Filter Functions (DAX)

What kind of operations Filter functions perform?

ADDMISSINGITEMS	EARLIEST	RELATED
ALL	FILTER	RELATEDTABLE
ALLEXCEPT	FILTERS	SELECTEDVALUE
CALCULATE	HASONEFILTER	SUBSTITUTEWITHINDEX
CALCULATETABLE	HASONEVALUE	USERELATIONSHIP
CROSSFILTER	ISCROSSFILTERED	VALUES
DISTINCT	ISFILTERED	
EARLIER	KEEPFILTERS	

AddmissingItems

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

Syntax

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

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

Example

Add blank rows for columns with "show items with no data" turned on. The ADDMISSINGITEMS function will return NULLs/BLANKs for the IsSubtotal columns of blank rows it adds.

```
VAR 'RowHeadersShowAll' =
CALCULATETABLE
(
ADDMISSINGITEMS
(
[Sales Territory Country],
[Sales Territory Region],
'RowHeadersInCrossTab',
ROLLUPISSUBTOTAL
(
[Sales Territory Group],
[Subtotal for Sales Territory Group],
[Sales Territory Country],
[Subtotal for Sales Territory Country],
[Sales Territory Region],
[Subtotal for Sales Territory Region]
),
'RowHeaders'
),
'DateFilter','TerritoryFilter'
)
```

Example with ROLLUPGROUP

```
VAR 'RowHeadersShowAll' =
CALCULATETABLE
(
ADDMISSINGITEMS
(
[Sales Territory Country],
[Sales Territory Region],
'RowHeadersInCrossTab',
ROLLUPISSUBTOTAL
(
ROLLUPGROUP
(
[Sales Territory Group],
[Sales Territory Country]
),
[Subtotal for Sales Territory Country],
[Sales Territory Region],
[Subtotal for Sales Territory Region]
),
'RowHeaders'
)
```

ALL

Q: When do we go for ALL?

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>[,…]]]})

=SUMX(ResellerSales_USD, ResellerSales_USD[SalesAmount_USD])/SUMX(ALL(ResellerSales_USD), ResellerSales_USD[SalesAmount_USD])

Sales total = SUMX(sales, sales[salesincome])/SUMX(ALL(sales), sales[salesincome])

Sales total1 = SUMX(all(sales), sales[salesincome])

Sales total2 = SUMX(sales, sales[salesincome])

Scenario:

Take an year slicer and see the result, when year changes Discount total 1 changes, where as Discount total 2 does not change.

Take two cards and show the result.

```
Discount total 2 = sumx(all(FactPayments),FactPayments[Discount_Fee])
Discount total 1 = SUMX(FactPayments,FactPayments[Discount_Fee])
```

ALLEXCEPT

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

Syntax

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

=CALCULATE(SUM(ResellerSales_USD[SalesAmount_USD]), ALLEXCEPT(DateTime, DateTime[CalendarYear]))

Scenario:
Take year filter,change year, it does not change the values.Where as Monthname if you change, it will affect.

```
Discount total 2 =
calculate(sum(FactPayments[Discount_Fee]),allexcept(DimDate,DimDate[Monthname]))
```

AllNoBlankRow

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

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

Remarks
The ALLNOBLANKROW function only filters the blank row that a parent table, in a relationship, will show when there are one or more rows in the child table that have non-matching values to the parent column. See the example below for a thorough explanation.

The following table summarizes the variations of ALL that are provided in DAX, and their differences:

FUNCTION AND USAGE	DESCRIPTION
ALL(Column)	Removes all filters from the specified column in the table; all other filters in the table, over other columns, still apply.
ALL(Table)	Removes all filters from the specified table.
ALLEXCEPT(Table,Col1,Col2...)	Overrides all context filters in the table except over the specified columns.
ALLNOBLANK(table|column)	From the parent table of a relationship, returns all rows but the blank row, or all distinct values of a column but the blank row, and disregards any context filters that might exist

AllSelected

The 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>])

Q:Differences between calculate and calculate table?

Calculate

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

CALCULATE(<expression>,<filter1>,<filter2>…)
=(SUM('ResellerSales_USD'[SalesAmount_USD]))
/CALCULATE(SUM('ResellerSales_USD'[SalesAmount_USD])
,ALL('ResellerSales_USD'))

```
calculate(sum(FactPayments[Discount_Fee]),allexcept(DimDate,DimDate[Monthname]))
```

CalculateTable

Whereas the CALCULATE function requires as its first argument an expression that returns a single value, the CALCULATETABLE function takes a table of values. Evaluates a table expression in a context modified by the given filters.

Q: When do we go for Calcualte table?

