its sign.

You can use DAX ABS function as a nested function to a DAX function that requires a positive number as a parameter. The numbers that are returned from the expressions used as parameters can be returned as positive numbers with the nested ABS function.

Example

=ABS (-2.5) returns 2.5.

140. DAX Functions – ACOS Function

Description

Returns the arccosine, or inverse cosine, of a number.

DAX ACOS function is new in Excel 2016.

Syntax

ACOS (<number>)

Parameters

Parameter	Description
number	The cosine of the angle you want. It must be from -1 to 1.

Return Value

An angle given in radians in the range 0 (zero) to pi.

Remarks

The arccosine is the angle whose cosine is number.

If you want to convert the result from radians to degrees, multiply it by 180/PI () or use the DAX DEGREES function.

Example

=DEGREES(ACOS(-0.6)) returns 126.869897645844.

=DEGREES(ACOS(0)) returns 90.

141. DAX Functions – ACOSH Function

Description

Returns the inverse hyperbolic cosine of a number. The number must be greater than or equal to 1.

DAX ACOSH function is new in Excel 2016.

Syntax

ACOSH (<number>)

Parameters

Parameter	Description
number	Any real number equal to or greater than 1.

Return Value

Returns the inverse hyperbolic cosine of a number.

Remarks

The inverse hyperbolic cosine is the value whose hyperbolic cosine is number.

ACOSH (COSH (number)) equals number.

Example

=ACOSH(1) returns 0.

=ACOSH(6) returns 2.47788873028848.

142. DAX Functions – ASIN Function

Description

Returns the arcsine, or inverse sine, of a number.

DAX ASIN function is new in Excel 2016.

Syntax

ASIN (<number>)

Parameters

Parameter	Description
number	The sine of the angle you want and must be from -1 to 1.

Return Value

An angle given in radians in the range - pi/2 to pi/2.

Remarks

The arcsine is the angle whose sine is number.

To express the returned angle in degrees, multiply the result by 180/PI () or use the DAX DEGREES function.

Example

=DEGREES(ASIN(0)) returns 0.

=DEGREES(ASIN(1)) returns 90.

=DEGREES(ASIN(0.7)) returns 44.4270040008057.

143. DAX Functions – ASINH Function

Description

Returns the inverse hyperbolic sine of a number.

DAX ASINH function is new in Excel 2016.

Syntax

ASINH (<number>)

Parameters

Parameter	Description
number	Any real number.

Return Value

Returns the inverse hyperbolic sine of a number.

Remarks

The inverse hyperbolic sine is the value whose hyperbolic sine is number.

ASINH (SINH (number)) equals number.

Example

=ASINH(55) returns 4.7005630001772.

=ASINH(1.5) returns 1.19476321728711.

144. DAX Functions – ATAN Function

Description

Returns the arctangent, or inverse tangent, of a number.

DAX ATAN function is new in Excel 2016.

Syntax

ATAN (<number>)

Parameters

Parameter	Description
number	The tangent of the angle you want.

Return Value

An angle given in radians in the range - pi/2 to pi/2.

You can get the arctangent in degrees, by multiplying the result by 180/PI () or by using DAX DEGREES function.

Remarks

The arctangent is the angle whose tangent is a number.

Example

=DEGREES(ATAN(80)) returns 89.2838400545296.

145. DAX Functions – ATANH Function

Description

Returns the inverse hyperbolic tangent of a number.

DAX ATANH function is new in Excel 2016.

Syntax

ATANH (<number>)

Parameters

Parameter	Description
number	Any real number between 1 and -1 (excluding -1 and 1).

Return Value

Returns the inverse hyperbolic tangent of a number.

Remarks

The inverse hyperbolic tangent is the value whose hyperbolic tangent is a number.

ATANH (TANH (number)) equals number.

Example

=ATANH(.1) returns 0.100335347731076.

146. DAX Functions – CEILING Function

Description

Rounds a number up, to the nearest integer or to the nearest multiple of significance.

Syntax

CEILING (<number>, <significance>)

Parameters

Parameter	Description
number	The number you want to round, or a reference to a column that contains numbers.
significance	The multiple of significance to which you want to round.

Return Value

Number rounded as specified.

The return type is usually of the same type of the significance parameter, with the following exceptions:

- If the number parameter type is currency, the return type is currency.
- If the significance parameter type is Boolean, the return type is integer.
- If the significance parameter type is non-numeric, the return type is real.

Remarks

DAX has two CEILING functions –

- **CEILING**: Same as the Excel CEILING function.
- **ISO.CEILING**: Follows the ISO-defined behavior for determining the ceiling value.

Both DAX CEILING and ISO.CEILING functions return the same value for positive numbers. However, they return different values for negative numbers.

- When using a positive multiple of significance, both CEILING and ISO.CEILING round negative numbers upward (toward positive infinity).

- When using a negative multiple of significance, CEILING rounds negative numbers downward (toward negative infinity), while ISO.CEILING rounds negative numbers upward (toward positive infinity).

Example

=CEILING(4.6,1) returns 5.

=CEILING(4.6,2) returns 6.

=CEILING(-4.6,-1) returns -5.

=CEILING(-4.6,-2)) returns -6.

147. DAX Functions – COMBIN Function

Description

Returns the number of combinations for a given number of items.

DAX COMBIN function is new in Excel 2016.

Syntax

COMBIN (<number>, <number_chosen>)

Parameters

Parameter	Description
number	The number of items.
number_chosen	The number of items in each combination.

Return Value

A whole number.

Remarks

A combination is any set or subset of items, regardless of their internal order.

If number = **n** and number_chosen = **k**, then the number of combinations is given as –

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)}$$

You can use DAX COMBIN function to determine the total possible number of groups for a given number of items.

- Numeric parameters are rounded to integers.
- If either parameter is non-numeric, COMBIN returns #ERROR.
- If number < 0, or number_chosen < 0, or number < number_chosen, COMBIN returns #ERROR.

Example

=COMBIN (6,2) returns 15.

=COMBIN (5.6,2) returns 15.

=COMBIN (5.4,2) returns 10.

148. DAX Functions – COMBINA

Description

Returns the number of combinations (with repetitions) for a given number of items.

DAX COMBINA function is new in Excel 2016.

Syntax

COMBINA (<number>, <number_chosen>)

Parameters

Parameter	Description
number	The number of items. Must be greater than or equal to 0, and greater than or equal to number_chosen. Non-integer values are rounded.
number_chosen	The number of items in each combination. Must be greater than or equal to 0. Non-integer values are rounded.

Return Value

A whole number.

Remarks

- If the value of either parameter is outside of its constraints, COMBINA returns #ERROR.
- If either parameter is a non-numeric value, COMBINA returns #ERROR.

The following equation is used, where N is number and M is number_chosen:

$$\binom{N+M-1}{N-1}$$

Example

=COMBINA (5,2) returns 15.

=COMBINA (5.4,2) returns 15.

=COMBINA (5.5,2) returns 21.

149. DAX Functions – COS Function

Description

Returns the cosine of the given angle. DAX COS function is new in Excel 2016.

Syntax

COS (<number>)

Parameters

Parameter	Description
number	The angle in radians for which you want the cosine.

Return Value

The cosine of the given angle.

Remarks

If the parameter angle is in degrees, either multiply the angle by PI ()/180 or use DAX RADIANS function to convert the angle to radians.

Example

=COS (RADIANS (60)) returns 0.5.

=COS (RADIANS (30)) returns 0.866025403784439

150. DAX Functions – COSH Function

Description

Returns the hyperbolic cosine of a number. DAX COSH function is new in Excel 2016.

Syntax

COSH (<number>)

Parameters

Parameter	Description
number	Any real number for which you want to find the hyperbolic cosine.

Return Value

The hyperbolic cosine of number.

Remarks

The formula for the hyperbolic cosine is:

$$\text{COSH}(z) = \frac{e^{z} + e^{-z}}{2}$$

Example

=COSH (1.5) returns 2.35240961524325.

=COSH (2.0) returns 3.76219569108363.

151. DAX Functions – CURRENCY Function

Description

Evaluates the parameter and returns the result as currency data type.

Syntax

CURRENCY (<value>)

Parameters

Parameter	Description
value	Any DAX expression that returns a single scalar value where the expression is to be evaluated exactly once before all other operations.

Return Value

The value of the expression evaluated and returned as a currency type value.

Remarks

- DAX CURRENCY function rounds up the 5th significant decimal, in value, to return the 4th decimal digit. Rounding up occurs if the 5th significant decimal is equal or larger than 5.
- If the data type of the expression is Logical (TRUE/FALSE), then DAX CURRENCY function will return $1.0000 for TRUE values and $0.0000 for FALSE values.
- If the data type of the expression is Text, then DAX CURRENCY function will try to convert text to a number. If conversion succeeds, the number will be converted to currency. Otherwise, an error is returned.
- If the data type of the expression is datetime, then DAX CURRENCY function will convert the datetime value to a number and then that number to currency. If the value of the expression is not a proper datetime value, an error is returned.

Example

=CURRENCY (5.0) returns 5.

=CURRENCY (55.55555555555) returns 55.5556.

152. DAX Functions – DEGREES Function

Description

Converts an angle in radians to degrees.

DAX DEGREES function is new in Excel 2016.

Syntax

DEGREES (<angle>)

Parameters

Parameter	Description
angle	The angle in radians that you want to convert.

Return Value

Angle in degrees.

Example

=DEGREES (1.5) returns 85.9436692696235.

=DEGREES (0.525) returns 30.0802842443682.

153. DAX Functions – DIVIDE Function

Description

Performs division and returns alternate result or BLANK () on division by 0.

Syntax

DIVIDE (<numerator>, <denominator>, [<alternateresult>])

Parameters

Parameter	Description
numerator	The dividend or the number to divide.
denominator	The divisor or the number to divide by.
alternateresult	Optional. The value returned when division by zero results in an error. If omitted, the default value is BLANK ().

Return Value

A decimal number or BLANK ().

Remarks

alternateresult on 'divide by 0' must be a constant.

Example

=DIVIDE (5,0,"Div by zero Error") returns Div by zero Error.

=DIVIDE (5,0) returns (blank).

=DIVIDE (5,4) returns 1.25.

154. DAX Functions – EVEN Function

Description

Returns the number rounded up to the nearest even integer.

DAX EVEN function is new in Excel 2016.

Syntax

EVEN (<number>)

Parameters

Parameter	Description
number	The number value to round.

Return Value

Number rounded up to the nearest even integer.

Remarks

You can use this function for processing items that come in twos.

Example

=EVEN (3) returns 4.

=EVEN (2.25) returns 4.

=EVEN (2.55) returns 4.

=EVEN (-2.55) returns -4.

=EVEN (-2.25) returns -4.

155. DAX Functions – EXP Function

Description

Returns **e** raised to the power of a given number.

The constant **e** equals 2.71828182845904, the base of the natural logarithm.

Syntax

EXP (<number>)

Parameters

Parameter	Description
number	The exponent applied to the base **e**.

Return Value

A decimal number.

Remarks

DAX EXP function is the inverse of DAX LN function, which is the natural logarithm of the given number.

To calculate the powers of bases other than **e**, use the DAX exponentiation operator (^).

Example

=EXP (2.0) returns 7.38905609893065.

=EXP (2.5) returns 12.1824939607035.

156. DAX Functions – FACT Function

Description

Returns the factorial of a number, equal to the series 1*2*3*...*, ending in the given number.

Syntax

FACT (<number>)

Parameters

Parameter	Description
number	The non-negative integer for which you want to calculate the factorial.

Return Value

An integer.

Remarks

- If the number is not an integer, it is truncated.
- If the number is negative, an error is returned.
- If the result is too large, an error is returned.

Example

=FACT (8) returns 40320.

=FACT (8.6) returns 40320.

157. DAX Functions – FLOOR Function

Description

Rounds a number down, toward zero, to the nearest multiple of significance.

Syntax

FLOOR (<number>, <significance>)

Parameters

Parameter	Description
number	The numeric value you want to round.
significance	The multiple to which you want to round.

The parameters number and significance must either both be positive, or both negative.

Return Value

A decimal number.

Remarks

- If either parameter is non-numeric, FLOOR returns #VALUE! error.
- If the number is positive and significance is negative, FLOOR returns error.
- Regardless of the sign of the number, a value is rounded down when adjusted away from zero.
- If the number is an exact multiple of significance, no rounding occurs.

Example

=FLOOR (2.8,2) returns 2.

=FLOOR (2.8,3) returns 0.

=FLOOR (-2.8,2) returns -4.

=FLOOR (-2.8, -2) returns -2.

158. DAX Functions – GCD Function

Description

Returns the greatest common divisor of two integers.

DAX GCD function is new in Excel 2016.

Syntax

GCD (<number1>, <number2>)

Parameters

Term	Definition
number1	A positive number.
number2	A positive number.

Return Value

A positive integer.

Remarks

The greatest common divisor is the largest integer that divides both number1 and number2 without a remainder.

- If any of the parameters is not an integer, it will be rounded.
- If any of the parameters is negative, the function returns error.
- If any of the parameters is non-numeric, the function returns error.
- If the parameters do not have a common divisor > 1, the function returns 1.

Example

=GCD (16,20) returns 4.

=GCD (15,25) returns 5.

159. DAX Functions – INT Function

Description

Rounds a number down to the nearest integer.

Syntax

INT (<number>)

Parameters

Parameter	Description
Number	The number you want to round down to an integer.

