PARENT CHILD FUNCTIONS OVERVIEW

What are the advantages of Parent and Child Functions?

PATH

PATH function (DAX) -Returns a delimited text with the identifiers of all the parents to the current row, startingwith the oldest or top most until current.

EMPLOYEEKEY	PARENTEMPLOYEEKEY	PATH
14	112	112|14
3	14	112|14|3
11	3	112|14|3|11
13	3	112|14|3|13
162	3	112|14|3|162
117	162	112|14|3|162|117
221	162	112|14|3|162|221
81	162	112|14|3|162|81

Note:

Take values like below

eid ename mgrid

1 x

2 y 1

3 z 2

4 k 2

5 r 4

=Path([eid],[mgrid])

Class room: `Managers_Hie = path(Users[UserID],Users[ManagerID])`

PATHCONTAINS

Returns TRUE if the specified item exists within the specified path.

Syntax:

PATHCONTAINS(<path>, <item>)

Ex:New caluclated column

Add new column to the EMP_MGR Query [This query in Business_Details Sheet]

Scenario :The below gives you employee and hierarchical path

Managers hierarchy=path(EMP_MGR[EID],EMP_MGR[MGRID])

Scenario:The below gives you existence of 1003 in the hierarchy

UserID availability = PATHCONTAINS(path(Users[UserID],Users[ManagerID]),1003)

Scenario:Identifying the length or number of levels managers available.

Pathlength= pathlength(path(Users[UserID],Users[ManagerID]))

PATHLENGTH

PATHLENGTH function (DAX) -Returns the number of levels in a given PATH(), starting at current level until theoldest or top most parent level. In the following example column PathLength is defined as ' =PATHLENGTH([Path]) '; the example includes all data from the Path() example to help understand how this function works.

EMPLOYEEKEY	PARENTEMPLOYEEKEY	PATH	PATHLENGTH
112		112	1
14	112	112|14	2
3	14	112|14|3	3
11	3	112|14|3|11	4
13	3	112|14|3|13	4
162	3	112|14|3|162	4
117	162	112|14|3|162|117	5
221	162	112|14|3|162|221	5
81	162	112|14|3|162|81	5

Levelsbelong to = PATHLENGTH(path(Users[UserID],Users[ManagerID]))

PATHITEM

PATHITEM function (DAX) -Returns the item at the specified position from a PATH() like result, counting from leftto right. In the following example column PathItem - 4th from left is defined as ' =PATHITEM([Path], 4) '; this example returns the EmployeKey at fourth position in the Path string from the left, using the same sample data from the Path() example.

EMPLOYEEKEY	PARENTEMPLOYEEKEY	PATH	PATHITEM - 4TH FROM LEFT
112		112	
14	112	112|14	

EMPLOYEEKEY	PARENTEMPLOYEEKEY	PATH	PATHITEM - 4TH FROM LEFT
3	14	112|14|3	

11	3	112|14|3|11	11
13	3	112|14|3|13	13
162	3	112|14|3|162	162
117	162	112|14|3|162|117	162
221	162	112|14|3|162|221	162
81	162	112|14|3|162|81	162

PATHITEMREVERSE

PATHITEMREVERSE function (DAX) -Returns the item at*position*from a PATH() like function result, countingbackwards from right to left.

In the following example column PathItemReverse - 3rd from right is defined as ' =PATHITEMREVERSE([Path], 3) '; this example returns the EmployeKey at third position in the Path string from the right, using the same sample data from the Path() example.

EMPLOYEEKEY	PARENTEMPLOYEEKEY	PATH	PATHITEMREVERSE - 3RD FROM RIGHT
112		112	
14	112	112|14	
3	14	112|14|3	112
11	3	112|14|3|11	14
13	3	112|14|3|13	14
162	3	112|14|3|162	14
117	162	112|14|3|162|117	3
221	162	112|14|3|162|221	3
81	162	112|14|3|162|81	3

PATHCONTAINS

PATHCONTAINS function (DAX) -Returns**TRUE**if the specified*item*exists within the specified*path*. In thefollowing example column PathContains - employee 162 is defined as ' =PATHCONTAINS([Path], "162") '; this example returns **TRUE** if the given path contains employee 162. This example uses the results from the Path() example above.

EMPLOYEEKEY	PARENTEMPLOYEEKEY	PATH	PATHCONTAINS - EMPLOYEE 162
112		112	FALSE

EMPLOYEEKEY	PARENTEMPLOYEEKEY	PATH	PATHCONTAINS - EMPLOYEE 162
14	112	112|14	FALSE
3	14	112|14|3	FALSE
11	3	112|14|3|11	FALSE
13	3	112|14|3|13	FALSE
162	3	112|14|3|162	TRUE
117	162	112|14|3|162|117	TRUE

WARNING

In SQL Server 2012 Analysis Services, the xVelocity in-memory analytics engine (VertiPaq) does not support the definition of parent-child hierarchies; however, the DAX language provides a set of functions that allows users to explore parent-child hierarchies and to use these hierarchies in formulas.

PATH

Returns a delimited text string with the identifiers of all the parents of the current identifier, starting with the oldest and continuing until current.

Syntax

PATH(<ID_columnName>, <parent_columnName>)

Parameters

TERM	DEFINITION
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 also 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 text or 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

This 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, in an Employees table that contains employees, the managers of employees, and the managers of the managers, you can 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.

- 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_columnName* value is different for those duplicates then PATH() returns an error. In other words, if you have two listings for the same if there are two identical employee IDs that have different manager IDs, the 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

The following example creates a calculated column that lists all the managers for each employee.

```
=PATH(Employee[EmployeeKey], Employee[ParentEmployeeKey])
```

PATHCONTAINS

Returns **TRUE** if the specified *item* exists within the specified *path*.

Syntax

PATHCONTAINS(<path>, <item>)

Parameters

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

A value of **TRUE** if *item* exists in *path*; otherwise **FALSE**.

Remarks

If *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

The following example creates a calculated column that takes a manager ID and checks a set of employees. If the manager ID is among the list of managers returned by the PATH function, the PATHCONTAINS function returns true; otherwise it returns false.

```
=PATHCONTAINS(PATH(Employee[EmployeeKey], Employee[ParentEmployeeKey]), "23")
```

PATHITEM

Returns the item at the specified *position* from a string resulting from evaluation of a PATH function. Positions are counted from left to right.

Syntax

```
PATHITEM(<path>, <position>[, <type>])
```

Parameters

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:

Enumeration	Alternate Enumeration	Description
TEXT	0	Results are returned with the data type text. (default).
INTEGER	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 most distant to current.

Remarks

- This function can be used to return a specific level from a hierarchy returned by a PATH function. For example, you could return just the skip-level managers for all employees.
- If you specify a number for *position* that is less than one (1) or greater than the number of elements in *path*, the PATHITEM function returns BLANK
- If *type* is not a valid enumeration element an error is returned.

Example

The following example returns the third tier manager of the current employee; it takes the employee and manager IDs as the input to a PATH function that returns a string with the hierarchy of parents to current employee. From that string PATHITEM returns the third entry as an integer.

```
=PATHITEM(PATH(Employee[EmployeeKey], Employee[ParentEmployeeKey]), 3, 1)
```

PATHITEMREVERS

Returns the item at the specified *position* from a string resulting from evaluation of a PATH function. Positions are counted backwards from right to left.

Syntax

PATHITEMREVERSE(<path>, <position>[, <type>])

Parameters

path

A text string resulting from evaluation of a PATH function.

position

An integer expression with the position of the item to be returned. Position is counted backwards from right to left.

(Optional) An enumeration that defines the data type of the result:

Enumeration	Alternate Enumeration	Description
TEXT	0	Results are returned with the data type of text. (default)
INTEGER	1	Results are returned with the data type of integer.

Return value

The n-position ascendant in the given path, counting from current to the oldest.

Remarks

- This 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 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 because 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 PATHITEM function returns BLANK.
- If *type* is not a valid enumeration element an error is returned.

Example

The following example takes an employee ID column as the input to a PATH function, and reverses the list of grandparent elements that are returned. The position specified is 3 and the return type is 1; therefore, the PATHITEMREVERSE function returns an integer representing the manager two levels up from the employee.

```
=PATHITEMREVERSE(PATH(Employee[EmployeeKey], Employee[ParentEmployeeKey]), 3, 1)
```

PATHLENGTH

Returns the number of parents to the specified item in a given PATH result, including self.

Syntax

PATHLENGTH(<path>)

Parameters

TERM	DEFINITION

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.

Remarks

This DAX function is not supported for use in DirectQuery mode.

Example

The following example takes an employee ID as input to a PATH function and returns a list of the managers above that employee in the hierarchy, The PATHLENGTH function takes that result and counts the different levels of employees and managers, including the employee you started with.

```
=PATHLENGTH(PATH(Employee[EmployeeKey], Employee[ParentEmployeeKey]))
```

STATISTICAL FUNCTIONS OVERVIEW

Why do we require statistical functions?