Syntax

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

=SUMX(CALCULATETABLE('InternetSales_USD', 'DateTime'[CalendarYear]=2006)

, [SalesAmount_USD])

Calculate: Perform calculations

Calculate table: Returns table of values [so that we can apply operations on that data]

```
Scenario: Display 2017 year data
   New Table: Caltable 2017 rows= CALCULATETABLE(FactPayments,DimDate[Year]=2017)
```

Scenario: Total Discount fee in 2017 using calculate table

New Measure:Calcuatetable_2017_total = sumx(CALCULATETABLE(FactPayments,DimDate[Year]=2017),FactPayments[Discount_Fee])

CrossFilter

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

Syntax

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

Remarks

- In the case of a 1:1 relationship, there is no difference between the one and both direction. CROSSFILTER can only be used in functions that take a filter as an argument, for example: CALCULATE, CALCULATETABLE, CLOSINGBALANCEMONTH, CLOSINGBALANCEQUARTER, CLOSINGBALANCEYEAR, OPENINGBALANCEMONTH, OPENINGBALANCEQUARTER OPENINGBALANCEYEAR, TOTALMTD, TOTALQTD and TOTALYTD functions.
  CROSSFILTER uses existing relationships in the model, identifying relationships by their ending point columns.

BiDi:= CALCULATE([Distinct Count of ProductKey], CROSSFILTER(FactInternetSales[ProductKey], DimProduct[ProductKey] , Both))

Practical:

Directly you will not get distinct coursesID count in an year due to 1-Many between DimCourse and FactPayments

Test like this:

Distinct count of CourseID =DistinctCount(DimCourse[COURSEID])

Take Year Slicer, change year and see the result. It won't change.

Put Bidirectional relationship between DimCourse and FactPayments and see the result.

CountRows

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.

NOTE

This function cannot be used to Return values into a cell or column on a worksheet; rather, you nest the DISTINCT functionwithin a formula, to get a list of distinct values that can be passed to another function and then counted, summed, or used for other operations.

Syntax

Q: Difference beteen countrows and distinct?

COUNTROWS(<column>)

Return value

A column of unique values.

Example

The following formula counts the number of unique customers who have generated orders over the internet channel. The table that follows illustrates the possible results when the formula is added to a PivotTable.

=COUNTROWS(DISTINCT(InternetSales_USD[CustomerKey]))

DISTINCT

Returns a table by removing duplicate rows from another table or expression.

Syntax

DISTINCT(<table>)

Related functions

There is another version of the DISTINCT function, DISTINCT (column), that takes a column name as input parameter.

Example

The following query:

EVALUATE DISTINCT({ (1, "A"), (2, "B"), (1, "A") })

Returns table

[VALUE1]	[VALUE2]
1	A
2	B

Earlier

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

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.

Syntax

EARLIER(<column>, <number>)

Exceptions

Description of errors

Remarks

EARLIER succeeds if there is a row context prior to the beginning of the table scan. Otherwise it returns an error.
NOTE

In practice, the xVelocity in-memory analytics engine (VertiPaq) performs optimizations to reduce the actual number of calculations, but you should be cautious when creating formulas that involve recursion.

A new calculated column, **SubCategorySalesRanking**, is created by using the following formula.

> = **COUNTROWS(FILTER(ProductSubcategory, EARLIER(ProductSubcategory[TotalSubcategorySales])<ProductSubcategory[TotalSubcategorySales]))+1**

The following steps describe the method of calculation in more detail.

1. The **EARLIER** function gets the value of *TotalSubcategorySales* for the current row in the table. In this case, because the process is starting, it is the first row in the table

2. **EARLIER**([*TotalSubcategorySales*]) evaluates to $156,167.88, the current row in the outer loop.

3. The **FILTER** function now returns a table where all rows have a value of *TotalSubcategorySales* larger than $156,167.88 (which is the current value for **EARLIER**).

4. The **COUNTROWS** function counts the rows of the filtered table and assigns that value to the new calculated column in the current row plus 1. Adding 1 is needed to prevent the top ranked value from become a Blank.

5. The calculated column formula moves to the next row and repeats steps 1 to 4. These steps are repeated until the end of the table is reached.

The **EARLIER** function will always get the value of the column prior to the current table operation. If you need to get a value from the loop before that, set the second argument to 2.

Earliest

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

Syntax

EARLIEST(<column>)

Return value

A column with filters removed.

Differences between earlier and earliest?

Remarks

The EARLIEST function is similar to EARLIER, but lets you specify one additional level of recursion.

Example

The current sample data does not support this scenario.
=EARLIEST(<column>)

Filter

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

Syntax

FILTER(<table>,<filter>)

Return value

A table containing only the filtered rows.

