# CALCULATE

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

## Syntax

DAX Copy

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

### Parameters

| **Term** | **Definition** |
| --- | --- |
| expression | The expression to be evaluated. |
| filter1, filter2,… | (optional) A comma separated list of Boolean expression or a table expression that defines a filter. |

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

The following restrictions apply to Boolean expressions that are used as arguments:

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

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

## Return value

The value that is the result of the expression.

Example:

=( SUM('ResellerSales\_USD'[SalesAmount\_USD]))

/CALCULATE( SUM('ResellerSales\_USD'[SalesAmount\_USD])

,ALL('ResellerSales\_USD'))

**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**

DAX Copy

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

**Parameters**

| **Term** | **Definition** |
| --- | --- |
| table | The table that you want to clear filters on. |
| column | The column that you want to clear filters on. |

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

**Return value**

The table or column with filters removed.

Example:

# ALLSELECTED

Removes context filters from columns and rows in the current query, while retaining all other context filters or explicit filters.

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

DAX Copy

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

### Parameters

| **Term** | **Definition** |
| --- | --- |
| tableName | The name of an existing table, using standard DAX syntax. This parameter cannot be an expression. This parameter is optional. |
| columnName | The name of an existing column using standard DAX syntax, usually fully qualified. It cannot be an expression. This parameter is optional. |

## Return value

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

Example:

# ALLEXCEPT

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

## Syntax

DAX Copy

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

### Parameters

| **Term** | **Definition** |
| --- | --- |
| table | The table over which all context filters are removed, except filters on those columns that are specified in subsequent arguments. |
| column | The column for which context filters must be preserved. |

The first argument to the ALLEXCEPT function must be a reference to a base table; all subsequent arguments must be references to base columns. You cannot use table expressions or column expressions with the ALLEXCEPT function.  
Return value  
A table with all filters removed except for the filters on the specified columns.

Example:

=CALCULATE(SUM(ResellerSales\_USD[SalesAmount\_USD]), ALLEXCEPT(DateTime, DateTime[CalendarYear]))

# CALCULATETABLE

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

## Syntax

DAX Copy

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

### Parameters

| **Term** | **Definition** |
| --- | --- |
| Expression\*\* | The table expression to be evaluated |
| filter1, filter2,… | A Boolean expression or a table expression that defines a filter |

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

The following restrictions apply to Boolean expressions that are used as arguments:

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

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

## Return value

A table of values.

Example:

=SUMX( CALCULATETABLE('InternetSales\_USD', 'DateTime'[CalendarYear]=2006)

, [SalesAmount\_USD])

# CROSSFILTER

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

## Syntax

DAX Copy

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

### Parameters

| **Term** | **Definition** |
| --- | --- |
| columnName1 | The name of an existing column, using standard DAX syntax and fully qualified, that usually represents the many side of the relationship to be used; if the arguments are given in reverse order the function will swap them before using them. This argument cannot be an expression. |
| columnName2 | The name of an existing column, using standard DAX syntax and fully qualified, that usually represents the one side or lookup side of the relationship to be used; if the arguments are given in reverse order the function will swap them before using them. This argument cannot be an expression. |
| Direction | The cross-filter direction to be used. Must be one of the following: none No cross-filtering occurs along this relationship  **one** - Filters on the one or lookup side of the side of the relationship filter the many side.  **both** -  Filters on either side filter the other  **none** - No cross-filtering occurs along this relationship |

## Return value

The function returns no value; the function only sets the cross-filtering direction for the indicated relationship, for the duration of the query.

Example:

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

By using the CROSSFILTER function in our measure expression, we get the expected results.

![CROSSFILTER\_Examp\_PivotTable2](media/crossfilter-examp-pivottable2.png "CROSSFILTER\_Examp\_PivotTable2")

# DISTINCT (column)

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 function within 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

DAX Copy

DISTINCT(<column>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| column | The column from which unique values are to be returned. Or, an expression that returns a column. |

## Return value A column of unique values. Example:

=COUNTROWS(DISTINCT(InternetSales\_USD[CustomerKey]))

# DISTINCT (table)

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

## Syntax

DAX Copy

DISTINCT(<table>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| table | The table from which unique rows are to be returned. The table can also be an expression that results in a table. |

## Return value

A table containing only distinct rows.

Example:

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

# DISTINCTCOUNT

Counts the number of distinct values in a column.

## Syntax

DAXCopy

DISTINCTCOUNT(<column>)

### Parameters

| **Term** | **Description** |
| --- | --- |
| column | The column that contains the values to be counted |

## Return value

The number of distinct values in column.

## Remarks

The only argument allowed to this function is a column. You can use columns containing any type of data. When the function finds no rows to count, it returns a BLANK, otherwise it returns the count of distinct values.

# 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

DAX Copy

EARLIER(<column>, <number>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| column | A column or expression that resolves to a column. |
| num | (Optional) A positive number to the outer evaluation pass.  The next evaluation level out is represented by 1; two levels out is represented by 2 and so on.  When omitted default value is 1. |

## Property Value/Return value

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

Example:

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

<https://exceleratorbi.com.au/earlier-vs-earliest-dax/>

<https://powerpivotpro.com/2012/03/the-correct-usage-of-earlier/>

<https://www.sqlbi.com/articles/variables-in-dax/>

# EARLIEST

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

## Syntax

DAX Copy

EARLIEST(<column>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| column | A reference to a column. |

## Property Value/Return value

A column with filters removed.