ADDCOLUMNS

What is the use of this function?

Adds calculated columns to the given table or table expression.

Syntax

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

Example

The following example returns an extended version of the Product Category table that includes total sales values from the reseller channel and the internet sales.

```
ADDCOLUMNS(ProductCategory,

, "Internet Sales", SUMX(RELATEDTABLE(InternetSales_USD),
  InternetSales_USD[SalesAmount_USD])

, "Reseller Sales", SUMX(RELATEDTABLE(ResellerSales_USD),
  ResellerSales_USD[SalesAmount_USD]))
```

Example:

Example 1: Class room practice: Refer to manual date table creation in the Date Time Functions


```
Example 2: Coursesmode_new = ADDCOLUMNS(DimCourseMode,"count_co", count('Caltable
2017 rows'[CourseID]))
```

AVERAGE

Differences between Average, AverageA, and AverageX?

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

Syntax

AVERAGE(<column>)

Example

The following formula returns the average of the values in the column, ExtendedSalesAmount, in the table, InternetSales.

=AVERAGE(InternetSales[ExtendedSalesAmount])

Related functions

The AVERAGEX function can take as its argument an expression that is evaluated for each row in a table. This enables you to perform calculations and then take the average of the calculated values.

The AVERAGEA function takes a column as its argument, but otherwise is like the Excel function of the same name. By using the AVERAGEA function, you can calculate a mean on a column that contains empty values.

AVERAGEA

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

Syntax

AVERAGEA(<column>)

=AVERAGEA([Amount])

AVERAGEX

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

Syntax

AVERAGEX(<table>,<expression>)

Example

The following example calculates the average freight and tax on each order in the InternetSales table, by first summing Freight plus TaxAmt in each row, and then averaging those sums.

=AVERAGEX(InternetSales, InternetSales[Freight]+ InternetSales[TaxAmt])

If you use multiple operations in the expression used as the second argument, you must use parentheses to control the order of calculations.

Differences between Count, CountA, and CountX?

COUNT

The COUNT function counts the number of cells in a column that contain numbers.

Syntax

COUNT(<column>)

Example

The following example shows how to count the number of numeric values in the column, ShipDate.

=COUNT([ShipDate])

To count logical values or text, use the COUNTA or COUNTAX functions.

COUNTA

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

Syntax

COUNTA(<column>)

Example

The following example returns all rows in the Reseller table that have any kind of value in the column that stores phone numbers. Because the table name does not contain any spaces, the quotation marks are optional.

=COUNTA('Reseller'[Phone])

COUNTAX

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

Syntax

COUNTAX(<table>,<expression>)

Example

The following example counts the number of nonblank rows in the column, Phone, using the table that results from filtering the Reseller table on [Status] = **Active**.

=COUNTAX(FILTER('Reseller',[Status]="Active"),[Phone])

COUNTBLANK

Counts the number of blank cells in a column.

Syntax

COUNTBLANK(<column>)

Example

The following example shows how to count the number of rows in the table Reseller that have blank values for BankName.

=COUNTBLANK(Reseller[BankName])

To count logical values or text, use the COUNTA or COUNTAX functions.

Differences between Count, CountRows?

COUNTROWS

The COUNTROWS function counts the number of rows in the specified table, or in a table defined by an expression.

Syntax

COUNTROWS(<table>)

Example

The following example shows how to count the number of rows in the table Orders. The expected result is 52761.

=COUNTROWS('Orders')

Example

The following example demonstrates how to use COUNTROWS with a row context. In this scenario, there are two sets of data that are related by order number. The table Reseller contains one row for each reseller; the table ResellerSales contains multiple rows for each order, each row containing one order for a particular reseller. The tables are connected by a relationship on the column, ResellerKey.

The formula gets the value of ResellerKey and then counts the number of rows in the related table that have the same reseller ID. The result is output in the column, **CalculatedColumn1**.

=COUNTROWS(RELATEDTABLE(ResellerSales))

The following table shows a portion of the expected results:

RESELLERKEY	CALCULATEDCOLUMN1
1	73
2	70

3	394

COUNTX

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>)

Example

The following formula returns a count of all rows in the Product table that have a list price.

=COUNTX(Product,[ListPrice])

Example

The following formula illustrates how to pass a filtered table to COUNTX for the first argument. The formula uses a filter expression to get only the rows in the Product table that meet the condition, ProductSubCategory = "Caps", and then counts the rows in the resulting table that have a list price. The FILTER expression applies to the table Products but uses a value that you look up in the related table, ProductSubCategory.

=COUNTX(FILTER(Product,RELATED(ProductSubcategory[EnglishProductSubcategoryName])="Caps", Product[ListPrice])

CROSSJOIN

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

Syntax

CROSSJOIN(<table>, <table>[, <table>]…)

Example

CROSSJOIN(Colors, Stationery)

DISTINCTCOUNT

The DISTINCTCOUNT function counts the number of distinct values in a column.

Syntax

DISTINCTCOUNT(<column>)

Example

The following example shows how to count the number of distinct sales orders in the column ResellerSales_USD[SalesOrderNumber].

=DISTINCTCOUNT(ResellerSales_USD[SalesOrderNumber])

Difference between GENERATE and GENERATE ALL?

Based on row section in table1, it does Cartesian join with table2. Whereas ALL ignore all filters and perform

GENERATE

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>)

Difference between GENERATE and GENERATESERIES?

Example

In the following example the user wants a summary table of the sales by Region and Product Category for the Resellers channel, like the following table:

SalesTerritory[SalesTerritoryGroup]	ProductCategory[ProductCategoryName]	[Reseller Sales]
Europe	Accessories	$ 142,227.27
Europe	Bikes	$ 9,970,200.44
Europe	Clothing	$ 365,847.63
Europe	Components	$ 2,214,440.19
North America	Accessories	$ 379,305.15
North America	Bikes	$ 52,403,796.85
North America	Clothing	$ 1,281,193.26
North America	Components	$ 8,882,848.05
Pacific	Accessories	$ 12,769.57
Pacific	Bikes	$ 710,677.75
Pacific	Clothing	$ 22,902.38
Pacific	Components	$ 108,549.71

The following code produces the above table:

```
GENERATE(
SUMMARIZE(SalesTerritory, SalesTerritory[SalesTerritoryGroup])
,SUMMARIZE(ProductCategory
,  [ProductCategoryName]

,  "Reseller Sales", SUMX(RELATEDTABLE(ResellerSales_USD), ResellerSales_USD[SalesAmount_USD])
)

)
```

1. The first SUMMARIZE statement, SUMMARIZE(SalesTerritory, SalesTerritory[SalesTerritoryGroup]) , produces a table of territory groups, where each row is a territory group, as shown below:

SALESTERRITORY[SALESTERRITORYGROUP]
North America
Europe
Pacific
NA

2. The second SUMMARIZE statement,

```
SUMMARIZE(ProductCategory, [ProductCategoryName], "Reseller Sales",
SUMX(RELATEDTABLE(ResellerSales_USD), ResellerSales_USD[SalesAmount_USD]))
```

, produces a table of Product Category groups with the Reseller sales for each group, as shown below:

PRODUCTCATEGORY[PRODUCTCATEGORYNAME]	[RESELLER SALES]
Bikes	$ 63,084,675.04
Components	$ 11,205,837.96
Clothing	$ 1,669,943.27
Accessories	$ 534,301.99

3. However, when you take the above table and evaluate it under the context of each row from the territory groups table, you obtain different results for each territory.

GENERATEALL

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

TERM	DEFINITION
table1	Any DAX expression that returns a table.
table2	Any DAX expression that returns a table.

Return value

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.

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

Example

In the following example, the user wants a summary table of the sales by Region and Product Category for the Resellers channel, like the following table:

SALESTERRITORY[SALESTERRITORYGROUP]	PRODUCTCATEGORY[PRODUCTCATEGORYNAME]	[RESELLER SALES]
Europe	Accessories	$ 142,227.27
Europe	Bikes	$ 9,970,200.44
Europe	Clothing	$ 365,847.63
Europe	Components	$ 2,214,440.19
NA	Accessories	

SALESTERRITORY[SALESTERRITORYGROUP]	PRODUCTCATEGORY[PRODUCTCATEGORYNAME]	[RESELLER SALES]

NA	Bikes	
NA	Clothing	
NA	Components	
North America	Accessories	$ 379,305.15
North America	Bikes	$ 52,403,796.85
North America	Clothing	$ 1,281,193.26
North America	Components	$ 8,882,848.05
Pacific	Accessories	$ 12,769.57
Pacific	Bikes	$ 710,677.75
Pacific	Clothing	$ 22,902.38
Pacific	Components	$ 108,549.71

The following code produces the above table:

```
GENERATEALL(
SUMMARIZE(SalesTerritory, SalesTerritory[SalesTerritoryGroup])
,SUMMARIZE(ProductCategory
, [ProductCategoryName]
, "Reseller Sales", SUMX(RELATEDTABLE(ResellerSales_USD), ResellerSales_USD[SalesAmount_USD])
)
)
```


1. The first SUMMARIZE produces a table of territory groups, where each row is a territory group, like those listed below:

SALESTERRITORY[SALESTERRITORYGROUP]
North America
Europe
Pacific
NA

2. The second SUMMARIZE produces a table of Product Category groups with the Reseller sales for each group, as shown below:

PRODUCTCATEGORY[PRODUCTCATEGORYNAME]	[RESELLER SALES]
Bikes	$ 63,084,675.04

PRODUCTCATEGORY[PRODUCTCATEGORYNAME]	[RESELLER SALES]
Components	$ 11,205,837.96

Clothing	$ 1,669,943.27
Accessories	$ 534,301.99

3. However, when you take the above table and evaluate the table under the context of each row from the territory groups table, you obtain different results for each territory.Returns the geometric mean of the numbers in a column.