Example

In this example, the expression

FILTER('InternetSales_USD', RELATED('SalesTerritory'[SalesTerritoryCountry])<>"United States") returns a table

that is a subset of Internet Sales minus all rows that belong to the United States sales territory. The RELATED function is what links the Territory key in the Internet Sales table to SalesTerritoryCountry in the SalesTerritory table.

Table 2. Comparing non- U.S. sales by product categories

SUMX(FILTER('InternetSales_USD', RELATED('SalesTerritory'[SalesTerritoryCountry])<>"United States")

,'InternetSales_USD'[SalesAmount_USD])

FILTERS

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

Syntax

FILTERS(<columnName>)

Example

The following example shows how to determine the number of direct filters a column has.
=COUNTROWS(FILTERS(ResellerSales_USD[ProductKey]))
The example above lets you know how many direct filters on ResellerSales_USD[ProductKey] have been applied to the context where the expression is being evaluated.

```
Numberoffilters = COUNTROWS(filters(FactPayments[CourseID]))
```

HasOneFilter

Returns **TRUE** when the number of directly filtered values on *columnName* is one; otherwise returns **FALSE**.

Syntax

HASONEFILTER(<columnName>)

Example
The following example shows how to use HASONEFILTER() to return the filter for ResellerSales_USD[ProductKey]) if there is one filter, or to return BLANK if there are no filters or more than one filter on ResellerSales_USD[ProductKey]).
=IF(HASONEFILTER(ResellerSales_USD[ProductKey]),FILTERS(ResellerSales_USD[ProductKey]),BLANK())

Scenario: If courseID has filter, then it will show you count. Otherwise returns BLANK.

```
DirectFilter =
IF(HASONEFILTER(FactPayments[CourseID]),FILTERS(FactPayments[CourseID]),BLANK())
```

HASONEVALUE

Returns **TRUE** when the context for *columnName* has been filtered down to one distinct value only. Otherwise is **FALSE**.
Syntax
- **HASONEVALUE(<columnName>)**

Example

In the following example, you want to create a formula that verifies if the context is being sliced by one value in order to estimate a percentage against a predefined scenario; in this case you want to compare

Reseller Sales against sales in 2007, then you need to know if the context is filtered by single years. Also, if the comparison is meaningless you want to return BLANK.

Create a measure named [ResellerSales compared to 2007] using the following expression:

=IF(HASONEVALUE(DateTime[CalendarYear]),SUM(ResellerSales_USD[SalesAmount_USD])/CALCULATE(SUM(ResellerSales_US D[SalesAmount_USD]),DateTime[CalendarYear]=2007),BLANK())

ISCROSSFILTERED

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

Syntax

ISCROSSFILTERED(<columnName>)

What exactly IS and HAS perform?

ISFILTERED

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 function returns **FALSE**.

Syntax

ISFILTERED(<columnName>)

Remarks

- *columnName* is said to be filtered directly when the filter or filters apply over the column; a column is saidto be cross-filtered when a filter applied to another column in the same table or in a related table affects *columnName* the column by filtering it as well.
- The related function ISCROSSFILTERED function (DAX)returns TRUE when *columnName* or another column in the same or related table is being filtered.

KEEPFILTERS

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

Syntax

KEEPFILTERS(<expression>)

Example

The following example takes you through some common scenarios that demonstrate use of the KEEPFILTERS function as part of a CALCULATE or CALCULATETABLE formula.

The first three expressions obtain simple data to be used for comparisons:

- Internet Sales for the state of Washington.
- Internet Sales for the states of Washington and Oregon (both states combined).
- Internet Sales for the state of Washington and the province of British Columbia (both regions combined).

The fourth expression calculates Internet Sales for Washington and Oregon, while the filter for Washington and British Columbia is applied.
The next expression calculates Internet Sales for Washington and Oregon but uses KEEPFILTERS; the filter for Washington and British Columbia is part of the prior context.

```
EVALUATE ROW(
  "$$ in WA"
    , CALCULATE('Internet Sales'[Internet Total Sales]
                    , 'Geography'[State Province Code]="WA"
          )
    , "$$ in WA and OR"
        , CALCULATE('Internet Sales'[Internet Total Sales]
                    , 'Geography'[State Province Code]="WA"
                      || 'Geography'[State Province Code]="OR"
          )

    , "$$ in WA and BC"
        , CALCULATE('Internet Sales'[Internet Total Sales]
                    , 'Geography'[State Province Code]="WA"
                      || 'Geography'[State Province Code]="BC"
          )

    , "$$ in WA and OR ??"
        , CALCULATE(
              CALCULATE('Internet Sales'[Internet Total Sales]
                     ,'Geography'[State Province Code]="WA"
                         || 'Geography'[State Province Code]="OR"
             )
             , 'Geography'[State Province Code]="WA"
               || 'Geography'[State Province Code]="BC"
          )

    , "$$ in WA !!"
        , CALCULATE(
              CALCULATE('Internet Sales'[Internet Total Sales]
                      , KEEPFILTERS('Geography'[State Province Code]="WA"
                                || 'Geography'[State Province Code]="OR"
                        )
             )
             , 'Geography'[State Province Code]="WA"
               || 'Geography'[State Province Code]="BC"
          )
)
```

When this expression is evaluated against the sample database AdventureWorks DW Tabular 2012, the following results are obtained.