Return Value

A whole number.

Remarks

DAX INT and TRUNC functions are similar - both return integers.

- INT rounds the numbers down to the nearest integer based on the value of the fractional part of the number.
- TRUNC removes the fractional part of the number.

Thus, both INT and TRUNC return same results for positive numbers. For example,

- INT (5.3) returns 5 because 5 is the lower number.
- TRUNC (5.3) returns 5 as it removes the fractional part.

However, INT and TRUNC return different results for negative numbers. For example,

- INT (-4.3) returns -5 because -5 is the lower number.
- TRUNC (-4.3) returns -4 as it removes the functional part.

Example

=INT (5.3) returns 5.

=INT (-5.3) returns -6.

160. DAX Functions – ISO.CEILING Function

Description

Rounds a number up, to the nearest integer or to the nearest multiple of significance.

Syntax

ISO.CEILING (<number>, [<significance>])

Parameters

Parameter	Description
number	The number you want to round, or a reference to a column that contains numbers.
significance	Optional. The multiple of significance to which you want to round. If omitted, the number is rounded up to the nearest integer.

Return Value

A number, of the same type as the number argument, rounded as specified.

Remarks

DAX has two CEILING functions –

- **CEILING**: Same as the Excel CEILING function.
- **ISO.CEILING**: Follows the ISO-defined behavior for determining the ceiling value.

Both DAX CEILING and ISO.CEILING functions return the same value for positive numbers.

However, they return different values for negative numbers.

- When using a positive multiple of significance, both CEILING and ISO.CEILING round negative numbers upward (toward positive infinity).

- When using a negative multiple of significance, CEILING rounds negative numbers downward (toward negative infinity), while ISO.CEILING rounds negative numbers upward (toward positive infinity).

Example

=ISO.CEILING(2.5) returns 3.

=ISO.CEILING(2.5,4) returns 4.

=ISO.CEILING(-2.5,4) returns 0.

161. DAX Functions – LCM Function

Description

Returns the least common multiple of integers.

DAX LCM function is new in Excel 2016.

Syntax

LCM (<number1>, [<number2>] ...)

Parameters

Parameter	Description
number1, number2, …	number1 is required, subsequent numbers are optional - 1 to 255 values for which you want the least common multiple. If the value is not an integer, it is truncated.

Return Value

An Integer.

Remarks

The least common multiple is the smallest positive integer that is a multiple of all integer arguments number1, number2, etc.

You can use LCM to add fractions with different denominators.

- If any parameter is non-numeric, LCM returns error.
- If any parameter is less than zero, LCM returns error.
- If LCM (a, b) >= 2^53, LCM returns error.

Example

=LCM (2,4) returns 4.

=LCM (5,3) returns 15.

162. DAX Functions – LN Function

Description

Returns the natural logarithm of a number. Natural logarithms are based on the constant **e** (2.71828182845904).

Syntax

LN (<number>)

Parameters

Parameter	Description
number	The positive number for which you want the natural logarithm.

Return Value

A decimal number.

Remarks

DAX LN function is the inverse of the DAX EXP function.

Example

=LN (5) returns 1.6094379124341.

=LN (15) returns 2.70805020110221.

163. DAX Functions – LOG Function

Description

Returns the logarithm of a number to the base you specify.

Syntax

LOG (<number>, [<base>])

Parameters

Parameter	Description
number	The positive number for which you want the logarithm.
base	Optional. The base of the logarithm. If omitted, default is 10.

Return Value

A decimal number.

Remarks

You might receive an error, if the return value is too large to be displayed.

The DAX function LOG10 is similar, but always returns the common logarithm, i.e. the logarithm for the base 10.

Example

=LOG (5,10) returns 0.698970004336019.

=LOG (5) returns 0.698970004336019.

164. DAX Functions – LOG10 Function

Description

Returns the base 10 logarithm of a number.

Syntax

LOG10 (<number>)

Parameters

Parameter	Description
number	A positive number for which you want the base 10 logarithm.

Return Value

A decimal number.

Remarks

You can use DAX LOG function, if you want to use a base other than 10 of the logarithm.

Example

=LOG10 (5) returns 0.698970004336019.

This is equivalent to LOG (5) or LOG (5,10).

165. DAX Functions – MROUND Function

Description

Returns a number rounded to the desired multiple.

Syntax

MROUND (<number>, <multiple>)

Parameters

Parameter	Description
number	The number to round.
multiple	The multiple of significance to which you want to round the number.

Return Value

A decimal number.

Remarks

MROUND rounds up, away from zero, if the remainder of the dividing number by the specified multiple is greater than or equal to half the value of multiple.

If the number and multiple are not of the same sign, the function returns an error.

Example

=MROUND (8.2,3) returns 9.

=MROUND (-8.2, -3) returns -9.

166. DAX Functions – MOD Function

Description

Returns the remainder after a number is divided by a divisor. The result always has the same sign as the divisor.

Syntax

MOD (<number>, <divisor>)

Parameters

Parameter	Description
number	The number for which you want to find the remainder after the division is performed.
divisor	The number by which you want to divide.

Return Value

A whole number.

Remarks

If the divisor is 0 (zero), MOD returns an error.

Example

=MOD (5,2) returns 1.

=MOD (5, -2) returns -1.

167. DAX Functions – ODD Function

Description

Returns the number rounded up to the nearest odd integer.

DAX ODD function is new in Excel 2016.

Syntax

ODD (<number>)

Parameters

Parameter	Description
number	The number to round.

Return Value

An odd integer.

Remarks

- If the number is non-numeric, ODD returns the #VALUE! error.
- Regardless of the sign of the number, a value is rounded up when adjusted away from zero.
- If the number is an odd integer, no rounding occurs.

Example

=ODD (2.3) returns 3.

=ODD (-2.3) returns -3.

168. DAX Functions – PERMUT Function

Description

Returns the number of permutations for a given number of objects that can be selected from the number of objects. A permutation is any set or subset of objects or events where the internal order is significant.

Permutations are different from combinations, for which the internal order is not significant.

DAX PERMUT function is new in Excel 2016.

Syntax

PERMUT (<number>, <number_chosen>)

Parameters

Term	Definition
number	A number that specifies the number of objects.
number_chosen	A number that specifies the number of objects in each permutation.

Return Value

A whole number.

Remarks

You can use this function for lottery-style probability calculations.

- Both parameters are truncated to integers.
- If the number or number_chosen is non-numeric, PERMUT returns the #VALUE! error value.
- If the number ≤ 0 or if number_chosen < 0, PERMUT returns the #NUM! error value.

- If the number < number_chosen, PERMUT returns the #NUM! error value.

- The equation for the number of permutations is:

$$P_{k,n} = \frac{n!}{(n-k)!}$$

Example

=PERMUT (8,3) returns 336.

=PERMUT (9,2) returns 72.

169. DAX Functions – PI Function

Description

Returns the value of Pi, 3.14159265358979, accurate to 15 digits.

Syntax

PI ()

Parameters

This function has no parameters.

Return Value

3.14159265358979, accurate to 15 digits.

Remarks

Pi is a mathematical constant.

In DAX, Pi is represented as a real number accurate to 15 digits.

Example

=PI ().

170. DAX Functions – POWER Function

Description

Returns the result of a number raised to a power.

Syntax

POWER (<number>, <power>)

Parameters

Parameter	Description
number	The base number, which can be any real number.
power	The exponent to which the base number is raised.

Return Value

A decimal number.

Example

=POWER (5,3) returns 125.

=POWER (-5,3) returns -125.

=POWER (5, -3) returns 0.008.

=POWER (-5, -3) returns -0.008.

171. DAX Functions – QUOTIENT Function

Description

Performs division and returns only the integer portion of the division result.

Syntax

QUOTIENT (<numerator>, <denominator>)

Parameters

Parameter	Description
numerator	The dividend, or number to divide.
denominator	The divisor, or number to divide by.

Return Value

A whole number.

Remarks

You can use this function when you want to discard the remainder of division.

If either parameter is non-numeric, QUOTIENT returns the #VALUE! error.

You can use a column reference instead of a literal value for either parameter. However, if the column that you reference contains a 0 (zero), an error is returned for the entire column of values.

Example

=QUOTIENT (5,3) returns 1.

=QUOTIENT (-5,3) returns -1.

172. DAX Functions – RADIANS Function

Description

Converts degrees to radians. DAX RADIANS function is new in Excel 2016.

Syntax

RADIANS (<angle>)

Parameters

Parameter	Description
angle	An angle in degrees that you want to convert.

Return Value

Angle in radians.

Example

=RADIANS (30) returns 0.523598775598299.

=RADIANS (55) returns 0.959931088596881.

173. DAX Functions – RAND Function

Description

Returns a random number greater than or equal to 0 and less than 1, evenly distributed.

The number that is returned changes each time the cell containing this function is recalculated.

Syntax

RAND ()

Parameters

No parameters for this function.

Return Value

A decimal number between 0 and 1.

Remarks

DAX RAND function will be recalculated when the entire column is recalculated. This can happen in the following situations –

- When the data from an external data source is refreshed.
- When the DAX formulas that contain RAND function are edited, causing re-evaluation of formulas.

DAX RAND function will always be recalculated, if the function is used in the definition of a calculated field. Also, in such contexts the RAND function cannot return a result of zero, to prevent errors such as division by zero.

However, DAX RAND function will not be recalculated when the execution of a query or filtering happens.

Example

=RAND () * 100 returns 51.2418613788544.

174. DAX Functions – RANDBETWEEN

Description

Returns a random number in the range between two numbers you specify.

Syntax

RANDBETWEEN (<bottom>, <top>)

Parameters

Parameter	Description
bottom	The smallest integer the function will return.
top	The largest integer the function will return.

Return Value

A whole number.

Example

=RANDBETWEEN (0,100)

175. DAX Functions – ROUND

Description

Rounds a number to the specified number of digits.

Syntax

ROUND (<number>, <num_digits>)

Parameters

Parameter	Description
number	The number you want to round.
num_digits	The number of digits to which you want to round.

Return Value

A decimal number.

Remarks

- If num_digits > 0, then the number is rounded to the specified number of decimal places.
- If num_digits = 0, the number is rounded to the nearest integer.
- If num_digits < 0, the number is rounded to the left of the decimal point.

You can also use the following DAX functions in specific instances –

- DAX ROUNDUP function to always round up (away from zero).
- DAX ROUNDDOWN function to always round down (toward zero).
- DAX MROUND function to round a number to a specific multiple.
- DAX TRUNC and INT functions to obtain the integer portion of the number.

Example

=ROUND (5.5555555,2) returns 5.56.

=ROUND (5.5555555,0) returns 6.

=ROUND (5.5555555, -1) returns 10.

=ROUND (5.5555555, -2) returns 0.

176. DAX Functions – ROUNDDOWN

Description

Rounds a number down, toward zero.

Syntax

ROUNDDOWN (<number>, <num_digits>)

Parameters

Parameter	Description
number	A real number that you want rounded down.
num_digits	The number of digits to which you want to round.

Return Value

A decimal number.

Remarks

- **Positive**: Number is rounded down to the specified number of decimal places.
- **Zero**: Number is rounded to the nearest integer.
- **Negative**: Number is rounded to the left of the decimal point.

You can also use the following DAX functions –

- DAX ROUND function to round to the specified number of digits.
- DAX ROUNDUP function to always round up (away from zero).
- DAX MROUND function to round a number to a specific multiple.
- DAX TRUNC and INT functions to obtain the integer portion of the number.

Example

=ROUNDDOWN (5.55555,2) returns 5.55.

=ROUNDDOWN (5.55555,0) returns 5.

=ROUNDDOWN (5.55555, -1) returns 0.

=ROUNDDOWN (5.55555, -2) returns 0.

177. DAX Functions – ROUNDUP

Description

Rounds a number up, away from 0 (zero).

Syntax

ROUNDUP (<number>, [<num_digits>])

Parameters

Parameter	Description
number	A real number that you want to round up.
num_digits	The number of digits to which you want to round.

Return Value

A decimal number.

Remarks

- If number > 0, rounds up to the specified number of decimal places.
- If number = 0, rounds to the nearest integer.
- If number < 0, rounds to the left of the decimal point.

You can also use the following DAX functions in specific instances –

- DAX ROUND function to always round to the specified number of digits.
- DAX ROUNDDOWN function to always round down (toward zero).
- DAX MROUND function to round a number to a specific multiple.
- DAX TRUNC and INT functions to obtain the integer portion of the number.

Example

=ROUNDUP (5.55555,2) returns 5.56.

=ROUNDUP (5.55555,0) returns 6.

=ROUNDUP (5.55555, -1) returns 10.

=ROUNDUP (5.55555, -2) returns 100.

178. DAX Functions – SIGN Function

Description

Determines the sign of a number, or the result of a calculation, or a number in a column. The function returns 1 if the number is positive, 0 (zero) if the number is zero, or -1 if the number is negative.

Syntax

SIGN (<number>)

Parameters

Parameter	Description
number	Any real number, or a column that contains numbers, or an expression that evaluates to a number.

Return Value

One of the following:

- 1
- 0
- -1

Example

=SIGN (55) returns 1.

=SIGN (-55) returns -1.

=SIGN (0) returns 0.

179. DAX Functions – SIN Function

Description

Returns the sine of the given angle. DAX SIN function is new in Excel 2016.