GEOMEAN

To return the geometric mean of an expression evaluated for each row in a table, use GEOMEANX function (DAX).

Syntax

GEOMEAN(<column>)

Parameters

TERM	DEFINITION
	geometric mean is to be computed.

Return value

A decimal number.

Remarks

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

GEOMEAN(Table[Column]) is equivalent to GEOMEANX(Table, Table[Column])

Example

The following computes the geometric mean of the Return column in the Investment table:

=GEOMEAN(Investment[Return])

GEOMEANX

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

To return the geometric mean of the numbers in a column, use GEOMEAN function (DAX).

Syntax

GEOMEANX(<table>, <expression>)

Parameters

TERM	DEFINITION
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 GEOMEANX function takes as its first argument a table, or an expression that returns a table. The second argument 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 counted. Blanks, logical values, and text are ignored.

Example

The following computes the geometric mean of the ReturnPct column in the Investments table:

=GEOMEANX(Investments, Investments[ReturnPct] + 1)

Difference between MAX, MAXA, and MAXX?

MAX

Returns the largest numeric value in a column, or between two scalar expressions.

Syntax

MAX(<column>)

MAX(<expression1>, <expression2>)

Parameters

TERM	DEFINITION
column	The column in which you want to find the largest numeric value.
expression	Any DAX expression which returns a single numeric value.

Return value

A decimal number.

Remarks

When evaluating a single column that contains numeric values, if the column contains no numbers, MAX returns a blank. If you want to evaluate values that are not numbers, use the MAXA function.

When comparing two expressions, blank is treated as 0 when comparing. That is, Max(1, Blank()) returns 1, and Max(-1, Blank()) returns 0. If both arguments are blank, MAX returns a blank. If either expression returns a value which is not allowed, MAX returns an error.

Example

The following example returns the largest value found in the ExtendedAmount column of the InternetSales table.

=MAX(InternetSales[ExtendedAmount])

Example

The following example returns the largest value between the result of two expressions.

=Max([TotalSales], [TotalPurchases])

MAXA

Returns the largest value in a column. Logical values and blanks are counted.

Syntax

MAXA(<column>)

Difference between MIN, MINA, and MINX?

Parameters

TERM	DEFINITION
column	The column in which you want to find the largest value.

Return value

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; 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).
If you do not want to include logical values and blanks as part of the calculation, use the MAX function.

Example

The following example returns the greatest value from a calculated column, named **ResellerMargin**, that computes the difference between list price and reseller price.

```
=MAXA([ResellerMargin])
```


Example

The following example returns the largest value from a column that contains dates and times. Therefore, this formula gets the most recent transaction date.

```
=MAXA([TransactionDate])
```

MAXX

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

Syntax

MAXX(<table>,<expression>)

Parameters

TERM	DEFINITION
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 **table** argument to the MAXX function can be a table name, or an expression that evaluates to a table. The second argument indicates the expression to be evaluated for each row of the table.

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. If you want to include non-numeric values in the formula, use the MAXA function.
If a blank cell is included in the column or expression, MAXX returns an empty column.

Example

The following formula uses an expression as the second argument to calculate the total amount of taxes and shipping for each order in the table, InternetSales. The expected result is 375.7184.

```
=MAXX(InternetSales, InternetSales[TaxAmt]+ InternetSales[Freight])
```

Example

The following formula first filters the table InternetSales, by using a FILTER expression, to return a subset of orders for a specific sales region, defined as [SalesTerritory] = 5. The MAXX function then evaluates the expression used as the second argument for each row of the filtered table, and returns the highest amount for taxes and

shipping for just those orders. The expected result is 250.3724.

```
=MAXX(FILTER(InternetSales,[SalesTerritoryCode]="5"), InternetSales[TaxAmt]+ InternetSales[Freight])
```

MEDIAN

Returns the median of numbers in a column.

To return the median of an expression evaluated for each row in a table, use MEDIANX function (DAX).

Syntax

```
MEDIAN(<column>)
```


Parameters

TERM	DEFINITION
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 counted. Blanks, logical values, and text are ignored.
MEDIAN(Table[Column]) is equivalent to MEDIANX(Table, Table[Column]).

Example

The following computes the median of a column named Age in a table named Customers:

```
=MEDIAN( Customers[Age] )
```

MEDIANX

Returns the median number of an expression evaluated for each row in a table.
To return the median of numbers in a column, use MEDIAN function (DAX).

Syntax

MEDIANX(<table>, <expression>)

Parameters

TERM	DEFINITION
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 MEDIANX function takes as its first argument a table, or an expression that returns a table. The second argument 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 counted.
Logical values and text are ignored.
MEDIANX does not ignore blanks; however, MEDIAN does ignore blanks

Example

The following computes the median age of customers who live in the USA.

=MEDIANX(FILTER(Customers, RELATED(Geography[Country]="USA")), Customers[Age])

MIN

Returns the smallest numeric value in a column, or between two scalar expressions. Ignores logical values and text.

Syntax

MIN(<column>)
MIN(<expression1>, <expression2>)

Parameters

TERM	DEFINITION
column	The column in which you want to find the smallest numeric value.
expression	Any DAX expression which returns a single numeric value.

Return value

A decimal number.

Remarks

The MIN function takes a column or two expressions as an argument, and returns the smallest numeric value.
The following types of values in the columns are counted:

- Numbers
- Dates
- Blanks
- If the column contains no numerical data, MIN returns blanks.

When evaluating a column, empty cells, logical values, and text are ignored. If you want to include logical values and text representations of numbers in a reference as part of the calculation, use the MINA function.

When comparing expressions, blank is treated as 0 when comparing. That is, Min(1,Blank()) returns 0, and Min(- 1, Blank()) returns -1. If both arguments are blank, MIN returns a blank. If either expression returns a value which is not allowed, MIN returns an error.

Example

The following example returns the smallest value from the calculated column, ResellerMargin.

=MIN([ResellerMargin])

Example

The following example returns the smallest value from a column that contains dates and times, TransactionDate.

This formula therefore returns the date that is earliest.

=MIN([TransactionDate])

Example

The following example returns the smallest value from the result of two scalar expressions.

=Min([TotalSales], [TotalPurchases])

MINA

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

Syntax

MINA(<column>)

Parameters

TERM	DEFINITION
column	The column for which you want to find the minimum value.

Return value

A decimal number.

Remarks

The MINA function takes as argument a column that contains numbers, and determines the smallest value as follows:

- If the column contains no numeric values, MINA returns 0 (zero).
- Rows in the column that evaluates to logical values, such as TRUE and FALSE are treated as 1 if TRUE and 0 (zero) if FALSE.
- Empty cells are ignored.

If you do not want to include logical values and text as part of the calculation, use the MIN function instead.

Example

The following expression returns the minimum freight charge from the table, InternetSales.

=MINA(InternetSales[Freight])

Example

The following expression returns the minimum value in the column, PostalCode. Because the data type of the column is text, the function does not find any numeric values, and the formula returns zero (0).

=MINA([PostalCode])

MINX

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

Syntax

MINX(<table>, < expression>)

Parameters

TERM	DEFINITION
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 MINX function takes as its first argument a table, or an expression that returns a table. The second argument contains the expression that is evaluated for each row of the table.

The MINX function evaluates the results of the expression in the second argument 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.

If you want to include logical values and text representations of numbers in a reference as part of the calculation, use the MINA function.

Example

The following example filters the table, InternetSales, and returns only rows for a specific sales territory. The formula then finds the minimum value in the column, Freight.

=MINX(FILTER(InternetSales, [SalesTerritoryKey] = 5),[Freight])

Example

The following example uses the same filtered table as in the previous example, but instead of merely looking up values in the column for each row of the filtered table, the function calculates the sum of two columns, Freight and TaxAmt, and returns the smallest value resulting from that calculation.

=MINX(FILTER(InternetSales, InternetSales[SalesTerritoryKey] = 5), InternetSales[Freight] + InternetSales[TaxAmt])

NORM.DIST

Returns the normal distribution for the specified mean and standard deviation.