COLUMN	VALUE
[$$ in WA]	$ 2,467,248.34
[$$ in WA and OR]	$ 3,638,239.88
[$$ in WA and BC]	$ 4,422,588.44
[$$ in WA and OR ??]	$ 3,638,239.88
[$$ in WA !!]	$ 2,467,248.34

RELATED

Q: When do we go for Related Table?

Q: What are the limitations it has?

Returns a related value from another table.

Syntax

RELATED (<COLUMN>)

If the example does not work, you might need to create a relationship between the tables.

```
= SUMX(FILTER( 'InternetSales_USD'
              ,   RELATED('SalesTerritory'[SalesTerritoryCountry
                  ]) <>"United States"
                )
        ,'InternetSales_USD'[SalesAmount_USD])
```

Scenario: Display CourseName in the fact table against to each CourseID value

New column in the FactPayment Course-name=Related(DimCourse[CourseName])

Scenario: MSBI Training Institute DiscountFee required from FactPayments

```
DiscountFee based on Institutename =
sumx(
      filter(FactPayments,related(DimInstitute[Institutename])="MSBI Training
         Institute"),
     FactPayments[Discount_Fee]
)
```

Q: Differences between Related and Related Table?

Q: Differences between Related Table and Calculated Table?

RELATEDTABLE

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

Syntax

RELATEDTABLE(<tableName>)

```
= SUMX( RELATEDTABLE('InternetSales_USD')
     , [SalesAmount_USD])
```

```
Relatedtbl = sumx(RELATEDTABLE(FactPayments),FactPayments[Discount_Fee])
```

Take Year slicer and change, the values will change. Whereas in the previous related calculation filters not applicable.

SELECTEDVALUE

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

Syntax

SELECTEDVALUE(<columnName>[, <alternateResult>])

Example

The following DAX query:

DEFINE MEASURE DimProduct[Selected Color] = SELECTEDVALUE(DimProduct[Color], "No Single Selection") EVALUATE SUMMARIZECOLUMNS(ROLLUPADDISSUBTOTAL(DimProduct[Color], "Is Total"), "Selected Color", [Selected Color])

ORDER BY [Is Total] ASC, [Color] ASC

USERELATIONSHIP

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

Syntax

USERELATIONSHIP(<columnName1>,<columnName2>)

- 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, ie. the following sample expression is nested 3 levels deep but only 2 for USEREALTIONSHIP:

=CALCULATE(CALCULATE(CALCULATE(&lt;anyExpression&gt;, USERELATIONSHIP(t1[colA], t2[colB])), t99[colZ]=999), USERELATIONSHIP(t1[colA], t2[colA]))

Example

The following sample shows how to override the default, active, relationship between InternetSales and DateTime tables. The default relationship exists between the OrderDate column, in the InternetSales table, and the Date column, in the DateTime table.

To calculate the sum of internet sales and allow slicing by ShippingDate instead of the traditional OrderDate you need to create a measure, [InternetSales by ShippingDate] using the following expression:

=CALCULATE(SUM(InternetSales[SalesAmount]), USERELATIONSHIP(InternetSales[ShippingDate], DateTime[Date]))

Example: Display sum of discount fee based on the relationship between FactInternet Sales and Date Table

```
Discount_Fee_Relationship = CALCULATE(sum(FactPayments[Discount_Fee]),
USERELATIONSHIP(FactPayments[Date],DimDate[Date]))
```

Here the relationship between Fact table and Date table system uses to perform calculation

VALUES

Syntax

VALUES(<TableNameOrColumnName>)

Example

=COUNTROWS(VALUES('InternetSales_USD'[SalesOrderNumber]))

Scenario: Display only coursenames

```
Values_table = VALUES(DimCourse[Coursename])
```

Scenario: Display the count of coursenames

=COUNTROWS(VALUES(DimCourse[Coursename]))

INFORMATION FUNCTIONS [DAX]

How Information functions are helpful?

DAX information functions look at the cell or row that is provided as an argument and tells you whether the value matches the expected type. For example, the ISERROR function returns TRUE if the value that you reference contains an error.