Syntax

SIN (<number>)

Parameters

Term	Definition
number	The angle in radians for which you want the sine.

Return Value

The sine of the given angle.

Remarks

If the value for your parameter is in degrees, multiply it by PI ()/180 or use DAX RADIANS function to convert it to radians.

Example

=SIN (RADIANS (90)) returns 1.

=SIN (RADIANS (0)) returns 0.

180. DAX Functions – SINH Function

Description

Returns the hyperbolic sine of a number. DAX SINH function is new in Excel 2016.

Syntax

SINH (<number>)

Parameters

Term	Definition
number	Any real number.

Return Value

The hyperbolic sine of a number.

Remarks

The formula for the hyperbolic sine is:

$$SINH(z) = \frac{e^{z} - e^{-z}}{2}$$

Example

=SINH (5) returns 74.2032105777888.

=SINH (1) returns 1.1752011936438.

181. DAX Functions – SQRT Function

Description

Returns the square root of a number.

Syntax

SQRT (<number>)

Parameters

Parameter	Description
number	The number for which you want the square root, a column that contains numbers, or an expression that evaluates to a number.

Return Value

A decimal number.

Remarks

If the number is negative, the SQRT function returns an error.

Example

=SQRT (25) returns 5.

=SQRT (225) returns 15.

182. DAX Functions – SQRTPI

Description

Returns the square root of (number * pi). DAX SQRTPI function is new in Excel 2016.

Syntax

SQRTPI (<number>)

Parameters

Term	Definition
number	The number by which pi is multiplied.

Return Value

A real number.

Example

=SQRTPI (5) returns 3.96332729760601.

=SQRTPI (1) returns 1.77245385090552.

183. DAX Functions – TAN Function

Description

Returns the tangent of the given angle. DAX TAN function is new in Excel 2016.

Syntax

TAN (<number>)

Parameters

Term	Definition
number	The angle in radians for which you want the tangent.

Return Value

The tangent of the given angle.

Remarks

If your parameter value is in degrees, multiply it by PI ()/180 or use DAX RADIANS function to convert it to radians.

Example

=TAN (RADIANS (0)) returns 0.

=TAN (RADIANS (60)) returns 1.73205080756888.

184. DAX Functions – TANH Function

Description

Returns the hyperbolic tangent of a number. DAX TANH function is new in Excel 2016.

Syntax

TANH (<number>)

Parameters

Term	Definition
number	Any real number.

Return Value

The hyperbolic tangent of a number.

Remarks

The formula for the hyperbolic tangent is:

$$\text{TANH}(z) = \frac{\text{SINH}(z)}{\text{COSH}(z)}$$

Example

=TANH (5) returns 0.999909204262595.

=TANH (1) returns 0.761594155955765.

185. DAX Functions – TRUNC

Description

Truncates a number to an integer by removing the decimal part of the number.

Syntax

TRUNC (<number>)

Parameters

Parameter	Description
number	The number you want to truncate.

Return Value

A whole number.

Remarks

DAX TRUNC and INT functions result in the same integer for positive numbers.

- TRUNC removes the fractional part of the number.
- INT rounds the number down to the nearest integer.

However, DAX TRUNC and INT functions might result in different integers for negative numbers.

- TRUNC removes the fractional part of the number.
- INT rounds the number down to the nearest integer based on the value of the fractional part of the number.

For example, TRUNC (-6.3) returns -6, but INT (-6.3) returns -7 because -7 is the nearest integer.

Example

=TRUNC (5.8) returns 5.

=TRUNC (-5.8) returns -5.

DAX Parent & Child Functions

186. DAX Parent & Child Functions – Overview

DAX Parent and Child functions are useful in managing data that is presented as a parent/child hierarchy in the Data Model.

Following are the DAX Parent and Child functions:

- DAX PATH function
- DAX PATHCONTAINS function
- DAX PATHITEM function
- DAX PATHITEMREVERSE function
- DAX PATHLENGTH function

187. DAX Functions – PATH Function

Description

Returns a delimited text string with the identifiers of all the parents of the current identifier, starting with the oldest and continuing until the current identifier.

Syntax

PATH (<ID_columnName>, <parent_columnName>)

Parameters

Parameter	Description
ID_columnName	The name of an existing column containing the unique identifier for rows in the table. This cannot be an expression. The data type of the value in ID_columnName must be text or integer, and must be the same data type as the column referenced in parent_columnName.
parent_columnName	The name of an existing column containing the unique identifier for the parent of the current row. This cannot be an expression. The data type of the value in parent_columnName data type must be a text or an integer, and must be the same data type as the value in ID_columnName.

Return Value

A delimited text string containing the identifiers of all the parents to the current identifier.

Remarks

DAX PATH function is used in tables that have some kind of internal hierarchy, to return the items that are related to the current row value.

For example, suppose you have a table Employees that contains the details of employees in an organization. The table contains -

- Employee ID of employees.
- Employee ID of the managers of employees.
- Employee ID of the managers of the managers.

You can use DAX PATH function to return the path that connects an employee to his or her manager.

The path is not constrained to a single level of parent-child relationships. It can return related rows that are several levels up from the specified starting row, i.e., the path that connects an employee to his or her manager's manager.

- The delimiter used to separate the ascendants is the vertical bar, '|'.
- The values in ID_columnName and parent_columnName must have the same data type, text, or integer.
- Values in parent_columnName must be present in ID_columnName. That is, you cannot look up a parent, if there is no value at the child level.
- If parent_columnName is BLANK then PATH () returns ID_columnName value. In other words, if you look for the manager of an employee but the parent_columnName column has no data, the PATH function returns just the employee ID.
- If ID_columnName has duplicates and parent_columnName is the same for those duplicates, then PATH () returns the common parent_columnName value. However, if parent_columnNamevalue is different for those duplicates then PATH () returns an error. In other words, if you have two listings for the same employee ID and they have the same manager ID, the PATH function returns the ID for that manager. However, if there are two identical employee IDs that have different manager IDs, PATH function returns an error.
- If ID_columnName is BLANK, then PATH () returns BLANK.
- If ID_columnName contains a vertical bar '|' then PATH () returns an error.

Example

```
=PATH (Employee[EmployeeID],Employee[ManagerEmployeeID])
```

This DAX formula returns a calculated column containing the delimited strings of EmployeeIDs of all the managers in the hierarchy above each employee starting from the topmost employee.

For example, OrgEmp0001|OrgEmp0002|OrgEmp0006|OrgEmp0015 is the PATH returned for an employee with ID OrgEmp0015, where the reporting hierarchy is OrgEmp0015 -> OrgEmp0006 -> OrgEmp0002 -> OrgEmp0001.

188. DAX Functions – PATHCONTAINS

Description

Returns TRUE if the specified item exists within the specified path. Otherwise, returns FALSE.

Syntax

PATHCONTAINS (<path>, <item>)

Parameters

Parameter	Description
path	A string created as the result of evaluating a PATH function.
item	A text expression to look for in the path result.

Return Value

TRUE or FALSE.

Remarks

If the item is an integer number, it is converted to text and then the function is evaluated. If conversion fails, then the function returns an error.

Example

=PATHCONTAINS("OrgEmp0001|OrgEmp0002|OrgEmp0006|OrgEmp0014","OrgEmp0007") returns FALSE.

=PATHCONTAINS("OrgEmp0001|OrgEmp0002|OrgEmp0006|OrgEmp0014","OrgEmp0006") returns TRUE.

189. DAX Functions – PATHITEM

Description

Returns the item at the specified position from a string, resulting from an evaluation of a PATH function.

Positions are counted from the left to the right.

Syntax

PATHITEM (<path>, <position>, [<type>])

Parameters

Parameter	Description
path	A text string in the form of the results of a PATH function.
position	An integer expression with the position of the item to be returned.
type	Optional. An enumeration that defines the data type of the result. TEXT or 0: Results are returned with the data type text. (If omitted, this is default). INTEGER or 1: Results are returned as integers.

Return Value

The identifier returned by the PATH function at the specified position in the list of identifiers.

Items returned by the PATH function are ordered by the most distant to the current.

Remarks

- This function can be used to return a specific level from a hierarchy returned by a PATH function.
- If you specify a number for position that is less than one (1) or greater than the number of elements in path, DAX PATHITEM function returns BLANK.
- If type is not a valid enumeration element, an error is returned.

Example

Suppose you want to return just the skip-level managers for all employees.

```
=PATHITEM(PATH(Employee[EmployeeID],Employee[ManagerEmployeeID]),Employee[Path Length]-2)
```

190. DAX Functions – PATHITEMREVERSE

Description

Returns the item at the specified position from a string resulting from evaluation of a PATH function.

Positions are counted backwards from the right to the left.

Syntax

PATHITEMREVERSE (<path>, <position>, [<type>])

Parameters

Parameter	Description
path	A text string resulting from an evaluation of a PATH function.
position	An integer expression with the position of the item to be returned. Position is counted backwards from the right to the left.
type	Optional. An enumeration that defines the data type of the result. TEXT or 0: Results are returned with the data type text. (If omitted, this is default). INTEGER or 1: Results are returned as integers.

Return Value

The n^{th} position ascendant in the given path, counting from the current to the oldest.

Remarks

- DAX PATHITEMREVERSE function can be used to get an individual item from a hierarchy resulting from a PATH function.
- This function reverses the standard order of the hierarchy, so that the closest items are listed first. For example, if the PATH function returns a list of managers above an employee in a hierarchy, the PATHITEMREVERSE function returns the employee's immediate manager in position 2 as position 1 contains the employee's ID.

- If the number specified for position is less than one (1) or greater than the number of elements in path, the PATHITEMREVERSE function returns BLANK.
- If the type is not a valid enumeration element, an error is returned.

Example

```
=PATHITEMREVERSE(PATH(Employee[EmployeeID],Employee[ManagerEmployeeID]),2)
```

Returns the immediate managers of each employee.

191. DAX Functions – PATHLENGTH

Description

Returns the number of parents to the specified item in a given PATH result, including self.

Syntax

PATHLENGTH (<path>)

Parameters

Parameter	Description
path	A text expression resulting from evaluation of a PATH function.

Return Value

The number of items that are parents to the specified item in a given PATH result, including the specified item.

Example

```
=PATHLENGTH(PATH(Employee[EmployeeID],Employee[ManagerEmployeeID]))
```

DAX Statistical Functions

192. DAX Statistical Functions – Overview

DAX Statistical functions are very similar to Excel Statistical functions.

Following are the DAX Statistical functions:

- DAX BETA.DIST function
- DAX BETA.INV function
- DAX CHISQ.DIST function
- DAX CHISQ.DIST.RT function
- DAX CHISQ.INV function
- DAX CHISQ.INV.RT function
- DAX CONFIDENCE.NORM function
- DAX CONFIDENCE.T function
- DAX EXPON.DIST function
- DAX GEOMEAN function
- DAX GEOMEANX function
- DAX MEDIAN function
- DAX MEDIANX function
- DAX PERCENTILE.EXC function
- DAX PERCENTILE.INC function
- DAX PERCENTILEX.EXC function
- DAX PERCENTILEX.INC function
- DAX POISSON.DIST function
- DAX RANK.EQ function
- DAX RANKX function
- DAX SAMPLE function
- DAX STDEV.P function
- DAX STDEV.S function
- DAX STDEVX.P function
- DAX STDEVX.S function
- DAX VAR.P function
- DAX VAR.S function
- DAX VARX.P function
- DAX VARX.S function
- DAX XIRR function
- DAX XNPV function

193. DAX Functions – BETA.DIST

Description

Returns the beta distribution.

The beta distribution is commonly used to study the variation in the percentage of something across samples.

DAX BETA.DIST function is new in Excel 2016.

Syntax

BETA.DIST (x, alpha, beta, cumulative, [A, [B]])

Parameters

Parameter	Description
x	The value between A and B at which to evaluate the function.
alpha	A parameter of the distribution.
beta	A parameter of the distribution.
cumulative	Logical value that determines the function's form: TRUE: Returns the cumulative distribution function. FALSE: Returns the probability density function.
A	Optional. A lower bound to the interval of x.
B	Optional. An upper bound to the interval of x.

Return Value

Returns the beta distribution.

A real number, probability, where 0 < probability <= 1.

Remarks

- If any parameter is nonnumeric, BETA.DIST returns the #VALUE! error value.
- If alpha ≤ 0 or beta ≤ 0, BETA.DIST returns the #NUM! error value.
- If x < A, or x > B, or A = B, BETA.DIST returns the #NUM! error value.
- If you omit values for A and B, BETA.DIST uses the standard cumulative beta distribution, so that A = 0 and B = 1.

Example

=BETA.DIST(0.5,9,10, TRUE (),0,1) returns 0.592735290527344.

=BETA.DIST(0.5,9,10, FALSE (),0,1) returns 3.33847045898437.

194. DAX Functions – BETA.INV

Description

Returns the inverse of the beta cumulative probability density function (BETA.DIST).

If BETA.DIST (x ...) = probability, then BETA.INV (probability ...) = x.

DAX BETA.INV function is new in Excel 2016.

Syntax

BETA.INV (probability, alpha, beta, [A, [B]])

Parameters

Parameter	Description
probability	A probability associated with the beta distribution.
alpha	A parameter of the distribution.
beta	A parameter of the distribution.
A	Optional. A lower bound to the interval of x.
B	Optional. An upper bound to the interval of x.

Return Value

Returns the inverse of the beta cumulative probability density function (BETA.DIST).