Syntax

NORM.DIST(X, Mean, Standard_dev, Cumulative)

Example

```
EVALUATE { NORM.DIST(42, 40, 1.5, TRUE) }
```

Returns

[VALUE]
0.908788780274132

NORM.INV

The inverse of the normal cumulative distribution for the specified mean and standard deviation.

Syntax

```
NORM.INV(Probability, Mean, Standard_dev)
```

Example

```
EVALUATE { NORM.INV(0.908789, 40, 1.5) }
```

Returns

[VALUE]
42.00000200956628780274132

NORM.S.DIST

Returns the standard normal distribution (has a mean of zero and a standard deviation of one).

Syntax

```
NORM.S.DIST(Z, Cumulative)
```

Example

```
EVALUATE { NORM.S.DIST(1.333333, TRUE) }
```

Returns

[VALUE]
0.908788725604095

NORM.S.INV

Returns the inverse of the standard normal cumulative distribution. The distribution has a mean of zero and a standard deviation of one.

Syntax

```
NORM.S.INV(Probability)
```

Example

```
EVALUATE { NORM.S.INV(0.908789) }
```

Returns

[VALUE]
1.33333467304411

PERCENTILEX.EXC

Returns the k-th percentile of values in a range, where k is in the range 0..1, exclusive.

To return the percentile number of an expression evaluated for each row in a table, use PERCENTILEX.EXCfunction (DAX).

Syntax

PERCENTILE.EXC(<column>, <k>)

Return value

The k-th percentile of values in a range, where k is in the range 0..1, exclusive.

PERCENTILEX.INC

Returns the k-th percentile of values in a range, where k is in the range 0..1, inclusive.

To return the percentile number of an expression evaluated for each row in a table, use PERCENTILEX.INCfunction (DAX).

Syntax

PERCENTILE.INC(<column>, <k>)

Return value

The k-th percentile of values in a range, where k is in the range 0..1, inclusive.

PERCENTILEX.EXC

Returns the percentile number of an expression evaluated for each row in a table.
To return the percentile of numbers in a column, use PERCENTILE.EXC function (DAX).

Syntax

PERCENTILEX.EXC(<table>, <expression>, k)

Return value

The percentile number of an expression evaluated for each row in a table.

PERCENTILEX.INC

Returns the percentile number of an expression evaluated for each row in a table.
To return the percentile of numbers in a column, use PERCENTILE.INC function (DAX).

Syntax

PERCENTILEX.INC(<table>, <expression>;, k)

Return value

The percentile number of an expression evaluated for each row in a table.

PERMUT

Returns the number of permutations for a given number of objects that can be selected from number objects. A permutation is any set or subset of objects or events where internal order is significant. Permutations are different from combinations, for which the internal order is not significant. Use this function for lottery-style probability calculations.

Syntax

PERMUT(number, number_chosen)

Return value

Returns the number of permutations for a given number of objects that can be selected from number objects

Both arguments are truncated to integers.
If number or number_chosen is nonnumeric, PERMUT returns the #VALUE! error value.
If number ≤ 0 or if number_chosen < 0, PERMUT returns the #NUM! error value.
If 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

FORMULA	DESCRIPTION	RESULT
=PERMUT(3,2)	Permutations possible for a group of 3 objects where 2 are chosen.	6

POISSON.DIST

Returns the Poisson distribution. A common application of the Poisson distribution is predicting the number of events over a specific time, such as the number of cars arriving at a toll plaza in 1 minute.

Syntax

POISSON.DIST(x,mean,cumulative)

Return value

Returns the Poisson distribution.

Remarks

If x is not an integer, it is rounded.
If x or mean is nonnumeric, 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:

$$CUMPOISSON = \sum_{k=0}^{x} \frac{e^{-\lambda}\,\lambda^{k}}{k!}$$

RANK.EQ

Difference between RANK.EQ and RANKX?

Returns the ranking of a number in a list of numbers.

Syntax

RANK.EQ(<value>, <columnName>[, <order>])

- **Return value**

A number indicating the rank of *value* among the numbers in *columnName*.

Example
The following example creates a calculated column that ranks the values in SalesAmount_USD, from the *InternetSales_USD* table, against all numbers in the same column.

=RANK.EQ(InternetSales_USD[SalesAmount_USD], InternetSales_USD[SalesAmount_USD])

Example

The following example ranks a subset of values against a given sample. Assume that you have a table of local students with their performance in a specific national test and, also, you have the entire set of scores in that national test. The following calculated column will give you the national ranking for each of the local students.

=RANK.EQ(Students[Test_Score], NationalScores[Test_Score])

RANKX

Returns the ranking of a number in a list of numbers for each row in the *table* argument.

Syntax

RANKX(<table>, <expression>[, <value>[, <order>[, <ties>]]])

Return value

The rank number of *value* among all possible values of *expression* evaluated for all rows of *table* numbers.

Remarks

- If *expression* or *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 *value* is not among all possible values of *expression* then RANKX temporarily adds *value* to the values from *expression* and re-evaluates RANKX to determine the proper rank of *value*.
- Optional arguments might be skipped by placing an empty comma (,) in the argument list, i.e.

RANKX(Inventory, [InventoryCost],,,"Dense")

Example

The following calculated column in the Products table calculates the sales ranking for each product in the Internet channel.

=RANKX(ALL(Products), SUMX(RELATEDTABLE(InternetSales), [SalesAmount]))

Scenario: Based on each course discount fee values providing rank

Ranking = RANKX(ALL(DimCourse), SUMX(RELATEDTABLE(FactPayments), FactPayments[Discount_Fee]))

NOTE: CREATE A NEW COLUMN WITH RANK EXPRESSION

ROW

Difference between Value,Row, and Data Table?

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

TERM	DEFINITION
name	The name given to the column, enclosed in double quotes.
expression	Any DAX expression that returns a single scalar value to populate. *name*.

Return value

A single row table

Remarks

Arguments must always come in pairs of *name* and *expression*.

Example

The following example returns a single row table with the total sales for internet and resellers channels.

```
ROW("Internet Total Sales (USD)",
    SUM(InternetSales_USD[SalesAmount_USD]), "Resellers Total Sales
    (USD)", SUM(ResellerSales_USD[SalesAmount_USD]))
```


The code is split in two lines for readability purposes

SAMPLE

Returns a sample of N rows from the specified table.

Difference between Random and Sample?

Syntax

SAMPLE(<n_value>, <table>, <orderBy_expression>, [<order>[, <orderBy_expression>, [<order>]]…])

Display sample 3 records

Sample 3 records = sample(3,'Sales',Sales[CountryID])

SELECTCOLUMNS

Difference between SELECTCOLUMNS and ADDCOLUMNS?

Adds calculated columns to the given table or table expression.

Syntax

SELECTCOLUMNS(<table>, <name>, <scalar_expression> [, <name>, <scalar_expression>]…)

Example: **SELECTCOLUMNS(Info, "StateCountry", [State]&", "&[Country])**

SIN

Returns the sine of the given angle.

Syntax

SIN(NUMBER)

Return value

Returns the sine of the given angle.

Parameters

TERM	DEFINITION
number	Required. The angle in radians for which you want the sine.

Remarks

If your argument is in degrees, multiply it by PI()/180 or use the RADIANS function to convert it to radians.

Example

FORMULA	DESCRIPTION	RESULT
=SIN(PI())	Sine of pi radians (0, approximately).	0.0
=SIN(PI()/2)	Sine of pi/2 radians.	1.0
=SIN(30*PI()/180)	Sine of 30 degrees.	0.5
=SIN(RADIANS(30))	Sine of 30 degrees.	0.5

SINH

Returns the hyperbolic sine of a number

Syntax

SINH(NUMBER)

Parameters

TERM	DEFINITION
number	Required. Any real number.

Return value

Returns the hyperbolic sine of a number.

Remarks

The formula for the hyperbolic sine is:

$$SINH(z) = \frac{e^{z} - e^{-z}}{2}$$

Example

FORMULA	DESCRIPTION	RESULT
=2.868*SINH(0.0342*1.03)	Probability of obtaining a result of less than 1.03 seconds.	0.1010491

STDEV.S

Returns the standard deviation of a sample population.

Syntax

STDEV.S(<ColumnName>)

Example

The following example shows the formula for a measure that calculates the standard deviation of the column, SalesAmount_USD, when the table InternetSales_USD is the sample population.

```
=STDEV.S(InternetSales_USD[SalesAmount_USD])
```

STDEV.P

Returns the standard deviation of the entire population.

Syntax

```
STDEV.P(<ColumnName>)
```

Example

The following example shows the formula for a measure that calculates the standard deviation of the column, SalesAmount_USD, when the table InternetSales_USD is the entire population.

```
=STDEV.P(InternetSales_USD[SalesAmount_USD])
```

STDEVX.S

Returns the standard deviation of a sample population.