CONTAINS	ISEVEN	ISNONTEXT	ISTEXT
CUSTOMDATA	ISINSCOPE	ISNUMBER	LOOKUPVALUE
ISBLANK	ISLOGICAL	ISONORAFTER	USERNAME
ISERROR			

CONTAINS

Returns true if values for all referred columns exist, or are contained, in those columns; otherwise, the function returns false.

Syntax

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

Example

Q: What is the equivalent in SQL? -- EXISTS

The following example creates a calculated measure that tells you whether there were any Internet sales of the product 214 and to customer 11185 at the same time.

=CONTAINS(InternetSales, [ProductKey], 214, [CustomerKey], 11185)

Scenario: Return true or false, if courseID=”MSBI-F” and ModeID=”Online” available in fact table

```
Course_Mode_Exists = CONTAINS(FactPayments,FactPayments[CourseID],"MSBI-F",
FactPayments[ModeID],"Online")
```

CUSTOMDATA

Returns the content of the **CustomData** property in the connection string.
Syntax

CUSTOMDATA()
Example

The following DAX code verifies if the CustomData property was set to "OK".
=IF(CUSTOMDATA()="OK", "Correct Custom data in connection string", "No custom data in connection string property or unexpected value")

CONTAINSROW

Returns TRUE if a row of values exists or contained in a table, otherwise returns FALSE. Except syntax, the IN operator and CONTAINSROW function are functionally equivalent.IN Operator

Q: What is the difference between contains and contains row?

Syntax

<scalarExpr> IN <tableExpr>
(<scalarExpr1>, <scalarExpr2>, …) IN <tableExpr>
CONTAINSROW function
CONTAINSROW(<tableExpr>, <scalarExpr>[, <scalarExpr>, …])

Example 1

The following equivalent DAX queries:

EVALUATE FILTER(ALL(DimProduct[Color]), [Color] IN { "Red", "Yellow", "Blue" }) ORDER BY [Color]

and

EVALUATE FILTER(ALL(DimProduct[Color]), ([Color]) IN { "Red", "Yellow", "Blue" })

ORDER BY [Color]

and

EVALUATE FILTER(ALL(DimProduct[Color]), CONTAINSROW({ "Red", "Yellow", "Blue" }, [Color]))

ORDER BY [Color]

Return the following table with a single column:

DIMPRODUCT[COLOR]
Blue
Red
Yellow

Example 2

The following equivalent DAX queries:

EVALUATE FILTER(SUMMARIZE(DimProduct, [Color], [Size]), ([Color], [Size]) IN { ("Black", "L") })

And

EVALUATE FILTER(SUMMARIZE(DimProduct, [Color], [Size]), CONTAINSROW({ ("Black", "L") }, [Color], [Size]))

Return:

DIMPRODUCT[COLOR]	DIMPRODUCT[SIZE]
Black	L

Example 3

The following equivalent DAX queries:

EVALUATE FILTER(ALL(DimProduct[Color]), NOT [Color] IN { "Red", "Yellow", "Blue" })

ORDER BY [Color]

and

EVALUATE FILTER(ALL(DimProduct[Color]), NOT CONTAINSROW({ "Red", "Yellow", "Blue" }, [Color])) ORDER BY [Color]

Return the following table with a single column:

DIMPRODUCT[COLOR]

Black
Grey
Multi
Silver
Silver\Black
White

Class room examples:

Example 1:

```
EVALUATE
(
FILTER(ALL(DimCourseMode[ModeID]), CONTAINSROW({ "Online", "Classroom" }, [ModeID]))
)
```

Example 2:

```
EVALUATE
(
FILTER(SUMMARIZE(FactPayments, [LocID], [ModeID]), CONTAINSROW({ ("HYD", "Online") }, [LocID], [ModeID]))
)
```

ISBLANK

Checks whether a value is blank, and returns TRUE or FALSE.

Syntax
ISBLANK(<value>)

Example
This formula computes the increase or decrease ratio in sales compared to the previous year. The example uses the IF function to check the value for the previous year's sales in order to avoid a divide by zero error.

ROW LABELS	TOTAL SALES	TOTAL SALES PREVIOUS YEAR	SALES TO PREVIOUS YEAR RATIO
2005	$10,209,985.08		
2006	$28,553,348.43	$10,209,985.08	179.66%
2007	$39,248,847.52	$28,553,348.43	37.46%
2008	$24,542,444.68	$39,248,847.52	-37.47%
Grand Total	$102,554,625.71		

```
    //Sales to Previous Year Ratio

    =IF( ISBLANK('CalculatedMeasures'[PreviousYearTotalSales])
       , BLANK()
,  ( 'CalculatedMeasures'[Total Sales]-'CalculatedMeasures'[PreviousYearTotalSales] )
/'CalculatedMeasures'[PreviousYearTotalSales])
```

ISERROR

Checks whether a value is an error, and returns TRUE or FALSE.