A real number, x, where A <= x <= B.

Remarks

- If any argument is nonnumeric, BETA.INV returns the #VALUE! error value.
- If alpha ≤ 0 or beta ≤ 0, BETA.INV returns the #NUM! error value.
- If probability ≤ 0 or probability > 1, BETA.INV returns the #NUM! error value.
- If you omit values for A and B, BETA.INV uses the standard cumulative beta distribution, so that A = 0 and B = 1.

Example

=BETA.INV (0.59,9,10,0,1) returns 0.5.

195. DAX Functions – CHISQ.DIST

Description

Returns the chi-squared distribution.

DAX CHISQ.DIST function is new in Excel 2016.

Syntax

CHISQ.DIST (x, deg_freedom, cumulative)

Parameters

Term	Definition
x	The value at which you want to evaluate the distribution.
deg_freedom	The number of degrees of freedom.
cumulative	A logical value that determines the form of the function. TRUE: CHISQ.DIST returns the cumulative distribution function. FALSE: CHISQ.DIST returns the probability density function.

Return Value

The chi-squared distribution.

Remarks

- If deg_freedom is not an integer, it is rounded to the nearest integer.
- If any of the parameters – x, deg_freedom is nonnumeric, CHISQ.DIST returns an error value.
- If any of the parameters – x, deg_freedom is negative, CHISQ.DIST returns an error value.

Example

=CHISQ.DIST(0.5,1, TRUE ()) returns 0.520499877813047.

196. DAX Functions – CHISQ.DIST.RT

Description

Returns the right-tailed probability of the chi-squared distribution. DAX CHISQ.DIST.RT function is new in Excel 2016.

Syntax

CHISQ.DIST.RT (x,deg_freedom)

Parameters

Term	Definition
x	The value at which you want to evaluate the distribution.
deg_freedom	The number of degrees of freedom.

Return Value

The right-tailed probability of the chi-squared distribution.

Remarks

- If deg_freedom is not an integer, it is rounded to the nearest integer.
- If either of the parameters is nonnumeric, CHISQ.DIST.RT returns an error value.
- If either of the parameters is <0, CHISQ.DIST.RT returns an error value.

Example

=CHISQ.DIST.RT (18.307,10) returns 0.0500005890913981.

197. DAX Functions – CHISQ.INV

Description

Returns the inverse of the left-tailed probability of the chi-squared distribution. The chi-squared distribution is commonly used to study variation in the percentage of something across samples. DAX CHISQ.INV function is new in Excel 2016.

Syntax

CHISQ.INV (probability, deg_freedom)

Parameters

Parameter	Description
probability	A probability associated with the chi-squared distribution.
deg_freedom	The number of degrees of freedom.

Return Value

Returns the inverse of the left-tailed probability of the chi-squared distribution.

Remarks

- If any parameter is nonnumeric, CHISQ.INV returns the #VALUE! error value.
- If probability < 0, or probability > 1, CHISQ.INV returns the #NUM! error value.
- If deg_freedom is not an integer, it is truncated.
- If deg_freedom < 1, or deg_freedom > 10^10, CHISQ.INV returns the #NUM! error value.

198. DAX Functions – CHISQ.INV.RT

Description

Returns the inverse of the right-tailed probability of the chi-squared distribution. If CHISQ.DIST.RT (x ...) = probability, then CHISQ.INV.RT (probability ...) = x.

You can use this function to compare the observed results with the expected ones in order to decide whether your original hypothesis is valid. DAX CHISQ.INV.RT function is new in Excel 2016.

Syntax

CHISQ.INV.RT (probability, deg_freedom)

Parameters

Parameter	Description
probability	A probability associated with the chi-squared distribution.
deg_freedom	The number of degrees of freedom.

Return Value

Returns the inverse of the right-tailed probability of the chi-squared distribution.

Remarks

- If any parameter is nonnumeric, CHISQ.INV.RT returns the #VALUE! error value.
- If probability < 0 or probability > 1, CHISQ.INV.RT returns the #NUM! error value.
- If deg_freedom is not an integer, it is truncated.
- If deg_freedom < 1, CHISQ.INV.RT returns the #NUM! error value.
- Given a value for probability, CHISQ.INV.RT seeks that value **x** such that CHISQ.DIST.RT(x, deg_freedom) = probability. Thus, precision of CHISQ.INV.RT depends on precision of CHISQ.DIST.RT.
- CHISQ.INV.RT uses an iterative search technique. If the search has not converged after 64 iterations, the function returns the #N/A error value.

199. DAX Functions – CONFIDENCE.NORM

Description

Returns the confidence interval of a sample mean.

DAX CONFIDENCE.NORM function is new in Excel 2016.

Syntax

CONFIDENCE.NORM (alpha, standard_dev, size)

Parameters

Parameter	Description
alpha	The significance level used to compute the confidence level. The confidence level equals 100 * (1 - alpha) %. For example, if alpha is 0.05, then confidence level is 95%.
standard_dev	The population standard deviation for the data range and is assumed to be known.
size	The sample size.

Return Value

A range of values.

Remarks

The confidence interval is a range of values. The sample mean $\bar{x}$ is at the center of this range and the range is $\bar{x}$ **± CONFIDENCE.NORM**.

For example, if **x** is the sample mean of delivery times for products ordered through the mail, **x ± CONFIDENCE.NORM** is a range of population means.

For any population mean, **μ0**, in this range, the probability of obtaining a sample mean further from **μ0** than **x** is greater than alpha; for any population mean, **μ0**, not in this range, the probability of obtaining a sample mean further from **μ0** than **x** is less than alpha. In other words, assume that we use **x**, standard_dev, and size to construct a two-tailed test at significance level alpha of the hypothesis that the population mean is **μ0**.

Then, we will not reject that hypothesis, where **μ0** is in the confidence interval, and will reject that hypothesis, where **μ0** is not in the confidence interval.

- If any parameter is non-numeric, CONFIDENCE.NORM returns the #VALUE! error value.
- If alpha ≤ 0 or alpha ≥ 1, CONFIDENCE.NORM returns the #NUM! error value.
- If standard_dev ≤ 0, CONFIDENCE.NORM returns the #NUM! error value.
- If size is not an integer, it is truncated.
- If size < 1, CONFIDENCE.NORM returns the #NUM! error value.
- If we assume alpha equals 0.05, we need to calculate the area under the standard normal curve that equals (1 - alpha), or 95 percent. This value is ± 1.96. The confidence interval is therefore:

$$\bar{x} \pm 1.96\left(\frac{\sigma}{\sqrt{n}}\right)$$

Example

=CONFIDENCE.NORM (0.05,2.5,50) returns 0.692951912174839.

200. DAX Functions – CONFIDENCE.T

Description

Returns the confidence interval for a population mean, using a Student's **t** distribution. DAX CONFIDENCE.T function is new in Excel 2016.

Syntax

CONFIDENCE.T (alpha, standard_dev, size)

Parameters

Parameter	Description
alpha	The significance level used to compute the confidence level. The confidence level equals 100 * (1 - alpha) %. For e.g. if alpha is 0.05, then confidence level is 95%.
standard_dev	The population standard deviation for the data range and is assumed to be known.
size	The sample size.

Return Value

Returns the confidence interval for a population mean, using a Student's t distribution.

Remarks

- If any argument is non-numeric, CONFIDENCE.T returns the #VALUE! error value.
- If alpha ≤ 0 or alpha ≥ 1, CONFIDENCE.T returns the #NUM! error value.
- If standard_dev ≤ 0, CONFIDENCE.T returns the #NUM! error value.
- If size is not an integer, it is truncated.
- If size equals 1, CONFIDENCE.T returns #DIV/0! error value.

Example

=CONFIDENCE.T(0.05,1,50) returns 0.28419685549573.

201. DAX Functions – EXPON.DIST

Description

Returns the exponential distribution. DAX EXPON.DIST function is new in Excel 2016.

Syntax

EXPON.DIST (x, lambda, cumulative)

Parameters

Parameter	Description
x	The value of the function.
lambda	The parameter value.
cumulative	A logical value that indicates which form of the exponential function to provide. TRUE: EXPON.DIST returns the cumulative distribution function. FALSE: EXPON.DIST returns the probability density function.

Return Value

Returns the exponential distribution.

Remarks

- If **x** or lambda is non-numeric, EXPON.DIST returns the #VALUE! error value.
- If x < 0, EXPON.DIST returns the #NUM! error value.
- If lambda ≤ 0, EXPON.DIST returns the #NUM! error value.
- The equation for the probability density function is:

$$f(x;\lambda) = \lambda e^{-\lambda x}$$

- The equation for the cumulative distribution function is:

$$F(x;\lambda) = 1 - e^{-\lambda x}$$

Example

=EXPON.DIST(0.2,10, TRUE ()) returns 0.864664716763387.

202. DAX Functions – GEOMEAN

Description

Returns the geometric mean of the numbers in a column. DAX GEOMEAN function is new in Excel 2016.

Syntax

GEOMEAN (<column>)

Parameters

Parameter	Description
column	The column that contains the numbers for which the geometric mean is to be computed.

Return Value

A decimal number.

Remarks

Only the numbers in the column are considered.

Blanks, logical values, and text are ignored.

Example

=GEOMEAN (Sales[Sales Amount])

203. DAX Functions – GEOMEANX

Description

Returns the geometric mean of an expression evaluated for each row in a table. DAX GEOMEANX function is new in Excel 2016.

Syntax

GEOMEANX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.

Return Value

A decimal number.

Remarks

- The first parameter is a table, or an expression that returns a table.
- The second parameter is a column that contains the numbers for which you want to compute the geometric mean, or an expression that evaluates to a column.
- Only the numbers in the column are considered. Blanks, logical values, and text are ignored.

Example

=GEOMEANX (Sales,Sales[Sales Amount])

204. DAX Functions – MEDIAN

Description

Returns the median of numbers in a column. DAX MEDIAN function is new in Excel 2016.

Syntax

MEDIAN (<column>)

Parameters

Parameter	Description
column	The column that contains the numbers for which the median is to be computed.

Return Value

A decimal number.

Remarks

Only the numbers in the column are considered.

Blanks, logical values, and text are ignored.

Example

=MEDIAN (Sales[Sales Amount])

205. DAX Functions – MEDIANX

Description

Returns the median number of an expression evaluated for each row in a table.

Syntax

MEDIANX (<table>, <expression>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.

Return Value

A decimal number.

Remarks

- The first parameter is a table, or an expression that returns a table.
- The second parameter is a column that contains the numbers for which you want to compute the median, or an expression that evaluates to a column.

Only the numbers in the column are considered.

Blanks, logical values, and text are ignored.

Example

```
=MEDIANX (Sales,Sales[Sales Amount])
```

206. DAX Functions – PERCENTILE.EXC

Description

Returns the k^{th} percentile of values in a range, where 0 < k < 1.

DAX PERCENTILE.EXC function is new in Excel 2016.

Syntax

PERCENTILE.EXC (<column>, <k>)

Parameters

Parameter	Description
column	A column containing the values that define relative standing.
k	A number, 0 < k < 1.

Return Value

The k^{th} percentile of values in a range, where 0 < k < 1.

Remarks

- If the column is empty, BLANK () is returned.
- If k <= 0, or k >= 1, it is out of range and an error is returned.
- If k is non-numeric, an error is returned.
- If k is not a multiple of 1/(n + 1), PERCENTILE.EXC will interpolate to determine the value at the k^{th} percentile.

PERCENTILE.EXC will interpolate when the value for the specified percentile is between two values in the array. If it cannot interpolate for the **k** percentile specified, an error is returned.

Example

```
=PERCENTILE.EXC (Sales[Sales Amount],0.25)
```

207. DAX Functions – PERCENTILE.INC

Description

Returns the k^{th} percentile of values in a range, where 0 <= k <= 1.

DAX PERCENTILE.INC function is new in Excel 2016.

Syntax

PERCENTILE.INC (<column>, <k>)

Parameters

Parameter	Description
column	A column containing the values that define relative standing.
k	A number, 0 <= k <= 1.

Return Value

The k^{th} percentile of values in a range, where 0 <= k <= 1.

Remarks

- If the column is empty, BLANK () is returned.
- If **k** is non-numeric or outside the range 0 to 1, an error is returned.
- If **k** is not a multiple of 1 / (n + 1), PERCENTILE.INC will interpolate to determine the value at the k^{th} percentile.

PERCENTILE.INC will interpolate when the value for the specified percentile is between two values in the array. If it cannot interpolate for the **k** percentile specified, an error is returned.

Example

```
=PERCENTILE.INC (Sales[Sales Amount],0.25)
```

208. DAX Functions – PERCENTILEX.EXC

Description

Returns the percentile number of an expression evaluated for each row in a table.

DAX PERCENTILEX.EXC function is new in Excel 2016.

Syntax

PERCENTILEX.EXC (<table>, <expression>, <k>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.
k	A number, 0 < k < 1.

Return Value

The percentile number of an expression evaluated for each row in a table.

Remarks

- If k <= zero, or k >= 1, then it is out of range and an error is returned.
- If k is non-numeric, an error is returned.
- If k is blank, percentile rank of 1 / (n+1) returns the smallest value.
- If k is not a multiple of 1 / (n + 1), PERCENTILEX.EXC will interpolate to determine the value at the k^{th} percentile.