Syntax

```
STDEVX.S(<table>, <expression>)
```

Example

The following example shows the formula for a calculated column that estimates the standard deviation of the unit price per product for a sample population, when the formula is used in the Product table.


```
=STDEVX.S(RELATEDTABLE(InternetSales_USD),
InternetSales_USD[UnitPrice_USD] –
(InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity
]))
```

STDEVX.P

Returns the standard deviation of the entire population.

Syntax

STDEVX.P(<table>, <expression>)

Example

The following example shows the formula for a calculated column that calculates the standard deviation of the unit price per product, when the formula is used in the *Product* table.

```
=STDEVX.P(RELATEDTABLE(InternetSales_USD),
InternetSales_USD[UnitPrice_USD] –
(InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity]))
```

SQRTPI

Returns the square root of (number * pi)

Syntax

```
SQRTPI(NUMBER)
```

Remarks

FORMULA	DESCRIPTION	RESULT
=SQRTPI(1)	Square root of pi.	1.772454
=SQRTPI(2)	Square root of 2 * pi.	2.506628

SUMMARIZE

Difference between Group, Summarize, and summarize columns?

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

Syntax

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

Parameters

TERM	DEFINITION
table	Any DAX expression that returns a table of data.
groupBy_columnName	(Optional) 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* arguments and the summarized columns designed by the name arguments.

Remarks

1. Each column for which you define a name must have a corresponding expression; otherwise, an error is returned. The first argument, name, defines the name of the column in the results. The second argument, expression, defines the calculation performed to obtain the value for each row in that column.
2. groupBy_columnName must be either in *table* or in a related table to *table*.
3. Each name must be enclosed in double quotation marks.
4. 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

The following example returns a summary of the reseller sales grouped around the calendar year and the product category name, this result table allows you to do analysis over the reseller sales by year and product category.

```
SUMMARIZE(ResellerSales_USD
DateTime[CalendarYear]
,  ProductCategory[ProductCategoryName]
,  "Sales Amount (USD)", SUM(ResellerSales_USD[SalesAmount_USD])

,  "Discount Amount (USD)", SUM(ResellerSales_USD[DiscountAmount])

)
```

The following table shows a preview of the data as it would be received by any function expecting to receive a table:

PRODUCTCATEGORY[PRODUC

DATETIME[CALENDARYEAR]	TCATEGORYNAME]	[SALES AMOUNT (USD)]	[DISCOUNT AMOUNT (USD)]
2008	Bikes	12968255.42	36167.6592
2005	Bikes	6958251.043	4231.1621
2006	Bikes	18901351.08	178175.8399
2007	Bikes	24256817.5	276065.992
2005	Components	574256.9865	0
2006	Components	3428213.05	948.7674
2007	Components	5195315.216	4226.0444
2008	Clothing	366507.844	4151.1235
2005	Clothing	31851.1628	90.9593
2006	Clothing	455730.9729	4233.039
2007	Clothing	815853.2868	12489.3835
2005	Accessories	18594.4782	4.293
2006	Accessories	86612.7463	1061.4872
2007	Accessories	275794.8403	4756.6546

Advanced SUMMARIZE options

SUMMARIZE with ROLLUP

The addition of the ROLLUP() syntax modifies the behavior of the SUMMARIZE function by adding roll-up rows to the result on the groupBy_columnName columns.

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

ROLLUP parameters

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.
Note: All other SUMMARIZE parameters are explained before and not repeated here for brevity.

- **Remarks**

  The columns mentioned in the ROLLUP expression cannot be referenced as part of a *groupBy_columnName* columns.

Example

The following example adds roll-up rows to the Group-By columns of the SUMMARIZE function call.

SUMMARIZE(ResellerSales_USD

, ROLLUP(DateTime[CalendarYear], ProductCategory[ProductCategoryName])

, "Sales Amount (USD)", SUM(ResellerSales_USD[SalesAmount_USD])

, "Discount Amount (USD)", SUM(ResellerSales_USD[DiscountAmount])

)

The following table shows a preview of the data as it would be received by any function expecting to receive a table:

DATETIME[CALENDARYEAR]	PRODUCTCATEGORY[PRODUC TCATEGORYNAME]	[SALES AMOUNT (USD)]	[DISCOUNT AMOUNT (USD)]
2008	Bikes	12968255.42	36167.6592
2005	Bikes	6958251.043	4231.1621
2006	Bikes	18901351.08	178175.8399
2007	Bikes	24256817.5	276065.992
2008	Components	2008052.706	39.9266
2005	Components	574256.9865	0
2006	Components	3428213.05	948.7674
2007	Components	5195315.216	4226.0444
2008	Clothing	366507.844	4151.1235
2005	Clothing	31851.1628	90.9593
2006	Clothing	455730.9729	4233.039
2007	Clothing	815853.2868	12489.3835
2008	Accessories	153299.924	865.5945

DATETIME[CALENDARYEAR]	PRODUCTCATEGORY[PRODUC TCATEGORYNAME]	[SALES AMOUNT (USD)]	[DISCOUNT AMOUNT (USD)]
2005	Accessories	18594.4782	4.293
2006	Accessories	86612.7463	1061.4872
2007	Accessories	275794.8403	4756.6546
2008		15496115.89	41224.3038
2005		7582953.67	4326.4144
2006		22871907.85	184419.1335
2007		30543780.84	297538.0745
		76494758.25	527507.9262

ROLLUPGROUP

ROLLUPGROUP() can be used to calculate groups of subtotals. If used in-place of ROLLUP, ROLLUPGROUP will yield the same result by adding roll-up rows to the result on the groupBy_columnName columns. However, the addition of ROLLUPGROUP() inside a ROLLUP syntax can be used to prevent partial subtotals in roll-up rows.

The following example shows only the grand total of all years and categories without the subtotal of each year with all categories:

SUMMARIZE(ResellerSales_USD

```
    , ROLLUP(ROLLUPGROUP( DateTime[CalendarYear], ProductCategory[ProductCategoryName]))
    , "Sales Amount (USD)", SUM(ResellerSales_USD[SalesAmount_USD])
    , "Discount Amount (USD)", SUM(ResellerSales_USD[DiscountAmount])
)
```

The following table shows a preview of the data as it would be received by any function expecting to receive a table:

DATETIME[CALENDARYEAR]	PRODUCTCATEGORY[PRODUCTCATEGORYNAME]	[SALES AMOUNT (USD)]	[DISCOUNT AMOUNT (USD)]
2008	Bikes	12968255.42	36167.6592
2005	Bikes	6958251.043	4231.1621
2006	Bikes	18901351.08	178175.8399
2007	Bikes	24256817.5	276065.992
2008	Components	2008052.706	39.9266
2005	Components	574256.9865	0
2006	Components	3428213.05	948.7674
2007	Components	5195315.216	4226.0444
2008	Clothing	366507.844	4151.1235
2005	Clothing	31851.1628	90.9593
2006	Clothing	455730.9729	4233.039
2007	Clothing	815853.2868	12489.3835
2008	Accessories	153299.924	865.5945
2005	Accessories	18594.4782	4.293
2006	Accessories	86612.7463	1061.4872
2007	Accessories	275794.8403	4756.6546
		76494758.25	527507.9262

SUMMARIZE with ISSUBTOTAL

Enables the user to create another column, in the Summarize function, that returns True if the row contains sub-total values for the column given as argument to ISSUBTOTAL, otherwise returns False.

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

ISSUBTOTAL parameters

columnName

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

Return value
A **True** value if the row contains a sub-total value for the column given as argument, otherwise returns **False**

Remarks
- ISSUBTOTAL can only be used in the expression part of a SUMMARIZE function.
- ISSUBTOTAL must be preceded by a matching *name* column.

Example
The following sample generates an ISSUBTOTAL() column for each of the ROLLUP() columns in the given SUMMARIZE() function call.

```
SUMMARIZE(ResellerSales_USD
        , ROLLUP( DateTime[CalendarYear], ProductCategory[ProductCategoryName])
        , "Sales Amount (USD)", SUM(ResellerSales_USD[SalesAmount_USD])
        , "Discount Amount (USD)", SUM(ResellerSales_USD[DiscountAmount])
        , "Is Sub Total for DateTimeCalendarYear", ISSUBTOTAL(DateTime[CalendarYear])
        , "Is Sub Total for ProductCategoryName", ISSUBTOTAL(ProductCategory[ProductCategoryName])
)
```

The following table shows a preview of the data as it would be received by any function expecting to receive a table:

[IS SUB TOTAL FOR DATETIMECALENDARYEAR]	[IS SUB TOTAL FOR PRODUCTCATEGORYNAME]	DATETIME[CALENDARYEAR]	PRODUCTCATEGORY[PRODUCTCATEGORYNAME]	[SALES AMOUNT (USD)]	[DISCOUNT AMOUNT (USD)]
FALSE	FALSE				
FALSE	FALSE	2008	Bikes	12968255.42	36167.6592
FALSE	FALSE	2005	Bikes	6958251.043	4231.1621
FALSE	FALSE	2006	Bikes	18901351.08	178175.8399
FALSE	FALSE	2007	Bikes	24256817.5	276065.992
FALSE	FALSE	2008	Components	2008052.706	39.9266
FALSE	FALSE	2005	Components	574256.9865	0
FALSE	FALSE	2006	Components	3428213.05	948.7674
FALSE	FALSE	2007	Components	5195315.216	4226.0444
FALSE	FALSE	2008	Clothing	366507.844	4151.1235
FALSE	FALSE	2005	Clothing	31851.1628	90.9593
FALSE	FALSE	2006	Clothing	455730.9729	4233.039
FALSE	FALSE	2007	Clothing	815853.2868	12489.3835
FALSE	FALSE	2008	Accessories	153299.924	865.5945

FALSE	FALSE	2005	Accessories	18594.4782	4.293
FALSE	FALSE	2006	Accessories	86612.7463	1061.4872
FALSE	FALSE	2007	Accessories	275794.8403	4756.6546
FALSE	TRUE				
FALSE	TRUE	2008		15496115.89	41224.3038
FALSE	TRUE	2005		7582953.67	4326.4144
FALSE	TRUE	2006		22871907.85	184419.1335
FALSE	TRUE	2007		30543780.84	297538.0745
TRUE	TRUE			76494758.25	527507.9262

Class room examples:

Scenario1: Based on courseID display sum and average of discount fee

Course wise total and average = summarize(FactPayments,FactPayments[CourseID],"Total",sum(FactPayments[Discount_Fee]),"Average",AVERAGE(FactPayments[Discount_Fee]))

Scenario2: Based on country and date display sum and average of discount fee

Sales by country and date = GENERATE(SUMMARIZE(SalesGeography,SalesGeography[CountryID]),SUMMARIZE(Sales,Dates[DateID],"Sales by country and date", SUM(Sales[Sales])))

T.DIST

Returns the Student's left-tailed t-distribution.

Syntax: **T.DIST(X,Deg_freedom,Cumulative)**

Return value
The Student's left-tailed t-distribution.
Example

```
EVALUATE { T.DIST(60, 1, TRUE) }
```

[VALUE]
0.994695326367377

T.DIST.2T

Returns the two-tailed Student's t-distribution.

Syntax: T.DIST.2T(X,Deg_freedom)

Return value
The two-tailed Student's t-distribution.
Example

```
EVALUATE { T.DIST.2T(1.959999998, 60) }
```

[VALUE]
0.054644929975921

T.DIST.RT

Returns the right-tailed Student's t-distribution.

Syntax: **T.DIST.RT(X,Deg_freedom)**

Return Value:

The right-tailed Student's t-distribution.

Example : **EVALUATE { T.DIST.RT(1.959999998, 60) }**

Returns:

0.0273224649879605

T.INV

Returns the left-tailed inverse of the Student's t-distribution.

Syntax

T.INV(Probability,Deg_freedom)

Parameters

TERM	DEFINITION
Probability	The probability associated with the Student's t-distribution.
Deg_freedom	The number of degrees of freedom with which to characterize the distribution.

Return value

The left-tailed inverse of the Student's t-distribution.

Example

EVALUATE { T.INV(0.75, 2) }

Returns

[VALUE]
0.816496580927726

T.INV.2T

Returns the two-tailed inverse of the Student's t-distribution.

Syntax

T.INV.2T(Probability,Deg_freedom)

Parameters

TERM	DEFINITION
Probability	The probability associated with the Student's t-distribution.

Deg_freedom	The number of degrees of freedom with which to characterize the distribution.

Return value

The two-tailed inverse of the Student's t-distribution.

Example

```
EVALUATE { T.INV.2T(0.546449, 60) }
```

Returns

[VALUE]
0.606533075825759

TAN

Returns the tangent of the given angle.

Syntax

TAN(NUMBER)

Return value

Parameters

TERM	DEFINITION
number	Required. The angle in radians for which you want the tangent.

Remarks

If your argument is in degrees, multiply it by PI()/180 or use the RADIANS function to convert it to radians.

Example

FORMULA	DESCRIPTION	RESULT
=TAN(0.785)	Tangent of 0.785 radians (0.99920)	0.99920
=TAN(45*PI()/180)	Tangent of 45 degrees (1)	1
=TAN(RADIANS(45))	Tangent of 45 degrees (1)	1

TANH

Returns the hyperbolic tangent of a number.

Syntax

TANH(NUMBER)

Parameters

TERM	DEFINITION
number	Required. Any real number.

Return value

Returns the hyperbolic tangent of a number.

Remarks

The formula for the hyperbolic tangent is:

$$TANH(z) = \frac{SINH(z)}{COSH(z)}$$

Example

=TANH(-2)

=TANH(0)

=TANH(0.5)

When do we go for TOP N? Difference between Sample and TOPN

TOPN

Returns the top N rows of the specified table.

Syntax

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

Example
The following sample creates a measure with the sales of the top 10 sold products.

```
=SUMX(TOPN(10, SUMMARIZE(Product, [ProductKey], "TotalSales",
SUMX(RELATED(InternetSales_USD[SalesAmount_USD]),
InternetSales_USD[SalesAmount_USD]) +
SUMX(RELATED(ResellerSales_USD[SalesAmount_USD]),
ResellerSales_USD[SalesAmount_USD]))
```

Display Top 4 CourseIDs

```
Top 4 CourseIDs =
TOPN(4,summarize(FactPayments,FactPayments[CourseID],"Total",sum(FactPayment
s[Discount_Fee]),"Average",AVERAGE(FactPayments[Discount_Fee])),
[Total],DESC)
```

VAR.S

Returns the variance of a sample population

Syntax

VAR.S(<COLUMNNAME>)

Example

The following example shows the formula for a measure that calculates the variance of the SalesAmount_USD column from the InternetSales_USD for a sample population.

=VAR.S(InternetSales_USD[SalesAmount_USD])

VAR.P

Returns the variance of the entire population.

Syntax

VAR.P(<columnName>)

Example

The following example shows the formula for a measure that estimates the variance of the SalesAmount_USD column from the InternetSales_USD table, for the entire population.

=VAR.P(InternetSales_USD[SalesAmount_USD])

VARX.S

Returns the variance of a sample population.

Syntax

VARX.S(<table>, <expression>)

Example

The following example shows the formula for a calculated column that estimates the variance of the unit price per product for a sample population, when the formula is used in the Product table.

=VARX.S(InternetSales_USD, InternetSales_USD[UnitPrice_USD] – (InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity]))

VARX.P

Returns the variance of the entire population.

Syntax

VARX.P(<table>, <expression>)

Example

The following example shows the formula for a calculated column that calculates the variance of the unit price per product, when the formula is used in the Product table

=VARX.P(InternetSales_USD, InternetSales_USD[UnitPrice_USD] – (InternetSales_USD[DiscountAmount_USD]/InternetSales_USD[OrderQuantity]))

XIRR

Returns the internal rate of return for a schedule of cash flows that is not necessarily periodic.

Syntax

XIRR(<table>, <values>, <dates>, [guess])

Example

The following calculates the internal rate of return of the CashFlows table:

Rate of return := XIRR(CashFlows, [Payment], [Date])

DATE	PAYMENT
1/1/2014	-10000
3/1/2014	2750
10/30/2014	4250
2/15/2015	3250
4/1/2015	2750

Rate of return = 37.49%

XNPV

Returns the present value for a schedule of cash flows that is not necessarily periodic.

Syntax

XNPV(<table>, <values>, <dates>, <rate>)

Example

The following calculates the present value of the CashFlows table:

Present value := XNPV(CashFlows, [Payment], [Date], 0.09)

DATE	PAYMENT
1/1/2014	-10000
3/1/2014	2750
10/30/2014	4250
2/15/2015	3250
4/1/2015	2750

Present value = 2086.65

TEXT FUNCTIONS OVERVIEW

When do we go for Text Functions?

Data Analysis Expressions (DAX) includes a set of text functions that is based on the library of string functions in Excel, but which has been modified to work with tables and columns. This section lists all text functions available in the DAX language.

BLANK **CODE** **CONCATENATE** **FORMAT** ✓ Pre-Defined Numeric Formats for the FORMAT function ✓ Custom Numeric Formats for the FORMAT function ✓ Pre-defined date and time formats for the FORMAT function ✓ Custom date and time formats for the FORMAT function	**LEFT** **LEN** **LOWER** **MID** **REPLACE** **FIND** **FIXED** **CONCATENATEX** **EXACT**	**REPT** **RIGHT** **SEARCH** **SUBSTITUTE** **TRIM** **UNICHAR** **UPPER** **VALUE**

BLANK

Returns a blank.