Syntax
ISERROR(<value>)
Example
The following example calculates the ratio of total Internet sales to total reseller sales. The ISERROR function is used to check for errors, such as division by zero. If there is an error a blank is returned, otherwise the ratio is returned.

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

```
Is error value = iserror([ClosingBalanceMonthm]/0)

Scenario: If there is an error substistuing with 99999
if(iserror(FactPayments[M1]/0),99999)
```

ISEVEN

Returns TRUE if number is even, or FALSE if number is odd.
Syntax
ISEVEN(number)

ISINSCOPE

Returns true when the specified column is the level in a hierarchy of levels.
Syntax
ISINSCOPE(<columnName>)
Example

```
DEFINE
MEASURE FactInternetSales[% of Parent] =
  SWITCH (TRUE(),
    ISINSCOPE(DimProduct[Subcategory]),
      DIVIDE(
        SUM(FactInternetSales[Sales Amount]),
        CALCULATE(
          SUM(FactInternetSales[Sales Amount]),
          ALLSELECTED(DimProduct[Subcategory]))
      ),

    ISINSCOPE(DimProduct[Category]),
      DIVIDE(
        SUM(FactInternetSales[Sales Amount]),
        CALCULATE(
          SUM(FactInternetSales[Sales Amount]),
          ALLSELECTED(DimProduct[Category]))
      ),
    1
  ) * 100
```

```
    EVALUATE
      SUMMARIZECOLU
      MNS
      (
        ROLLUPADDISSUBTOTAL
        (

          DimProduct[Category], "Category Subtotal",
          DimProduct[Subcategory], "Subcategory
          Subtotal"
        ),

        TREATAS(

          {"Bike Racks", "Bike Stands", "Mountain Bikes", "Road Bikes", "Touring
          Bikes"}, DimProduct[Subcategory]),

        "Sales", SUM(FactInternetSales[Sales
        Amount]), "% of Parent", [% of Parent]
)

      ORDER BY
        [Category Subtotal] DESC, [Category],
        [Subcategory Subtotal] DESC,
        [Subcategory]
```

Returns

DIMPRODUCT[CATEGORY]	DIMPRODUCT[SUBCATEGORY]	[CATEGORY SUBTOTAL]	[SUBCATEGORY SUBTOTAL]	[SALES]	[% OF PARENT]
		TRUE	TRUE	28,397,095.65	100.00
Accessories		FALSE	TRUE	78,951.00	0.28
Accessories	Bike Racks	FALSE	FALSE	39,360.00	49.85
Accessories	Bike Stands	FALSE	FALSE	39,591.00	50.15
Bikes		FALSE	TRUE	28,318,144.65	99.72
Bikes	Mountain Bikes	FALSE	FALSE	9,952,759.56	35.15
Bikes	Road Bikes	FALSE	FALSE	14,520,584.04	51.28
Bikes	Touring Bikes	FALSE	FALSE	3,844,801.05	13.58

ISLOGICAL

Checks whether a value is a logical value, (TRUE or FALSE), and returns TRUE or FALSE.

Syntax

ISLOGICAL(<value>)

Example

The following three samples show the behavior of ISLOGICAL.

```
//RETURNS: Is Boolean type or Logical

=IF(ISLOGICAL(true), "Is Boolean type or Logical", "Is different type")
//RETURNS: Is Boolean type or Logical
```

```
=IF(ISLOGICAL(false), "Is Boolean type or Logical", "Is different type")

//RETURNS: Is different type

=IF(ISLOGICAL(25), "Is Boolean type or Logical", "Is different type")
```

ISNONTEXT

Checks if a value is not text (blank cells are not text), and returns TRUE or FALSE.
Syntax
    ISNONTEXT(<value>)
Example

The following examples show the behavior of the ISNONTEXT function.

```
//RETURNS: Is Non-Text

=IF(ISNONTEXT(1), "Is Non-Text", "Is Text")

//RETURNS: Is Non-Text

=IF(ISNONTEXT(BLANK()), "Is Non-Text", "Is Text")

//RETURNS: Is Text

=IF(ISNONTEXT(""), "Is Non-Text", "Is Text")
```

ISNUMBER

Checks whether a value is a number, and returns TRUE or FALSE.
Syntax
    ISNUMBER(<value>)

```
//RETURNS: Is number

=IF(ISNUMBER(0), "Is number", "Is    Not number")

//RETURNS: Is number

=IF(ISNUMBER(3.1E-1),"Is number",  "Is Not number")

//RETURNS: Is Not number

=IF(ISNUMBER("123"), "Is number",  "Is Not number")
```

ISODD

Returns TRUE if number is odd, or FALSE if number is even.