PERCENTILEX.EXC will interpolate when the value for the specified percentile is between two values in the array. If it cannot interpolate for the k percentile specified, an error is returned.

Example

=PERCENTILEX.EXC (Sales,Sales[Sales Amount],0.25)

209. DAX Functions – PERCENTILEX.INC

Description

Returns the percentile number of an expression evaluated for each row in a table.

Syntax

PERCENTILEX.INC (<table>, <expression>, <k>)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.
k	A number, 0 <= k <= 1.

Return Value

The percentile number of an expression evaluated for each row in a table.

Remarks

- If k is zero or blank, percentile rank of 1 / (n - 1) returns the smallest value.
- If k is non-numeric, an error is returned.
- If k < 0, or k > 1, an error is returned.
- If k is not a multiple of 1 / (n - 1), PERCENTILEX.INC will interpolate to determine the value at the k^{th} percentile.

PERCENTILEX.INC will interpolate when the value for the specified percentile is between two values in the array. If it cannot interpolate for the k percentile specified, an error is returned.

Example

=PERCENTILEX.INC (Sales,Sales[Sales Amount],0.25)

210. DAX Functions – POISSON.DIST

Description

Returns the Poisson distribution. A common application of the Poisson distribution is predicting the number of events over a specific time.

Syntax

POISSON.DIST (<x>, <mean>, <cumulative>)

Parameters

Parameter	Description
x	The number of events.
mean	The expected numeric value.
cumulative	A logical value that determines the form of the probability distribution returned. TRUE: POISSON.DIST returns the cumulative Poisson probability that the number of random events occurring will be between zero and x inclusive. FALSE: POISSON.DIST returns the Poisson probability mass function that the number of events occurring will be exactly x.

Return Value

Returns the Poisson distribution.

Remarks

- If x is not an integer, it is truncated.
- If x or mean is non-numeric, POISSON.DIST returns the #VALUE! error value.
- If x < 0, POISSON.DIST returns the #NUM! error value.
- If mean < 0, POISSON.DIST returns the #NUM! error value.

- POISSON.DIST is calculated as follows.
  - For cumulative = FALSE:

$$POISSON = \frac{e^{-\lambda}\ \lambda^{x}}{x!}$$

  - For cumulative = TRUE:

$$CUMPOISSON = \sum_{k=0}^{x} \frac{e^{-\lambda}\ \lambda^{x}}{k!}$$

Example

=POISSON.DIST (2,5, TRUE ()) returns 0.124652019483081.

211. DAX Functions – RANK.EQ

Description

Returns the ranking of a number in a list of numbers.

Syntax

RANK.EQ (<value>, <columnName>, [<order>])

Parameters

Parameter	Description
value	Any DAX expression that returns a single scalar value whose rank is to be found. The expression is to be evaluated exactly once, before the function is evaluated, and its value passed as a parameter.
columnName	The name of a column in a table against which ranks will be determined. It cannot be an expression or a column created using these functions: ADDCOLUMNS, ROW or SUMMARIZE.
order	Optional. Specifies how to rank value, low to high, or high to low. ASC: Ranks in ascending order of columnName. DESC: Ranks in descending order of columnName. If omitted, default is DESC.

Return Value

A number indicating the rank of value among the numbers in columnName.

Remarks

- If a value is not in columnName or a value is a blank, then RANK.EQ returns a blank value.

- Duplicate values of a value receive the same rank value. The next rank value assigned will be the rank value plus the number of duplicate values. For example, if five (5) values receive a rank of 8, then the next value will receive a rank of 13 (8 + 5).

Example

```
=RANK.EQ (1025, Sales[Sales Amount],DESC)
```

212. DAX Functions – RANKX

Description

Returns the ranking of a number in a list of numbers for each row in the table.

Syntax

RANKX (<table>, <expression>, [<value>], [<order>], [<ties>])

Parameters

Parameter	Description
table	Any DAX expression that returns a table of data over which the expression is evaluated.
expression	Any DAX expression that returns a single scalar value. The expression is evaluated for each row of table, to generate all possible values for ranking.
value	Optional. Any DAX expression that returns a single scalar value whose rank is to be found. If omitted, the value of expression at the current row is used instead.
order	Optional. A value that specifies how to rank value, low to high, or high to low. ASC: Ranks in ascending order of columnName. DESC: Ranks in descending order of columnName. If omitted, default is DESC.
ties	Optional. An enumeration that defines how to determine ranking when there are ties. Skip: The next rank value, after a tie, is the rank value of the tie plus the count of tied values. For example, if five (5) values are tied with a rank of 8, then the next value will receive a rank of 13 (8 + 5). This is the default value when ties parameter is omitted. Dense: The next rank value, after a tie, is the next rank value. For example, if five (5) values are tied with a rank of 8, then the next value will receive a rank of 9.

Return Value

- If the parameter value is specified – returns the rank number of value among all possible values of expression evaluated for all rows of table.

- If the parameter value is not specified - returns the rank number of the value of expression at the current row among all possible values of expression evaluated for all rows of table.

Remarks

If an expression or a value evaluates to BLANK, it is treated as a 0 (zero) for all expressions that result in a number, or as an empty text for all text expressions.

If a value is not among all possible values of expression, then RANKX temporarily adds value to the values from expressio*n* and re-evaluates RANKX to determine the proper rank of value.

Example

```
=RANKX (Sales,Sales[Sales Amount],,DESC)
```

213. DAX SAMPLE Function

Description

Returns a sample of **N** rows from the specified table.

Syntax

SAMPLE (<n_value>, <table>, <orderBy_expression>, [<order>], [<orderBy_expression>, [<order>]] …)

Parameters

Parameter	Description
n_value	The number of rows to return as a sample. It is any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context). If a non-integer value (or expression) is entered, the result is cast as an integer.
table	Any DAX expression that returns a table of data from where to extract the n_value number of rows.
orderBy_expression	Optional. Any scalar DAX expression where the result value is evaluated for each row of table.
order	Optional. A value that specifies how to sort orderBy_expression values. 0/FALSE: Sorts in descending order of values of orderBy_expression. 1/TRUE: Sorts in ascending order of values of orderBy_expression. If omitted, default is 0.

Return Value

- A table consisting of a sample of n_value number of rows, if n_value > 0.
- An empty table if n_value <= 0.

Remarks

In order to avoid duplicate values in the sample, the table provided as the second parameter should be grouped by the column used for sorting.

If orderBy_expression and order parameters are provided, the sample will be stable and deterministic, returning the first row, the last row, and evenly distributed rows between them.

If no ordering is specified, the sample will be random, not stable, and not deterministic.

Example

```
=SUMX (SAMPLE (DISTINCTCOUNT (Sales[Month]),
Sales,Sales[Salesperson],ASC),[Sales Amount])
```

214. Functions – DAX STDEV.P

Description

Returns the standard deviation of the entire population.

Syntax

STDEV.P (<ColumnName>)

Parameters

Parameter	Description
ColumnName	The name of a column, usually fully qualified. It cannot be an expression.

Return Value

A real number.

Remarks

DAX STDEV.P function assumes that the column refers to the entire population. If your data represents a sample of the population, then use DAX STDEV.S function.

STDEV.P uses the following formula:

$$\sqrt{\sum \frac{(x - \bar{x})^2}{N}}$$

Where, $\bar{x}$ the average value of x for the entire population, and

N is the population size

Blank rows are filtered out from columnName and not considered in the calculation.

An error is returned if columnName contains less than 2 non-blank rows.

Example

=STDEV.P (Sales[Sales Amount])

215. DAX Functions – STDEV.S

Description

Returns the standard deviation of a sample population.

Syntax

STDEV.S (<ColumnName>)

Parameters

Parameter	Description
ColumnName	The name of an existing column, usually fully qualified. It cannot be an expression.

Return Value

A real number.

Remarks

DAX STDEV.S function assumes that the column refers to a sample of the population. If your data represents the entire population, then use DAX STDEV.P function.

STDEV.S uses the following formula:

$$\sqrt{\sum \frac{(x - \bar{x})^2}{(n-1)}}$$

Where $\bar{x}$ is the average value of **x** for the sample population, and

n is the sample size

Blank rows are filtered out from columnName and not considered in the calculation.

An error is returned if columnName contains less than 2 non-blank rows.

Example

=STDEV.S (West_Sales[Amount])

216. DAX Functions – STDEVX.P

Description

Evaluates the given expression for each row of the given table and returns the standard deviation of expression assuming that the table refers to the entire population.

Syntax

STDEVX.P (<table>, <expression>)

Parameters

Parameter	Description
table	A table or any DAX expression that returns a table of data.
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).

Return Value

A real number.

Remarks

DAX STDEVX.P function evaluates an expression for each row of table and returns the standard deviation of an expression assuming that the table refers to the entire population. If the data in the table represents a sample of the population, you should use DAX STDEVX.S function instead.

STDEVX.P uses the following formula:

$$\sqrt{\sum \frac{(x - \bar{x})^2}{N}}$$

Where, $\bar{x}$ the average value of x for the entire population, and

N is the population size

Blank rows are filtered out from the resulting column and not considered in the calculation.

An error is returned if the resulting column contains less than 2 non-blank rows.

Example

=STDEVX.P (Sales,[Sales Amount])

217. DAX Functions – STDEVX.S Function

Description

Evaluates an expression for each row of the table and returns the standard deviation of an expression, assuming that the table refers to a sample of the population.

Syntax

STDEVX.S (<table>, <expression>)

Parameters

Parameter	Description
table	A table or any DAX expression that returns a table of data.
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).

Return Value

A real number.

Remarks

DAX STDEVX.S function evaluates an expression for each row of the table and returns the standard deviation of expression assuming that the table refers to a sample of the population. If the table represents the entire population, then use DAX STDEVX.P function.

STDEVX.S uses the following formula:

$$\sqrt{\sum \frac{(x - \bar{x})^2}{(n-1)}}$$

Where $\bar{x}$ is the average value of **x** for the sample population, and

n is the sample size

Blank rows are filtered out from the resulting column and not considered in the calculation.

An error is returned, if the resulting column contains less than 2 non-blank rows.

Example

=STDEVX.S (West_Sales,West_Sales[Amount])

218. DAX Functions – VAR.P

Description

Returns the variance of the entire population.

Syntax

VAR.P (<columnName>)

Parameters

Parameter	Description
columnName	The name of a column, usually fully qualified. It cannot be an expression.

Return Value

A real number.

Remarks

DAX VAR.P function assumes that the column refers to the entire population. If your data represents a sample of the population, then use DAX VAR.S function.

VAR.P uses the following formula:

$$\sum \frac{(x - \bar{x})^2}{N}$$

Where, $\bar{x}$ the average value of x for the entire population, and

N is the population size

Blank rows are filtered out from columnName and not considered in the calculation.

An error is returned if columnName contains less than 2 non-blank rows.

Example

=VAR.P (Sales[Sales Amount])

219. DAX Functions – VAR.S

Description

Returns the variance of a sample population.

Syntax

VAR.S (<columnName>)

Parameters

Parameter	Description
columnName	The name of a column, usually fully qualified. It cannot be an expression.

Return Value

A real number.

Remarks

DAX VAR.S function assumes that the column refers to a sample of the population. If your data represents the entire population, then use DAX VAR.P function.

VAR.S uses the following formula:

$$\sum \frac{(x - \bar{x})^2}{(n-1)}$$

Where $\bar{x}$ is the average value of **x** for the sample population, and

n is the sample size

Blank rows are filtered out from columnName and not considered in the calculation.

An error is returned if columnName contains less than 2 non-blank rows.

Example

=VAR.S (West_Sales[Amount])

220. DAX Functions – VARX.P

Description

Evaluates the given expression for each row of the given table, and returns the variance of the expression assuming that the table refers to the entire population.

Syntax

VARX.P (<table>, <expression>)

Parameters

Parameter	Description
table	A table or any DAX expression that returns a table of data.
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).

Return Value

A real number.

Remarks

DAX VARX.P function evaluates <expression> for each row of <table> and returns the variance of <expression> assuming that <table> refers to the entire population. If <table> represents a sample of the population, then use DAX VARX.S function.

VARX.P uses the following formula:

$$\sum \frac{(x - \bar{x})^2}{N}$$

Where, $\bar{x}$ the average value of x for the entire population, and

N is the population size

Blank rows are filtered out from the resulting column and not considered in the calculation.

An error is returned if the resulting column contains less than 2 non-blank rows.

Example

=VARX.P (Sales, [Sales Amount])

221. DAX Functions – VARX.S

Description

Evaluates the given expression for each row of the given table and returns the variance of the expression, assuming that the table refers to a sample of the population.

Syntax

VARX.S (<table>, <expression>)

Parameters

Parameter	Description
table	A table or any DAX expression that returns a table of data.
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated multiple times (for each row/context).

Return Value

A real number.

Remarks

DAX VARX.S function evaluates <expression> for each row of <table> and returns the variance of expression assuming that the table refers to a sample of the population. If the table represents the entire population, then you use DAX VARX.P function.

VARX.S uses the following formula:

$$\sum \frac{(x - \bar{x})^2}{(n-1)}$$

Where $\bar{x}$ is the average value of x for the sample population, and

n is the sample size

Blank rows are filtered out from the resulting column and not considered in the calculation.

An error is returned if the resulting column contains less than 2 non-blank rows.

Example

=VARX.S (West_Sales, West_Sales[Amount])

222. DAX Functions – XIRR Function

Description

Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic. DAX XIRR function is new in Excel 2016.