Syntax

BLANK()

Example

The following example illustrates how you can work with blanks in formulas. The formula calculates the ratio of sales between the Resellers and the Internet channels. However, before attempting to calculate the ratio the denominator should be checked for zero values. If the denominator is zero then a blank value should be returned; otherwise, the ratio is calculated.

```
=IF( SUM(InternetSales_USD[SalesAmount_USD])= 0       , BLANK()   ,
SUM(ResellerSales_USD[SalesAmount_USD])/SUM(InternetSales_USD[SalesAmount_USD]) )
```

CODE

Returns a numeric code for the first character in a text string. The returned code corresponds to the character set used by your computer.

OPERATING ENVIRONMENT	CHARACTER SET
Macintosh	Macintosh character set
Windows	ANSI

Syntax

CODE(text)

Parameters

TERM	DEFINITION
text	The text for which you want the code of the first character.

Return value

A numeric code for the first character in a text string.

Example

FORMULA	DESCRIPTION	RESULT
=CODE("A")	Displays the numeric code for A	65
=CODE("!")	Displays the numeric code for !	33

COMBINEVALUES

Difference between CONCATENATE and COMBINEVALUES?

The COMBINEVALUES function joins two or more text strings into one text string. The primary purpose of this function is to support multi-column relationships in DirectQuery models, see **Remarks** for details.

Syntax

COMBINEVALUES(<delimiter>, <expression>, <expression>[, <expression>]…)

Example

The following DAX query:


```
EVALUATE DISTINCT(SELECTCOLUMNS(DimDate, "Month", COMBINEVALUES(",", [MonthName], [CalendarYear])))
```

CONCATENATE

Difference between CONCATENATE and CONCATENATEX?

Joins two text strings into one text string.

Syntax

CONCATENATE(<text1>, <text2>)

Example: Concatenation of Literals

Description

The sample formula creates a new string value by combining two string values that you provide as arguments.

Code

```
=CONCATENATE("Hello ", "World")
```

Example: Concatenation of Strings in Columns

DescriptionThe sample formula returns the customer's full name as listed in a phone book. Note how a nested function is used as the second argument. This is one way to concatenate multiple strings, when you have more than two values that you want to use as arguments.

Code

```
=CONCATENATE(Customer[LastName], CONCATENATE(", ", Customer[FirstName]))
```

Example: Conditional Concatenation of Strings in Columns

Description

The sample formula creates a new calculated column in the Customer table with the full customer name as a combination of first name, middle initial, and last name. If there is no middle name, the last name comes

directly after the first name. If there is a middle name, only the first letter of the middle name is used and the initial letter is followed by a period.

Code

```
=CONCATENATE( [FirstName]&" ", CONCATENATE( IF( LEN([MiddleName])>1,
LEFT([MiddleName],1)&" ", ""), [LastName]))
```

Comments

This formula uses nested CONCATENATE and IF functions, together with the ampersand (&) operator, to conditionally concatenate three string values and add spaces as separators.

Example: Concatenation of Columns with Different Data Types

The following example demonstrates how to concatenate values in columns that have different data types. If the value that you are concatenating is numeric, the value will be implicitly converted to text. If both values are numeric, both values will be cast to text and concatenated as if they were strings.

PRODUCT DESCRIPTION	PRODUCT ABBREVIATION (COLUMN 1 OF COMPOSITE KEY)	PRODUCT NUMBER (COLUMN 2 OF COMPOSITE KEY)	NEW GENERATED KEY COLUMN
Mountain bike	MTN	40	MTN40
Mountain bike	MTN	42	MTN42

Code

=CONCATENATE('Products'[Product abbreviation],'Products'[Product number])

Comments

The CONCATENATE function in DAX accepts only two arguments, whereas the Excel CONCATENATE function accepts up to 255 arguments. If you need to add more arguments, you can use the ampersand (&) operator. For example, the following formula produces the results, MTN-40 and MTN-42.

=[Product abbreviation] & "-" & [Product number]CAT

Concatenates the result of an expression evaluated for each row in a table.

CONCATENATEX

Syntax

CONCATENATEX(<table>, <expression>, [delimiter])

Example

Employees table

FIRSTNAME	LASTNAME
Alan	Brewer
Michael	Blythe

CONCATENATEX(Employees, [FirstName] & " " & [LastName], ",")

Returns "Alan Brewer, Michael Blythe"

EXACT

Compares two text strings and returns TRUE if they are exactly the same, FALSE otherwise. EXACT is case-sensitive but ignores formatting differences. You can use EXACT to test text being entered into a document.

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. (Boolean)

Example

The following formula checks the value of Column1 for the current row against the value of Column2 for the current row, and returns TRUE if they are the same, and returns FALSE if they are different.

=EXACT([Column1],[Column2])

Difference between FIND and SEARCH?

FIND

Returns the starting position of one text string within another text string. FIND is case-sensitive.

Syntax

FIND(<find_text>, <within_text>[, [<start_num>][, <NotFoundValue>]])

Example

The following formula finds the position of the first letter of the product designation, BMX, in the string that contains the product description.

=FIND("BMX","line of BMX racing goods")

FIXED

Rounds a number to the specified number of decimals and returns the result as text. You can specify that the result be returned with or without commas.

Syntax

FIXED(<number>, <decimals>, <no_commas>)

Example

The following example gets the numeric value for the current row in column, PctCost, and returns it as text with 4 decimal places and no commas.

=FIXED([PctCost],3,1)

Numbers can never have more than 15 significant digits, but decimals can be as large as 127.

FORMAT [FULL PRACTICE REQUIRED ****]

Converts a value to text according to the specified format.

Syntax

FORMAT(<VALUE>,<FORMAT STRING>)

Pre-Defined Numeric Formats for the FORMAT

The following table identifies the predefined numeric format names. These may be used by name as the style argument for the Format function.

Example

The following samples show the usage of different predefined formatting strings to format a numeric value.

FORMAT(12345.67, "General Number")

FORMAT(12345.67, "Currency")

FORMAT(12345.67, "Fixed")

FORMAT(12345.67, "Standard")

FORMAT(12345.67, "Percent")

FORMAT(12345.67, "Scientific")

Pre-defined date and time formats for the FORMAT function

The following table identifies the predefined date and time format names. If you use strings other than these predefined strings, they will be interpreted as a custom date and time format.

FORMAT SPECIFICATION	DESCRIPTION
`"General Date"`	Displays a date and/or time. For example, 3/12/2008 11:07:31 AM. Date display is determined by your application's current culture value.
`"Long Date"` or `"Medium Date"`	Displays a date according to your current culture's long date format. For example, Wednesday, March 12, 2008.
`"Short Date"`	Displays a date using your current culture's short date format. For example, 3/12/2008.
`"Long Time"` or	Displays a time using your current culture's long time format; typically includes hours, minutes, seconds. For example, 11:07:31 AM.
`"Medium Time"`	Displays a time in 12 hour format. For example, 11:07 AM.
`"Short Time"`	Displays a time in 24 hour format. For example, 11:07.

Custom date and time formats for the FORMAT

The following table shows characters you can use to create user-defined date/time formats.

FORMAT SPECIFICATION	DESCRIPTION
(:)	Time separator. In some locales, other characters may be used to represent the time separator. The time separator separates hours, minutes, and seconds when time values are formatted. The actual character that is used as the time separator in formatted output is determined by your application's current culture value.
(/)	Date separator. In some locales, other characters may be used to represent the date separator. The date separator separates the day, month, and year when date values are formatted. The actual character that is used as the date separator in formatted output is determined by your application's current culture.
(%)	Used to indicate that the following character should be read as a single-letter format without regard to any trailing letters. Also used to indicate that a single-letter format is read as a user-defined format. See what follows for additional details.
d	Displays the day as a number without a leading zero (for example, 1). Use %d if this is the only character in your user-defined numeric format.
dd	Displays the day as a number with a leading zero (for example, 01).
ddd	Displays the day as an abbreviation (for example, Sun).
dddd	Displays the day as a full name (for example, Sunday).
M	Displays the month as a number without a leading zero (for example, January is represented as 1). Use %M if this is the only character in your user-defined numeric format.
MM	Displays the month as a number with a leading zero (for

	example, 01/12/01).
MMM	Displays the month as an abbreviation (for example, Jan).
MMMM	Displays the month as a full month name (for example, January).

gg	Displays the period/era string (for example, A.D.).

FORMAT SPECIFICATION	DESCRIPTION
h	Displays the hour as a number without leading zeros using the 12-hour clock (for example, 1:15:15 PM). Use %h if this is the only character in your user-defined numeric format.
hh	Displays the hour as a number with leading zeros using the 12-hour clock (for example, 01:15:15 PM).
H	Displays the hour as a number without leading zeros using the 24-hour clock (for example, 1:15:15). Use %H if this is the only character in your user-defined numeric format.
HH	Displays the hour as a number with leading zeros using the 24-hour clock (for example, 01:15:15).
m	Displays the minute as a number without leading zeros (for example, 12:1:15). Use %m if this is the only character in your user-defined numeric format.
mm	Displays the minute as a number with leading zeros (for example, 12:01:15).
s	Displays the second as a number without leading zeros (for example, 12:15:5). Use %s if this is the only character in your user-defined numeric format.
ss	Displays the second as a number with leading zeros (for example, 12:15:05).
AM/PM	Use the 12-hour clock and display an uppercase AM with any hour before noon; display an uppercase PM with any hour between noon and 11:59 P.M.
am/pm	Use the 12-hour clock and display a lowercase AM with any hour before noon; display a lowercase PM with any hour between noon and 11:59 P.M.