Syntax
    ISODD(number)

Return value
Returns TRUE if number is odd, or FALSE if number is even.

Remarks
If number is nonnumeric, ISODD returns the #VALUE! error value.
A boolean function that emulates the behavior of a ‘Start At’ clause and returns true for a row that meets all of the condition parameters.

ISONORAFTER

This function 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.

Based on the sort order, the first parameter is compared with the second parameter. If the sort order is ascending, the comparison to be done is first parameter greater than or equal to second parameter. If the sort order is descending, the comparison to be done is second parameter less than or equal to first parameter.

Syntax

ISONORAFTER(<scalar_expression>, <scalar_expression>sort_order] [,scalar_expression>, <scalar_expression>, [sort_order][,…])

Return value

True or false.

Example

Table name: 'Info'

COUNTRY	STATE	COUNT	TOTAL
IND	JK	20	800
IND`	MH	25	1000
IND	WB	10	900
USA	CA	5	500

FILTER(Info, ISONORAFTER(Info[Country], "IND", ASC, Info[State], "MH", ASC))

Scenario: Create a table with CourseID MSBI-F sort oder (Asc) and Coursename MSBI Fast Track sort order (ASC)

TBLNEW = FILTER(DimCourse, ISONORAFTER(DimCourse[CourseID], "MSBI-F", ASC, DimCourse[Coursename], "MSBI Fast Track", ASC))

ISTEXT

Checks if a value is text, and returns TRUE OR FALSE

Syntax

ISTEXT(<VALUE>)

Example

The following examples show the behavior of the ISTEXT function.

```
//RETURNS: Is Text

=IF(ISTEXT("text"), "Is Text", "Is Non-Text")

//RETURNS: Is Text

=IF(ISTEXT(""), "Is Text", "Is Non-Text")

//RETURNS: Is Non-Text

=IF(ISTEXT(1), "Is Text", "Is Non-Text")

//RETURNS: Is Non-Text

=IF(ISTEXT(BLANK()), "Is Text", "Is Non-Text")
```

LOOKUPVALUE

Returns the value in *result_columnName* for the row that meets all criteria specified by *search_columnName* and *search_value*.

Q: Difference related and lookup value?

Syntax

LOOKUPVALUE(<result_columnName>, <search_columnName>, <search_value>[, <search_columnName>, <search_value>]…)

Example

The following example returns the SafetyStocklLevel for the bike model "Mountain-400-W Silver, 46".
=LOOKUPVALUE(Product[SafetyStockLevel], [ProductName], " Mountain-400-W Silver, 46")

Scenario: Return course name when courseID= "MSBI-F"

```
Coursename=  LOOKUPVALUE(DimCourse[Coursename],DimCourse[CourseID],"MSBI-F")
```

USERNAME

Returns the domain name and username from the credentials given to the system at connection time

Syntax

USERNAME()

Q: Difference between Username nd UserPricipalName?

Parameters

Return value

The username from the credentials given to the system at connection time
Example

The following code verifies if the user login is part of the UsersTable.
=IF(CONTAINS(UsersTable,UsersTable[login], USERNAME()), "Allowed", BLANK())

```
Usravailable =
if(contains(DimUsers,DimUsers[Login],USERPRINCIPALNAME()),"Allowed",BLANK())

Usravailable = if(contains(DimUsers,DimUsers[Login],USERNAME()),"Allowed",BLANK())
```

LOGICAL FUNCTIONS [DAX]

Logical functions act upon an expression to return information about the values or sets in the expression. For example, you can use the IF function to check the result of an expression and create conditional results.

AND	**IF**	**NOT**	**SWITCH**
FALSE	**IFERROR**	**OR**	**TRUE**

AND

Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE. Otherwise returns false.

Syntax

AND(<logical1>,<logical2>)

Example
The following formula shows the syntax of the AND function.
=IF(AND(10 > 9, -10 < -1), "All true", "One or more false"

Because both conditions, passed as arguments, to the AND function are true, the formula returns "All True".

Example
The following sample uses the AND function with nested formulas to compare two sets of calculations at the same time. For each product category, the formula determines if the current year sales and previous year sales of the Internet channel are larger than the Reseller channel for the same periods. If both conditions are true, for each category the formula returns the value, "Internet hit".

```
IF( AND( SUM( 'InternetSales_USD'[SalesAmount_USD]) >SUM('ResellerSales_USD'[SalesAmount_USD])

            , CALCULATE(SUM('InternetSales_USD'[SalesAmount_USD]), PREVIOUSYEAR('DateTime'[DateKey]
              ))

  >CALCULATE(SUM('ResellerSales_USD'[SalesAmount_USD]), PREVIOUSYEAR('DateTime'[DateKey] ))

            )

     , "Internet Hit"

     , ""
     )
```

FALSE

Returns the logical value FALSE.
Syntax
FALSE()

Example

The formula returns the logical value FALSE when the value in the Column, 'InternetSales_USD'[SalesAmount_USD], is less than or equal to 200000.