Syntax

XIRR (<table>, <values>, <dates>, [<guess>])

Parameters

Parameter	Description
table	A table for which the values and dates expressions should be calculated.
values	An expression that returns the cash flow value for each row of the table.
dates	An expression that returns the cash flow date for each row of the table.
guess	Optional. An initial guess for the internal rate of return. If omitted, the default guess of 0.1 is used.

Return Value

Internal rate of return for the given inputs.

If the calculation fails to return a valid result, an error is returned.

Remarks

The value is calculated as the rate that satisfies the following function:

$$\sum_{j=1}^{N} \frac{P_j}{(1+\text{rate})^{\frac{d_j - d_1}{365}}}$$

Where P_j is the j-th payment, d_j is the j-th payment date, and d_1 is the first payment date

The series of cash flow values must contain at least one positive number and one negative number.

Example

=XIRR (CashFlows,[AMOUNT],[DATE])

223. DAX XNPV Function

Description

Returns the present value for a schedule of cash flows that is not necessarily periodic.

DAX XNPV function is new in Excel 2016.

Syntax

XNPV (<table>, <values>, <dates>, <rate>)

Parameters

Parameter	Description
table	A table for which the values and dates expressions should be calculated.
values	An expression that returns the cash flow value for each row of the table.
dates	An expression that returns the cash flow date for each row of the table.
rate	The discount rate to apply to the cash flow for each row of the table.

Return Value

Net present value.

Remarks

The value is calculated as the following summation:

$$\sum_{j=1}^{N} \frac{P_j}{(1+\text{rate})^{\frac{d_j - d_1}{365}}}$$

Where P_j is the j-th payment, d_j is the j-th payment date, and d_1 is the first payment date

The series of cash flow values must contain at least one positive number and one negative number.

Example

=XNPV (CashFlows, [AMOUNT], [DATE],8.5)

DAX Text Functions

224. DAX Text Functions – Overview

DAX Text functions work with tables and columns. With DAX Text functions, you can return a part of a string, search for text within a string, or concatenate string values. You can also control the formats for dates, times, and numbers.

Following are the DAX Text functions:

- DAX BLANK function
- DAX CODE function
- DAX CONCATENATE function
- DAX CONCATENATEX function
- DAX EXACT function
- DAX FIND function
- DAX FIXED function
- DAX FORMAT function
- DAX LEFT function
- DAX LEN function
- DAX LOWER function
- DAX MID function
- DAX REPLACE function
- DAX REPT function
- DAX RIGHT function
- DAX SEARCH function
- DAX SUBSTITUTE function
- DAX TRIM function
- DAX UPPER function
- DAX VALUE function

225. DAX Functions – BLANK Function

Description

Returns a blank.

Syntax

BLANK ()

Parameters

BLANK function has no parameters.

Return Value

A blank.

Remarks

- Blanks are not equivalent to nulls.
- DAX uses blanks for both database nulls and for blank cells in Excel.
- Some DAX functions treat blank cells somewhat differently from Microsoft Excel.
- Blanks and empty strings ("") are not always equivalent, but some operations may treat them as such.

Example

=IF ([No. of Units] =0, BLANK (), [Sales Amount]/ [No. of Units])

This DAX formula returns

- The Unit Price for each Sales Transaction, when No. of Units > 0
- BLANK, when No. of Units = 0

Avoids 'Divide by zero' error.

226. DAX Functions – CODE Function

Description

Returns a numeric code for the first character in a text string. The returned code corresponds to the character set used by your computer. For Windows Operating System, ANSI character set is used.

Syntax

CODE (<text>)

Parameters

Parameter	Description
text	The text for which you want the code of the first character.

Return Value

A numeric code for the first character in the specified text string.

Example

=CODE ("Results") returns the ANSI code of 'R', which is 82.

227. DAX Functions – CONCATENATE

Description

Joins two text strings into one text string.

Syntax

CONCATENATE (<text1>, <text2>)

Parameters

Parameter	Description
text1	The text strings to be joined into a single text string. Text strings can be • Text, or • Numbers, or • Column references.
text2	

Return Value

The concatenated string.

Remarks

DAX CONCATENATE function accepts only two arguments.

If you need to concatenate multiple columns, you can either nest the CONCATENATE functions or use the text concatenation operator (&) to join all of them.

If you want to use text strings directly, rather than using column references, you must enclose each text string in double quotation marks.

Example

=CONCATENATE ("John ","Lever") returns John Lever.

=CONCATENATE (LEFT ([Product], 5), [No. of Units]) returns a calculated column with values – left 5 characters of Product concatenated with No. of Units in the corresponding row.

228. DAX Functions – CONCATENATEX

Description

Concatenates the result of an expression evaluated for each row in a table.

DAX CONCATENATEX function is new in Excel 2016.

Syntax

CONCATENATEX (<table>, <expression>, [<delimiter>], [<OrderBy_Expression1>], [<Order>] …)

Parameters

Parameter	Description
table	The table containing the rows for which the expression will be evaluated.
expression	The expression to be evaluated for each row of the table.
delimiter	Optional. A separator to use during concatenation.
OrderBy_Expression1	Optional. Column by which the values are to be concatenated.
Order	Optional. ASC: To sort in an ascending order. DESC: To sort in a descending order. If omitted, default is ASC.

Return Value

A single text string.

Remarks

CONCATENATEX function takes as its first parameter, a table or an expression that returns a table.

The second parameter is a column that contains the values you want to concatenate, or an expression that returns a value.

Example

CONCATENATEX (Products, [Product],", ", [Product Key], ASC)

Returns Air Purifier, Detergent Powder, Floor Cleaner, Hand Wash, Soap.

=CONCATENATEX (Products, [Product],",", [Product Key], DESC)

Returns Soap, Hand Wash, Floor Cleaner, Detergent Powder, Air Purifier.

=CONCATENATEX (Products, [Product],", ", [Product Key])

Returns Air Purifier, Detergent Powder, Floor Cleaner, Hand Wash, Soap.

229. DAX Functions – EXACT

Description

Compares two text strings and returns TRUE if they are exactly the same, FALSE otherwise.

EXACT is case sensitive, but ignores formatting differences.

Syntax

EXACT (<text1>, <text2>)

Parameters

Term	Definition
text1	The first text string or column that contains text.
text2	The second text string or column that contains text.

Return Value

TRUE or FALSE.

Remarks

You can use EXACT function to test the text being entered into a document.

Example

=EXACT(Results[Sport], [Sport]) returns TRUE, if the value in Sport column in Results table match with the value in Sport column.

230. DAX Functions – FIND

Description

Returns the starting position of one text string within another text string.

DAX FIND function is case sensitive.

Syntax

FIND (<find_text>, <within_text>, [<start_num>], [<NotFoundValue>])

Parameters

Parameter	Description
find_text	The text you want to find. Use double quotes (empty text) to match the first character in within_text. You can use wildcard characters — the question mark (?) and asterisk (*) — in find_text. • A question mark matches any single character. • An asterisk matches any sequence of characters. If you want to find an actual question mark or asterisk, type a tilde (~) before the character.
within_text	The text in which you want to search.
start_num	Optional. The character at which to start the search. If omitted, start_num = 1. The first character in within_text is character number 1.
NotFoundValue	Optional. The value that should be returned when the DAX FIND function does not find find_text in within_text. It should be an Integer or BLANK ().

Return Value

- Number (Integer) that shows the starting position of the find_text in within_text, if it is found.

- If find_text is not found in within_text and NotFoundValue is specified, then that value (an Integer or BLANK ()).

Remarks

- If you provide the argument find_text as a text string, it should be enclosed in double quotation marks.
- If find_text is not found in within_text and NotFoundValue is omitted, DAX FIND function returns #ERROR.
- NotFoundValue should be an Integer or BLANK (). It should not be any other value.
- If you specify start_num that is greater than the start position of the first instance of find_text in within_text, then FIND function returns a number only if a second instance of find_text exists in within_text. Otherwise, it returns NotFoundValue. You can use this to find the duplicated text within a text string.

Example

=FIND ([ProductName], [Product Description],, BLANK ())

This returns a blank, if the product name is not mentioned in the product description.

You can use such verification to ensure that the product description contains the product name at least once.

=FIND ("Powder", [ProductName],, BLANK ())

This returns an integer only if the product name contains the text – Powder. Otherwise, it returns blank.

You can use such verification to find different types of products.

231. DAX Functions – FIXED

Description

Rounds a number and returns the result as text. You can specify the number of digits to the right of the decimal points. You can also specify whether the result be returned with or without commas.

Syntax

FIXED (<number>, [<decimals>, [<no_commas>]])

Parameters

Parameter	Description
number	• The number you want to round and convert to text, or • A column containing a number.
decimals	Optional. The number of digits to the right of the decimal point. If omitted, 2.
no_commas	Optional. Can be specified only if decimals are specified. Otherwise, should be omitted. • True () or 1: Does not display the commas in the returned text. • False () or 0 or omitted: Displays the commas in the returned text.

Return Value

A number represented as text.

Remarks

If no_commas is FALSE or 0 or is omitted, then the returned text includes commas.

You have two options to format a column containing numbers –

- By using a formatting command from the Ribbon.
- Using the DAX FIXED function.

The difference between the two options is that the FIXED function converts its result to text, whereas with the formatting command the result is still a number.

Numbers can never have more than 15 significant digits, but decimals can be as large as 127.

Example

=FIXED ([Sales Amount],2)

232. DAX Functions – FORMAT

Description

Converts a value to text according to the specified format.

Syntax

FORMAT (<value>, <format_string>)

Parameters

Parameter	Description
value	A value or expression that evaluates to a single value.
format_string	A string representing a formatting style. • To format numbers, you can either use predefined numeric formats or create user-defined numeric formats. Look at the sections given at the end of this chapter - Pre-Defined Numeric Formats and Custom Numeric Formats for the FORMAT function. • To format dates and times, you can use predefined date/time formats or create user defined date/time formats. Refer to the tutorial – DAX in this tutorials library for details on formatting dates and times.

Return Value

A string containing value formatted as defined by format_string.

Remarks

- If value is BLANK (), FORMAT function returns an empty string.
- If format_string is BLANK (), the value is formatted with a "General Number" or "General Date" format (according to value data type).

Example

Following table shows the results of the FORMAT function with the first argument value given in the first row and format_string given in the first column. Refer to the sections given below this table to understand the format strings.

	"5"	"-5"	"0.5"	"0"
Zero-length string ("")	5	-5	0.5	0
0	5	-5	1	0
0.00	5.00	-5.00	0.50	0.00
#,##0	5	-5	1	0
$#,##0;($#,##0)	$5	($5)	$1	$0
$#,##0.00;($#,##0.00)	$5.00	($5.00)	$0.50	$0.00
0%	500%	-500%	50%	0%
0.00%	500.00%	-500.00%	50.00%	0.00%
0.00E+00	5.00E+00	-5.00E+00	5.00E-01	0.00E+00
0.00E-00	5.00E00	-5.00E00	5.00E-01	0.00E00
"$#,##0;;\Z\e\r\o"	$5	$-5	$1	Zero

Pre-Defined Numeric Formats for the FORMAT function

The following table identifies the predefined numeric format names that can be used by name for the format style argument of the Format function.

Format String	Description
"General Number"	Displays number with no thousand separators.
"Currency"	Displays number with thousand separators, if appropriate. Displays two digits to the right of the decimal separator. Output is based on system locale settings.
"Fixed"	Displays at least one digit to the left and two digits to the right of the decimal separator.
"Standard"	Displays number with thousand separators, at least one digit to the left and two digits to the right of the decimal separator.
"Percent"	Displays number multiplied by 100 with a percent sign (%) appended immediately to the right. Always displays two digits to the right of the decimal separator.
"Scientific"	Uses standard scientific notation, providing two significant digits.
"Yes/No"	Displays No if number is 0. Otherwise, displays Yes.
"True/False"	Displays False if number is 0. Otherwise, displays True.
"On/Off"	Displays Off if number is 0. Otherwise, displays On.

Custom Numeric Formats for the FORMAT Function

A user defined format expression for numbers can have from one to three sections separated by semicolons.

If the format_string argument of the Format function contains one of the predefined numeric formats, only one section is allowed.

The following table shows how the sections are applied while formatting.

No. of Sections	Format Result
One section only	The format expression applies to all the values.
Two sections	The first section applies to positive values and zeros. The second applies to negative values.
Three sections	The first section applies to positive values. The second section applies to negative values. The third section applies to zeros.

If you include semicolons with nothing between them, the missing section is printed using the format of the positive value.

The following table identifies the characters you can use to create user-defined number formats.

Format Specification	Description
None	Displays the number with no formatting.
0 (zero character)	Digit placeholder. Displays a digit or a zero. If the expression has a digit in the position where the zero appears in the format string, displays the digit. Otherwise, displays a zero in that position. If the number has fewer digits than there are zeros (on either side of the decimal) in the format expression, displays leading or trailing zeros. If the number has more digits to the right of the decimal separator than there are zeros to the right of the decimal separator in the format expression, rounds the number to as many decimal places as there are zeros. If the number has more digits to the left of the decimal separator than there are zeros to the left of the decimal separator in the format expression, displays the extra digits without modification.