A/P	Use the 12-hour clock and display an uppercase A with any hour before noon; display an uppercase P with any hour between noon and 11:59 P.M.
a/p	Use the 12-hour clock and display a lowercase A with any hour before noon; display a lowercase P with any hour between noon and 11:59 P.M.
AMPM	Use the 12-hour clock and display the AM string literal as defined by your system with any hour before noon; display the PM string literal as defined by your system with any hour between noon and 11:59 P.M. AMPM can be either uppercase or lowercase, but the case of the string displayed matches the string as defined by your system settings. The default format is AM/PM.
y	Displays the year number (0-9) without leading zeros. Use %y if this is the only character in your user-defined numeric format.

FORMAT SPECIFICATION	DESCRIPTION
yy	Displays the year in two-digit numeric format with a leading zero, if applicable.
yyy	Displays the year in four-digit numeric format.
yyyy	Displays the year in four-digit numeric format.
z	Displays the timezone offset without a leading zero (for example, -8). Use %z if this is the only character in your user-defined numeric format.
zz	Displays the timezone offset with a leading zero (for example, -08)
zzz	Displays the full timezone offset (for example, -08:00)

```
DimDate=
ADDCOLUMNS (
CALENDAR (DATE(2000,1,1), DATE(2025,12,31)),
"DateAsInteger", FORMAT ( [Date], "YYYYMMDD" ),
"Year", YEAR ( [Date] ),
"Monthnumber", FORMAT ( [Date], "MM" ),
"YearMonthnumber", FORMAT ( [Date], "YYYY/MM" ),
"YearMonthShort", FORMAT ( [Date], "YYYY/mmm" ),
"MonthNameShort", FORMAT ( [Date], "mmm" ),
"MonthNameLong", FORMAT ( [Date], "mmmm" ),
"DayOfWeekNumber", WEEKDAY ( [Date] ),
"DayOfWeek", FORMAT ( [Date], "dddd" ),
"DayOfWeekShort", FORMAT ( [Date], "ddd" ),
"Quarter", "Q" & FORMAT ( [Date], "Q" ),
"YearQuarter", FORMAT ( [Date], "YYYY" ) & "/Q" & FORMAT ( [Date], "Q" )
```


LEFT

Returns the specified number of characters from the start of a text string.

Syntax

LEFT(<text>, <num_chars>)

Example

The following example returns the first five characters of the company name in the column [ResellerName] and the first five letters of the geographical code in the column [GeographyKey] and concatenates them, to create an identifier.

=CONCATENATE(LEFT('Reseller'[ResellerName],LEFT(GeographyKey,3))

If the **num_chars** argument is a number that is larger than the number of characters available, the function returns the maximum characters available and does not raise an error. For example, the column [GeographyKey] contains numbers such as 1, 12 and 311; therefore the result also has variable length.

LEN

Returns the number of characters in a text string.

Syntax

LEN(<text>)

Example

The following formula sums the lengths of addresses in the columns, [AddressLine1] and [AddressLine2].

=LEN([AddressLine1])+LEN([AddressLin2])

LOWER

Converts all letters in a text string to lowercase.

Syntax

LOWER(<TEXT>)

Return value

Text in lowercase.

TERM	DEFINITION
text	The text you want to convert to lowercase, or a reference to a column that contains text.

Remarks

Characters that are not letters are not changed. For example, the formula =LOWER("123ABC") returns **123abc**.

Example

The following formula gets each row in the column, [ProductCode], and converts the value to all lowercase.

Numbers in the column are not affected.

=LOWER('New Products'[ProductCode])

MID

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>)

Example

The following examples return the same results, the first 5 letters of the column, [ResellerName]. The first example uses the fully qualified name of the column and specifies the starting point; the second example omits the table name and the parameter, **num_chars**.

=MID('Reseller'[ResellerName],5,1))

=MID([ResellerName,5])

The results are the same if you use the following formula:

=LEFT([ResellerName],5)

REPLACE

REPLACE 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>)

Example

The following formula creates a new calculated column that replaces the first two characters of the product code in column, [ProductCode], with a new two-letter code, OB.

=REPLACE('New Products'[Product Code],1,2,"OB")

REPT [REPITITION]

Repeats text a given number of times. Use REPT to fill a cell with a number of instances of a text string.

Syntax

REPT(<text>, <num_times>)

Remarks

If **number_times** is 0 (zero), REPT returns a blank.
If **number_times** is not an integer, it is truncated.
The result of the REPT function cannot be longer than 32,767 characters, or REPT returns an error.
Example: Repeating Literal Strings

Description
The following example returns the string, 85, repeated three times.

Code

=REPT("85",3)

Example: Repeating Column Values
Description
The following example returns the string in the column, [MyText], repeated for the number of times in the column, [MyNumber]. Because the formula extends for the entire column, the resulting string depends on the text and number value in each row.

Code

=REPT([MyText],[MyNumber])

Comments

MYTEX T	MYNUMBER	CALCULATEDCOLUMN1
Text	2	TextText
Number	0	
85	3	858585

RIGHT

RIGHT returns the last character or characters in a text string, based on the number of characters you specify.

Syntax

RIGHT(<text>, <num_chars>)

Example: Returning a Fixed Number of Characters

Description
The following formula returns the last two digits of the product code in the New Products table.

Code
=RIGHT('New Products'[ProductCode],2)
Example: Using a Column Reference to Specify Character Count

Description

The following formula returns a variable number of digits from the product code in the New Products table, depending on the number in the column, MyCount. If there is no value in the column, MyCount, or the value is a blank, RIGHT also returns a blank.

Code

```
=RIGHT('New Products'[ProductCode],[MyCount])
```

Differences between Find and Search?

SEARCH

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>]])
```

Example: Search within a String

Description

The following formula finds the position of the letter "n" in the word "printer".

Code

```
=SEARCH("n","printer")
```

Comments

The formula returns 4 because "n" is the fourth character in the word "printer."

Example: Search within a Column

Description

You can use a column reference as an argument to SEARCH. The following formula finds the position of the character "-" (hyphen) in the column, [PostalCode].

Code

```
=SEARCH("-",[PostalCode])
```

Comments

The return result is a column of numbers, indicating the index position of the hyphen.

Example: Error-Handling with SEARCH

Description

The formula in the preceding example will fail if the search string is not found in every row of the source column. Therefore, the next example demonstrates how to use IFERROR with the SEARCH function, to ensure that a valid result is returned for every row.

The following formula finds the position of the character "-" within the column, and returns -1 if the string is not found.

Code

```
= IFERROR(SEARCH("-",[PostalCode]),-1)
```

Comments

Note that the data type of the value that you use as an error output must match the data type of the non-error output type. In this case, you provide a numeric value to be output in case of an error because SEARCH returns an integer value.
However, you could also return a blank (empty string) by using BLANK() as the second argument to IFERROR.

SUBSTITUTE

Difference between REPLACE and SUBSTITUTE?

Replaces existing text with new text in a text string.

Syntax

SUBSTITUTE(<text>, <old_text>, <new_text>, <instance_num>)

Example: Substitution within a String

Description

The following formula creates a copy of the column [Product Code] that substitutes the new product code **NW** for the old product code **PA** wherever it occurs in the column.

Code

=SUBSTITUTE([Product Code], "NW", "PA")

TRIM

Removes all spaces from text except for single spaces between words.

Syntax

TRIM(<text>)

Example

The following formula creates a new string that does not have trailing white space.

=TRIM("A column with trailing spaces.")

When you create the formula, the formula is propagated through the row just as you typed it, so that you see the original string in each formula and the results are not apparent. However, when the formula is evaluated the string is trimmed.

You can verify that the formula produces the correct result by checking the length of the calculated column created by the previous formula, as follows:

=LEN([Calculated Column 1])

UNICHAR

Returns the Unicode character referenced by the numeric value.

Syntax

UNICHAR(number)

Example

The following example returns the character represented by the Unicode number 66 (uppercase A).

=UNICHAR(65)

The following example returns the character represented by the Unicode number 32 (space character).

=UNICHAR(32)

The following example returns the character represented by the Unicode number 9733 (★ character).

=UNICHAR(9733)

UPPER

Converts a text string to all uppercase letters

Syntax

UPPER(<TEXT>)

Example

The following formula converts the string in the column, [ProductCode], to all uppercase. Non-alphabetic characters are not affected.

=UPPER(['New Products'[Product Code])

Differences between values and value?

VALUE

Converts a text string that represents a number to a number.

Syntax

VALUE(<text>)

Example

The following formula converts the typed string, "3", into the numeric value 3.

=VALUE("3")