Example:

=EARLIEST(<column>)

<https://exceleratorbi.com.au/earlier-vs-earliest-dax/>

# FILTER

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

## Syntax

DAX Copy

FILTER(<table>,<filter>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| table | The table to be filtered. The table can also be an expression that results in a table. |
| filter | A Boolean expression that is to be evaluated for each row of the table. For example, [Amount] > 0 or [Region] = "France" |

## Return value

A table containing only the filtered rows.

Example:

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

DAX Copy

FILTERS(<columnName>)

### Parameters

| **Term** | **Description** |
| --- | --- |
| columnName | The name of an existing column, using standard DAX syntax. It cannot be an expression. |

## Return value

The values that are directly applied as filters to columnName.

Example:

=COUNTROWS(FILTERS(ResellerSales\_USD[ProductKey]))

# HASONEFILTER

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

## Syntax

DAX Copy

HASONEFILTER(<columnName>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| columnName | The name of an existing column, using standard DAX syntax. It cannot be an expression. |

## Return value

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

Example:

=IF(HASONEFILTER(ResellerSales\_USD[ProductKey]),FILTERS(ResellerSales\_USD[ProductKey]),BLANK())

# HASONEVALUE

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

## Syntax

HTML Copy

HASONEVALUE(<columnName>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| columnName | The name of an existing column, using standard DAX syntax. It cannot be an expression. |

## Return value

**TRUE** when the context for columnName has been filtered down to one distinct value only. Otherwise is **FALSE**.

Example:

=IF(HASONEVALUE(DateTime[CalendarYear]),SUM(ResellerSales\_USD[SalesAmount\_USD])/ CALCULATE(SUM(ResellerSales\_USD[SalesAmount\_USD]),DateTime[CalendarYear]=2007),BLANK())

# ISCROSSFILTERED

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

## Syntax

DAX Copy

ISCROSSFILTERED(<columnName>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| columnName | The name of an existing column, using standard DAX syntax. It cannot be an expression. |

## Return value

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

Example:

# 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

DAX Copy

ISFILTERED(<columnName>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| columnName | The name of an existing column, using standard DAX syntax. It cannot be an expression. |

## Return value

TRUE when columnName is being filtered directly.

Example:

# KEEPFILTERS

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

## Syntax

DAX Copy

KEEPFILTERS(<expression>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| expression | Any expression. |

## Return value

A table of values.

Example:

# RELATED

Returns a related value from another table.

## Syntax

DAX Copy

RELATED(<column>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| column | The column that contains the values you want to retrieve. |

## Return value

A single value that is related to the current row.

Example:

# RELATEDTABLE

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

## Syntax

DAX Copy

RELATEDTABLE(<tableName>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| tableName | The name of an existing table using standard DAX syntax. It cannot be an expression. |

## Return value

A table of values.

= SUMX( RELATEDTABLE('InternetSales\_USD')

, [SalesAmount\_USD])

# REMOVEFILTERS

Clear filters from the specified tables or columns.

## Syntax

DAX Copy

REMOVEFILTERS([<table> | <column>[, <column>[, <column>[,…]]]])

### Parameters

| **Term** | **Definition** |
| --- | --- |
| table | The table that you want to clear filters on. |
| column | The column that you want to clear filters on. |

## Return value

# SELECTEDVALUE

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

## Syntax

DAX Copy

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

### Parameters

| **Term** | **Definition** |
| --- | --- |
| columnName | The name of an existing column, using standard DAX syntax. It cannot be an expression. |
| alternateResult | (Optional) The value returned when the context for columnName has been filtered down to zero or more than one distinct value. When not provided, the default value is BLANK(). |

## Return value

The value when the context for columnName has been filtered down to one distinct value only. Else, alternateResult.

Example:

# USERELATIONSHIP

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

## Syntax

DAX Copy

USERELATIONSHIP(<columnName1>,<columnName2>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| columnName1 | The name of an existing column, using standard DAX syntax and fully qualified, that usually represents the many side of the relationship to be used; if the arguments are given in reverse order the function will swap them before using them. This argument cannot be an expression. |
| columnName2 | The name of an existing column, using standard DAX syntax and fully qualified, that usually represents the one side or lookup side of the relationship to be used; if the arguments are given in reverse order the function will swap them before using them. This argument cannot be an expression. |

## Return value

The function returns no value; the function only enables the indicated relationship for the duration of the calculation.

# VALUES

When the input parameter is a column name, returns a one-column table that contains the distinct values from the specified column. Duplicate values are removed and only unique values are returned. A BLANK value can be added. When the input parameter is a table name, returns the rows from the specified table. Duplicate rows are preserved. A BLANK row can be added.

Note

This function cannot be used to Return values into a cell or column on a worksheet; rather, you use it as an intermediate function, nested in a formula, to get a list of distinct values that can be counted or used to filter or sum other values.

## Syntax

DAX Copy

VALUES(<TableNameOrColumnName>)

### Parameters

| **Term** | **Definition** |
| --- | --- |
| TableName or ColumnName | A column from which unique values are to be returned, or a table from which rows are to be returned. |

## Return value

When the input parameter is a column name, a single column table. When the input parameter is a table name, a table of the same columns is returned.

Example:

=COUNTROWS(VALUES('InternetSales\_USD'[SalesOrderNumber]))