#	Digit placeholder. Displays a digit or nothing. • If the expression has a digit in the position where the # character appears in the format string, displays the digit. • Otherwise, displays nothing in that position. This symbol works like the 0-digit placeholder, except that leading and trailing zeros are not displayed if the number has fewer digits than there are # characters on either side of the decimal separator in the format expression.
. (dot character)	Decimal placeholder. The decimal placeholder determines how many digits are displayed to the left and right of the decimal separator. • If the format expression contains only # characters to the left of this symbol, then the numbers smaller than 1 begin with a decimal separator. • To display a leading zero displayed with fractional numbers, use zero as the first digit placeholder to the left of the decimal separator. In some locales, a comma is used as the decimal separator. The actual character used as a decimal placeholder in the formatted output depends on the number format recognized by your system. Thus, you should use the period as the decimal placeholder in your formats even if you are in a locale that uses a comma as a decimal placeholder. The formatted string will appear in the format correct for the locale.
%	Percent placeholder. Multiplies the expression by 100. The percent character (%) is inserted in the position where it appears in the format string.
, (comma character)	Thousand separator. The thousand separator separates thousands from hundreds within a number that has four or more places to the left of the decimal separator. Standard use of the thousand separator is specified if the format contains a thousand separator surrounded by digit placeholders (0 or #). A thousand separator immediately to the left of the decimal separator (whether or not a decimal is specified) or as the rightmost character in the string means "scale the number by dividing it by 1,000, rounding as needed." • Numbers smaller than 1,000 but greater or equal to 500 are displayed as 1, and numbers smaller than 500 are displayed as 0.

	• Two adjacent thousand separators in this position scale by a factor of 1 million, and an additional factor of 1,000 for each additional separator. • Multiple separators in any position other than immediately to the left of the decimal separator or the rightmost position in the string are treated simply as specifying the use of a thousand separator. In some locales, a period is used as a thousand separator. The actual character used as the thousand separator in the formatted output depends on the Number Format recognized by your system. Thus, you should use the comma as the thousand separator in your formats, even if you are in a locale that uses a period as a thousand separator. The formatted string will appear in the format correct for the locale. **Examples**: • "#,0." Uses the thousands separator to format the number 100 million as the string "100,000,000". • "#0,." Uses scaling by a factor of one thousand to format the number 100 million as the string "100000". • "#,0,." Uses the thousands separator and scaling by one thousand to format the number 100 million as the string "100,000".
: (colon character)	Time separator. The time separator separates hours, minutes, and seconds when time values are formatted. In some locales, other characters may be used to represent the time separator. The actual character used as the time separator in formatted output is determined by your system settings.
/ (forward slash character)	Date separator. The date separator separates the day, month, and year when the date values are formatted. In some locales, other characters may be used to represent the date separator. The actual character used as the date separator in formatted output is determined by your system settings.

E- , E+ , e- , e+	**Scientific format**. If the format expression contains at least one-digit placeholder (0 or #) to the left of E-, E+, e-, or e+, the number is displayed in scientific format and E or e is inserted between the number and its exponent. • The number of digit placeholders to the left determines the number of digits in the exponent. • Use E- or e- to place a minus sign next to negative exponents. • Use E+ or e+ to place a minus sign next to negative exponents and a plus sign next to positive exponents. • You must also include digit placeholders to the right of this symbol to get correct formatting.
-+$()	Literal characters. These characters are displayed exactly as typed in the format string. To display a character other than one of those listed, precede it with a backslash (\) or enclose it in double quotation marks (" ").
\ (backward slash character)	Displays the next character in the format string. To display a character that has special meaning as a literal character, precede it with a backslash (\). • The backslash itself is not displayed. • Using a backslash is the same as enclosing the next character in double quotation marks. • To display a backslash, use two backslashes. However, some characters cannot be displayed as literal characters. For example, • The date-formatting and time-formatting characters (a, c, d, h, m, n, p, q, s, t, w, y, /, and :) • The numeric-formatting characters (#, 0, %, E, e, comma, and period) • The string-formatting characters (@, &, <, >, and !)
"ABC"	Displays the string inside the double quotation marks (" "). To include a string in the style argument from within code, you must use Chr(34) to enclose the text (34 is the character code for a quotation mark ("")).

233. DAX Functions – LEFT Function

Description

Returns the specified number of characters from the start of a text string.

Syntax

LEFT (<text>, <num_chars>)

Parameters

Parameter	Description
text	The text string containing the characters you want to extract, or a reference to a column that contains text.
num_chars	Optional. The number of characters you want LEFT to extract. If omitted, default is 1.

Return Value

A text string.

Remarks

DAX works with Unicode and stores all characters as the same length. Therefore, a single function LEFT is enough to extract the characters.

If the num_chars argument is a number that is larger than the number of characters in the text string, DAX LEFT function returns the maximum characters available and does not raise any error.

Example

=CONCATENATE (LEFT([Product], 5), [No. of Units])

Returns a calculated column with the first 5 characters of the Product value concatenated with the value in the No. of Units column in the same row.

234. DAX Functions – LEN Function

Description

Returns the number of characters in a text string.

Syntax

LEN (<text>)

Parameters

Parameter	Description
text	The text whose length you want to find, or a column that contains text.

Return Value

A whole number indicating the number of characters in the text string.

Remarks

Spaces count as characters.

DAX uses Unicode and stores all the characters with the same length. Hence, LEN always counts each character as 1, no matter what the default language setting is.

If you use DAX LEN function with a column that contains non-text values, such as dates or Boolean values, the function implicitly casts the value to text, using the current column format.

Example

=LEN ([Product])

Returns a calculated column with the number of characters in the corresponding Product text values.

235. DAX Functions – LOWER

Description

Converts all letters in a text string to lowercase.

Syntax

LOWER (<text>)

Parameters

Parameter	Description
text	The text you want to convert to lowercase, or a reference to a column that contains text.

Return Value

Text in lowercase.

Remarks

Characters that are not letters will not be changed.

Example

=LOWER ("ABCDE") returns abcde.

=LOWER ("123AB") returns 123ab.

236. DAX Functions – MID Function

Description

Returns a string of characters from the middle of a text string, given a starting position and length.

Syntax

MID (<text>, <start_num>, <num_chars>)

Parameters

Parameter	Description
text	The text string from which you want to extract the characters, or a column that contains text.
start_num	A whole number representing the position of the first character you want to extract. The numbers start from 1 at the beginning of the text.
num_chars	The number of characters to return.

Return Value

A text string.

Remarks

DAX uses Unicode and stores all characters with the same length.

Example

=MID ([Product], 1, 5), and

=LEFT ([Product], 5) return the same characters.

However, you can use DAX MID function to extract text from the middle of the input string.

=MID ([Product],5,5) returns 5 characters starting from the 5th character.

237. DAX Functions – REPLACE Function

Description

Replaces part of a text string, based on the number of characters you specify, with a different text string.

Syntax

REPLACE (<old_text>, <start_num>, <num_chars>, <new_text>)

Parameters

Parameter	Description
old_text	The string of text that contains the characters you want to replace, or a reference to a column that contains text.
start_num	The starting position in the old_text that you want to replace with new_text.
num_chars	The number of characters that you want to replace.
new_text	The replacement text for the specified characters in old_text.

Return Value

A text string.

Remarks

DAX uses Unicode and therefore stores all characters as the same length.

Note: If the argument, num_chars, is a blank or is a reference to a column that evaluates to a blank, then new_text is inserted at the position start_num, without replacing any characters. This is the same behavior as in Excel.

DAX REPLACE function is similar to DAX SUBSTITUTE function.

- You can use REPLACE function, if you want to replace any text of variable length that occurs at a specific position in a text string.

- You can use SUBSTITUTE function, if you want to replace specific text in a text string.

Example

=REPLACE([Product],1,2, [No. of Units])

This returns a calculated column with the first two characters of the Product in a row replaced with the value No. of Units in the same row.

238. DAX Functions – REPT

Description

Repeats text a given number of times.

Syntax

REPT (<text>, <num_times>)

Parameters

Parameter	Description
text	The text you want to repeat.
num_times	A positive number specifying the number of times to repeat the text.

Return Value

A text string.

Remarks

You can use DAX REPT function to fill a cell with a number of instances of a text string.

- If number_times is 0 (zero), DAX REPT function returns a blank.
- If number_times is not an integer, it is rounded.
- If the result of the DAX REPT function is longer than 32,767 characters, an error is returned.
- If you are using a text parameter directly, enclose it in double quotes.

Example

=REPT ("55",2) returns 5555.

=REPT (55,2) returns 5555.

=REPT("AB",5) returns ABABABABAB.

=REPT("AB",2.2) returns ABAB.

=REPT("AB",2.5) returns ABABAB.

=REPT("AB",0) returns Blank.

239. DAX Functions – RIGHT

Description

Returns the last characters in a text string, based on the number of characters you specify.

Syntax

RIGHT (<text>, [<num_chars>])

Parameters

Parameter	Description
text	The text string that contains the characters you want to extract, or a reference to a column that contains text. If the referenced column does not contain text, it will be implicitly cast to text.
num_chars	Optional. The number of characters you want RIGHT to extract. If omitted, default is 1. You can also use a reference to a column that contains numbers.

Return Value

A text string or a calculated column of text strings.

Remarks

RIGHT always counts each character, whether single byte or double byte, as 1, no matter what the default language setting is.

- If num_chars = 0, BLANK is returned.
- If num_chars < 0, #ERROR value is returned.

Example

=RIGHT([Product],2) returns the right two characters of the values in the Product column.
=RIGHT([Product]) returns Blank values.

240. DAX Functions – SEARCH

Description

Returns the number of the character at which a specific character or text string is first found, reading left to right.

Search is case insensitive and accent sensitive.

Syntax

SEARCH (<find_text>, <within_text>, [<start_num>], <NotFoundValue>)

Parameters

Parameter	Description
find_text	The text that you want to find. You can use the wildcard characters question mark (?) and asterisk (*) in find_text. A question mark matches any single character and an asterisk matches any sequence of characters. If you want to find an actual question mark or asterisk, type a tilde (~) before the character.
within_text	The text in which you want to search for find_text, or a column containing text.
start_num	Optional. The character position in within_text at which you want to start searching. If omitted, default is 1.
NotFoundValue	The value that should be returned when find_text is not found in within_text. This can be any specific integer or BLANK ().

Return Value

An integer, or Blank if specified as NotFoundValue.

Remarks

- DAX SEARCH function is case insensitive. Searching for "N" will find the first occurrence of 'N' or 'n'.
- DAX SEARCH function is accent sensitive. Searching for "á" will find the first occurrence of 'á' but not any of the occurrences of 'a', 'à', or the capitalized versions 'A', 'Á'.
- You can use the SEARCH function to determine the location of a text string within another text string, and then use the MID function to return the text, or use the REPLACE function to change the text.
- If find_text cannot be found in within_text, DAX SEARCH function returns NotFoundValue, if given. If omitted, returns #ERROR.
- Nulls in within_text will be interpreted as empty strings.

Example

=SEARCH ("Yes", "Yesterday",, BLANK()) returns 1.

=SEARCH("yes","Yesterday") returns 1.

=SEARCH ("no", "Yesterday",, BLANK()) returns (blank).

=SEARCH("no","Yesterday") returns #ERROR.

=MID("Yesterday",SEARCH("day","Yesterday"),2) returns da.

=REPLACE("Yesterday",SEARCH("day","Yesterday"),3,"fff") returns Yesterfff.

241. DAX Functions – Substitute

Description

Replaces the existing text with a new text in a text string.

Syntax

SUBSTITUTE (<text>, <old_text>, <new_text>, <instance_num>)

Parameters

Parameter	Description
text	The text in which you want to substitute the existing text with a new text, or a reference to a column containing text.
old_text	The existing text that you want to replace.
new_text	The text you want to replace old_text with.
instance_num	Optional. The occurrence of old_text you want to replace. If omitted, every instance of old_text is replaced.

Return Value

A string of text.

Remarks

SUBSTITUTE function is case-sensitive. If the case does not match between find_text and old_text, SUBSTITUTE will not replace the text.

- If find_text is "Not" and within_text contains "not", SUBSTITUTE will not replace the text.

DAX SUBSTITUTE function is similar to DAX REPLACE function.

- You can use SUBSTITUTE function if you want to replace the specific text in a text string.

- You can use REPLACE function, if you want to replace any text of variable length that occurs at a specific position in a text string.

Example

=SUBSTITUTE([Product],"Powder","Lotion") replaces all instances of "Powder" with "Lotion" in the Product column. If "Powder" is not found in any of the rows of the Product column, nothing is changed.

242. DAX Functions – TRIM

Description

Removes all the spaces from the text except for single spaces between words.

Syntax

TRIM (<text>)

Parameters

Parameter	Description
text	The text from which you want spaces removed, or a column that contains text.

Return Value

A text string.

Remarks

You can use DAX TRIM function on the text that you have received from another application that may have irregular spacing.

When you use DAX TRIM function on a column of text with trailing spaces, the results of the TRIM function may not be apparent in the calculated column. However, you can compare the length of the input text and the resulting text to find the difference.

Example

```
=TRIM (" Product    used     for   Fabric Care          ")
```

Returns Product used for Fabric Care.

If you use TRIM function on a column of text values, the results might not be visible. You can use LEN function on both, the parameter column and the resulting column, to compare the lengths of the strings.

243. DAX Functions – UPPER Function

Description

Converts a text string to all uppercase letters.