=IF(SUM('InternetSales_USD'[SalesAmount_USD]) >200000, TRUE(), false())

IF

IF

Checks if a condition provided as the first argument is met. 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)

Example

The following example uses nested IF functions that evaluate the number in the column, Calls, from the table FactCallCenter. The function assigns a label as follows: **low** if the number of calls is less than 200, **medium** if the number of calls is less than 300 but not less than 200, and **high** for all other values.

=IF([Calls]<200,"low",IF([Calls]<300,"medium","high"))

Example

=IF([StateProvinceCode]= "CA" && ([MaritalStatus] = "M" || [NumberChildrenAtHome] >1),[City])

Note that parentheses are used to control the order in which the AND (&&) and OR (||) operators are used. Also note that no value has been specified for **value_if_false**. Therefore, the function returns the default, which is an empty string.

IFERROR

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)
Example

The following example returns 9999 if the expression 25/0 evaluates to an error. If the expression returns a value other than error, that value is passed to the invoking expression.
=IFERROR(25/0,9999)

How IN Differences with CONTAINSROW?

IN

Returns True if the scalar value shows up in at least one row of the input relation.
Syntax
IN

Scenario: Find out the discount fee for the courses MSBI-F, MSBI-N, and POWER BI-F

Example
```
RequiredVal = CALCULATE(SUM(FactPayments[Discount_Fee]), DimCourse[CourseID] IN
{"MSBI-F","MSBI-N","POWER BI-F"})
```

NOT

Changes FALSE to TRUE, or TRUE to FALSE.

Syntax
NOT (<LOGICAL>)

Example
The following example retrieves values from the calculated column that was created to illustrate the IF function. For that example, the calculated column was named using the default name, **Calculated Column1**, and contains the following formula: =IF([Orders]<300,"true","false")

The formula checks the value in the column, [Orders], and returns "true" if the number of orders is under 300.
Now create a new calculated column, **Calculated Column2**, and type the following formula.

=NOT([CalculatedColumn1])

For each row in **Calculated Column1**, the values "true" and "false" are interpreted as the logical values TRUE or FALSE, and the NOT function returns the logical opposite of that value.

OR

Checks whether one of the arguments is TRUE to return TRUE. The function returns FALSE if both arguments are FALSE.

Syntax

OR(<logical1>,<logical2>)

Example

The following example shows how to use the OR function to obtain the sales people that belong to the Circle of Excellence. The Circle of Excellence recognizes those who have achieved more than a million dollars in Touring Bikes sales or sales of over two and a half million dollars in 2007.

SALESPERSONF

LAG TRUE

```
IF( OR( CALCULATE(SUM('ResellerSales_USD'[SalesAmount_USD]),
'ProductSubcategory'[ProductSubcategoryName]="Touring Bikes") >
1000000

          ,   CALCULATE(SUM('ResellerSales_USD'[SalesAmount_USD]), 'DateTime'[CalendarYear]=2007) >
              2500000
          )

    , "Circle of Excellence"

    , ""

    )
```

SWITCH

Evaluates an expression against a list of values and returns one of multiple possible result expressions.

Syntax

SWITCH(<expression>, <value>, <result>[, <value>, <result>]…[, <else>])

Q: How many conditionl statements DAX has?

Example

The following example creates a calculated column of month names.

```
=SWITCH([Month], 1, "January", 2, "February", 3, "March", 4, "April"

               , 5, "May", 6, "June", 7, "July", 8, "August"

               , 9, "September", 10, "October", 11, "November", 12, "December"

               , "Unknown month number" )
```

Scenario: Based on course mode provide hike to the courses

Goto FactPayments →New column→ `Discount_Inc =`

```
switch(FactPayments[ModeID],"Online",FactPayments[Discount_Fee]
*10/100,"Classroom",FactPayments[Discount_Fee] * 20/100,"Customized",
FactPayments[Discount_Fee]* 30/100)
```

TRUE

Returns the logical value TRUE.

Syntax

TRUE()

Example

The formula returns the logical value TRUE when the value in the column, 'InternetSales_USD'[SalesAmount_USD], is greater than 200000.

= IF(SUM('InternetSales_USD'[SalesAmount_USD]) >200000, TRUE(), false())