Syntax

UPPER (<text>)

Parameters

Parameter	Description
text	The text that you want to convert to uppercase, or a reference to a column that contains text.

Return Value

Same text string in upper case.

Remarks

The characters other than alphabets will not be changed.

Example

```
=UPPER("ab")
```

Returns AB.

```
=UPPER("12ab")
```

Returns 12AB.

244. DAX Functions – VALUE Function

Description

Converts a text string that represents a number to a number in numeric data type.

Syntax

VALUE (<text>)

Parameters

Parameter	Description
text	The text to be converted.

Return Value

A number.

Remarks

The text parameter for the DAX VALUE function can be a constant, a number, a date, or a time in DAX data types.

If the text is not in one of these following formats, DAX VALUE function returns an error.

- If the text represents an integer, the integer is returned.
- If the text represents a decimal number with only zero values to the right of the decimal point, then the integer part only is returned.
- If the text represents a decimal number, then the decimal number is returned.

Example

=VALUE ("5") returns 5.

=VALUE ("5.0") returns 5.

=VALUE ("5.1") returns 5.1.

DAX Other Functions

245. DAX Other Functions – Overview

These DAX functions perform unique actions that cannot be defined by any of the categories most other DAX functions belong to.

Following are the DAX Other functions:

- DAX EXCEPT function
- DAX GROUPBY function
- DAX INTERSECT function
- DAX NATURALINNERJOIN function
- DAX NATURALLEFTOUTERJOIN function
- DAX SUMMARIZECOLUMNS function
- DAX UNION function
- DAX VAR function

246. DAX Functions – EXCEPT

Description

Returns the rows of one table which do not appear in another table. DAX EXCEPT function is new in Excel 2016.

Syntax

EXCEPT (<table_expression1>, <table_expression2>)

Parameters

Term	Definition
table_expression1	Any DAX expression that returns a table.
table_expression2	

Return Value

A table that contains the rows of one table minus all the rows of another table.

Remarks

- If a row appears in both tables, that row and its duplicates are not present in the result table.
- If a row appears in only table_expression1, that row and its duplicates will appear in the result table.
- The two tables must have the same number of columns.
- The column names in the result table will match the column names in table_expression1.
- Columns are compared based on positioning, and data comparison with no type coercion.
- The set of rows returned depends on the order of the two expressions.

- The returned table has lineage based on the columns in table_expression1, regardless of the lineage of the columns in the second table. For example, if the first column of first table_expression has lineage to the base column C1 in the Data Model, DAX Except function will reduce the rows based on the availability of values in the first column of table_expression2 and keep the lineage on base column C1 intact.

- The returned table does not include columns from the tables related to table_expression1.

Example

=SUMX (EXCEPT (SalesNewData,SalesOldData),[Sales Amount])

This DAX formula returns the sum of Sales Amount for those transactions that appear in the table SalesNewData but do not appear in the table SalesOldData.

247. DAX Functions – GROUPBY

Description

Returns a table with a set of selected columns. Permits DAX CURRENTGROUP function to be used inside aggregation functions in the extension columns that it adds. GROUPBY attempts to reuse the data that has been grouped making it highly performant.

DAX GROUPBY function is similar to DAX SUMMARIZE function. However, GROUPBY does not do an implicit CALCULATE for any extension columns that it adds.

DAX GROUPBY function is new in Excel 2016.

Syntax

GROUPBY (<table>, [<groupBy_columnName1>], [<name>, <expression>] …)

Parameters

Term	Definition
table	Any DAX expression that returns a table of data.
groupBy_columnName1	The name of an existing column in the table (or in a related table), by which the data is to be grouped. This parameter cannot be an expression.
name	The name given to a new column that is being added to the list of GroupBy columns, enclosed in double quotes.
expression	Any DAX expression that returns a single scalar value, where the expression is to be evaluated for each set of GroupBy values. • It can include any of the "X" aggregation functions, such as SUMX, AVERAGEX, MINX, MAXX, etc. and when one of these functions is used in this way, the table parameter (which is a table expression) can be replaced by CURRENTGROUP function. (Refer Remarks Section for details).

	• However, CURRENTGROUP function can only be used at the top level of table scans in the expression. That means, ○ ABS (SUMX (CURRENTGROUP (), [Column])) is allowed, since ABS does not perform a scan. ○ But, SUMX (<table>, SUMX (CURRENTGROUP () …)) is not allowed. • DAX CALCULATE function and calculated fields are not allowed in the expression.

Return Value

A table with the selected columns for the groupBy_columnName parameters and the grouped by columns designated by the name parameters.

Remarks

The GROUPBY function does the following:

- Start with the specified table (and all related tables in the "to-one" direction).
- Create a grouping using all of the GroupBy columns (which are required to exist in the table from step 1).
- Each group is one row in the result, but represents a set of rows in the original table.
- For each group, evaluate the extension columns being added. Unlike the SUMMARIZE function, an implied CALCULATE is not performed, and the group is not placed into the filter context.

Parameters

- Each column for which you define a name must have a corresponding expression. Otherwise, an error is returned.
    - The first parameter, name, defines the name of the column in the results. The second parameter, expression, defines the calculation performed to obtain the value for each row in that column.
    - Each name must be enclosed in double quotation marks.
- groupBy_columnName must be either in a table or in a related table.
    - The function groups a selected set of rows into a set of summary rows by the values of one or more groupBy_columnName columns. One row is returned for each group.

CURRENTGROUP ()

- CURRENTGROUP function can only be used in an expression that defines a column within the GROUPBY function.
- CURRENTGROUP returns a set of rows from the table parameter of GROUPBY that belong to the current row of the GROUPBY result.
- CURRENTGROUP function takes no parameters and is only supported as the first parameter to one of the following aggregation functions: AverageX, CountAX, CountX, GeoMeanX, MaxX, MinX, ProductX, StDevX.S, StDevX.P, SumX, VarX.S, VarX.P.

Example

```
=GROUPBY (
        Sales,Sales[Salesperson],Products[Product],"Total Sales",
                                        SUMX (CURRENTGROUP (),[Sales Amount])
        )
```

248. DAX Functions – INTERSECT

Description

Returns the row intersection of two tables, retaining the duplicates.

DAX INTERSECT function is new in Excel 2016.

Syntax

INTERSECT (<table_expression1>, <table_expression2>)

Parameters

Term	Definition
table_expression1	Any DAX expression that returns a table.
table_expression2	

Return Value

A table that contains all the rows in table_expression1 that are also in table_expression2.

Remarks

- Intersect is not commutative. That means, Intersect (T1, T2) can have a different result set than Intersect (T2, T1).
- Duplicate rows are retained. That means, if a row appears in table_expression1 and table_expression2, it and all duplicates in table_expression_1 are included in the result set.
- The column names will match the column names in table_expression1.
- Columns are compared based on the positioning, and data comparison with no type coercion.
- The returned table does not include columns from the tables related to table_expression1.

Example

```
=SUMX (INTERSECT (SalesOldData,SalesNewData),[Sales Amount])
```

This DAX formula returns the sum of Sales Amount for all the rows that are present in the table SalesOldData, which are also present in the SalesNewData.

249. DAX Functions – NATURALINNERJOIN

Description

Performs an inner join of a table with another table. The tables are joined on common columns (by name) in the two tables. The two tables are to be related by one of these common columns.

If the two tables have no common column names, or if there is no relation between the two tables, an error is returned.

DAX NATURALINNERJOIN function is new in Excel 2016.

Syntax

NATURALINNERJOIN (<leftJoinTable>, <rightJoinTable>)

Parameters

Term	Definition
leftJoinTable	A table expression defining the table on the left side of the join.
rightJoinTable	A table expression defining the table on the right side of the join.

Return Value

A table which includes only rows for which the values in the common columns specified are present in both tables. The table returned will have the common columns from the left table and other columns from both the tables.

Remarks

- There is no sort order guarantee for the results.
- Columns being joined on must have the same data type in both tables.
- Only columns from the same source table are joined on.
- Strict comparison semantics are used during join. There is no type coercion.

Example

```
=SUMX (NATURALINNERJOIN (Salesperson,Sales),[Sales Amount])
```

250. DAX Functions – NATURALLEFTOUTERJOIN

Description

Performs an outer join of a table with another table. The tables are joined on common columns (by name) in the two tables. The two tables should be related.

If the two tables have no common column names, or if there is no relationship between the two tables, an error is returned.

DAX NATURALLEFTOUTERJOIN function is new in Excel 2016.

Syntax

NATURALLEFTOUTERJOIN (<leftJoinTable>, <rightJoinTable>)

Parameters

Term	Definition
leftJoinTable	A table expression defining the table on the left side of the join.
rightJoinTable	A table expression defining the table on the right side of the join.

Return Value

A table which includes only rows from rightJoinTable for which the values in the common columns specified are also present in leftJoinTable. The table returned will have the common columns from the left table and the other columns from both the tables.

Remarks

- There is no sort order guarantee for the results.
- Columns being joined on must have the same data type in both tables.
- Only columns from the same source table (have the same lineage) are joined on. For example, Products[ProductID], WebSales[ProductdID], StoreSales[ProductdID] with many-to-one relationships between WebSales and StoreSales and the Products table based on the ProductID column, WebSales and StoreSales tables are joined on [ProductID].
- Strict comparison semantics are used during join. There is no type coercion; for example, 1 does not equal 1.0.

Example

```
=SUMX (NATURALLEFTOUTERJOIN (Salesperson,Sales),[Sales Amount])
```

251. DAX Functions – SUMMARIZECOLUMNS

Description

Returns a summary table over a set of groups.

DAX SUMMARIZECOLUMNS function is new in Excel 2016.

Syntax

SUMMARIZECOLUMNS (<groupBy_columnName>, [< groupBy_columnName >] …, [<filterTable>] …, [<name>, <expression>] …)

Parameters

Term	Definition
groupBy_columnName	A fully qualified column reference (Table[Column]) to a base table for which the distinct values are included in the returned table. Each groupBy_columnName column is • cross-joined (different tables), or • auto-existed (same table) with the subsequent specified columns.
filterTable	A table expression which is added to the filter context of all columns specified as groupBy_columnName arguments. The values present in the filter table are used to filter before cross-join/auto-exist is performed.
name	A string representing the column name to use for the subsequent expression specified.
expression	Any DAX expression that returns a single value (not a table).

Return Value

A table which includes the combinations of values from the supplied columns, based on the grouping specified.

- Only rows for which at least one of the supplied expressions return a non-blank value are included in the table returned.
- If all expressions evaluate to BLANK/NULL for a row, that row is not included in the table returned.

Remarks

SUMMARIZECOLUMNS does not guarantee any sort order for the results.

A column cannot be specified more than once in the groupBy_columnName parameter.

Example

```
=SUMX (
        SUMMARIZECOLUMNS (Salesperson[Salesperson],
                        FILTER (Sales, Sales[Region]="South"),
                        "Sales Amount", SUMX (Sales, Sales[Sales Amount])),
        [Sales Amount])
```

252. DAX Functions – UNION

Description

Creates a union (join) table from the specified tables.

DAX UNION function is new in Excel 2016.

Syntax

UNION (<table_expression1>, <table_expression2>, [<table_expression3>] …)

Parameters

Term	Definition
table_expression	Any DAX expression that returns a table.

Return Value

A table that contains all the rows from each of the table expressions.

Remarks

- The tables must have the same number of columns.
- Columns are combined by the position in their respective tables.
- The column names in the return table will match the column names in table_expression1.
- Duplicate rows are retained.
- The returned table has lineage where possible. For example, if the first column of each table_expression has lineage to the same base column C1 in the model, the first column in the UNION result will have lineage to C1. However, if combined columns have a lineage to different base columns, or if there is an extension column, the resulting column in UNION will have no lineage.
- When data types differ, the resulting data type is determined based on the rules for data type coercion.
- The returned table will not contain columns from related tables.

Example

```
=SUMX (UNION (SalesOldData, SalesNewData), [Sales Amount])
```

253. DAX Functions – VAR

Description

Stores the result of an expression as a named variable, which can then be passed as a parameter to other calculated field expressions. Once the resultant values have been calculated for a variable expression, those values do not change, even if the variable is referenced in another expression.

DAX VAR function is new in Excel 2016.

Syntax

VAR <name> = <expression>

Parameters

Term	Definition

name	The name of the variable (identifier).
	• Delimiters are not supported. For e.g. 'varName' or [varName] will result in an error.
	• Supported character set: a-z, A-Z, 0-9.
	  ○ 0-9 are not valid as first character.
	  ○ __ (double underscore) is allowed as a prefix to the identifier name. No other special characters are supported.
	• Reserved keywords not allowed.
	• Names of the existing tables are not allowed.
	• Empty spaces are not allowed.

expression	A DAX expression which returns a scalar or table value.

Return Value

A named variable containing the result of the expression parameter.

Remarks

An expression passed as a parameter to VAR can contain another VAR declaration.

When referencing a variable:

- Calculated fields cannot refer to variables defined outside the calculated field expression, but can refer to functional scope variables defined within the expression.
- Variables can refer to calculated fields.
- Variables can refer to previously defined variables.
- Columns in table variables cannot be referenced via TableName[ColumnName] syntax.

Example

```
=Var SouthSales = SUMX(FILTER(Sales,Sales[Region]="South"),Sales[Sales Amount])
Var EastSales = SUMX(FILTER(Sales,Sales[Region]="East"),Sales[Sales Amount])
return SouthSales+EastSales
```