DAX FORMULAS REFERENCE GUIDE

ENTERPRISE DNA

Welcome to Enterprise DNA

As you may already know, I started Enterprise DNA with a mission to empower users to complete amazing work with Power BI.

I know that if learnt and used effectively individuals are going to really create huge value not only for themselves but for the organizations and teams they work in.

I'm seeing it happen time and time again. Power BI is changing the game out there for analysts, and those moving up the education curve are quickly becoming the go to people within their teams and business functions.

Power BI is a seriously powerful analytical tool and I focus much of my energy teaching others in a practical and commercial way to create valuable analytical work. I do this in many ways and through many virtual venues.

Firstly my authority courses and resources at Enterprise DNA Online are where all my structured learning modules are located. There is an immense amount of content and resources available to those interested in becoming real Power BI super users, so certainly check this out if you haven't already.

I also put a lot of content out there on the web around a wide range of topics, tips and techniques to use within Power BI.

These include:

- Enterprise DNA TV (YouTube)
- Enterprise DNA Blog
- Free training at Enterprise DNA Online
- Power BI Community Blogs
- LinkedIn Groups
- Facebook Groups
- and many other online mediums

Certainly utilize these content mediums as much as you can and also look to connect on the social platform of your choosing to get many updates about all things Power BI from me.

I'm glad you've taken the opportunity to download this DAX formula reference guide. I believe this will be a resource that you will likely need as you learn more and more about how to utilize DAX effectively within Power BI.

Look forward to staying connected!

All the best.

Sam McKay, CFA
CEO - Enterprise DNA

Table of Contents

DAX Formula Reference

- **Date and Time Functions (DAX):** These functions in DAX are similar to date and time functions in Microsoft Excel.
- **Time Intelligence Functions (DAX):** These functions help you create calculations that use built-in knowledge about calendars and dates. By using time and date ranges in combination with aggregations or calculations, you can build meaningful comparisons across comparable time periods for sales, inventory, and so on.
- **Filter Functions (DAX):** These functions help you return specific data types, look up values in related tables, and filter by related values. Lookup functions work by using tables and relationships between them. Filtering functions let you manipulate data context to create dynamic calculations.
- **Information Functions (DAX):** These functions look at a table or column provided as an argument to another function and tells you whether the value matches the expected type. For example, the ISERROR function returns TRUE if the value you reference contains an error.
- **Logical Functions (DAX):** These functions return information about values in an expression. For example, the TRUE function lets you know whether an expression that you are evaluating returns a TRUE value.
- **Math and Trig Functions (DAX):** Mathematical functions in DAX are similar to Excel's mathematical and trigonometric functions. However, there are some differences in the numeric data types used by DAX functions.
- **Other Functions (DAX):** These functions perform unique actions that cannot be defined by any of the categories most other functions belong to.
- **Parent and Child Functions (DAX):** These Data Analysis Expressions (DAX) functions help users manage data that is presented as a parent/child hierarchy in their data models.
- **Statistical Functions (DAX):** These functions perform aggregations. In addition to creating sums and averages, or finding minimum and maximum values, in DAX you can also filter a column before aggregating or create aggregations based on related tables.
- **Text Functions (DAX):** With these functions, you can return part of a string, search for text within a string, or concatenate string values. Additional functions are for controlling the forms for dates, times, and numbers.

1.

DATE AND TIME FUNCTIONS (DAX)

CALENDAR Function (DAX)

= CALENDAR(<start_date>,<end_date>)

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

Example:
The following formula returns a table with dates between January 1, 2005 and December 31, 2015.

=CALENDAR (DATE (2005, 1, 1), DATE (2015, 12, 31))

For a data model which includes actual sales data and future sales forecasts. The following expression returns the date table covering the range of dates in these two tables.

=CALENDAR (MINX (Sales, [Date]), MAXX (Forecast, [Date]))

CALENDARAUTO Function (DAX)

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

Syntax: **CALENDARAUTO([fiscal_year_end_month])**

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

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

Remarks:
The date range is calculated as follows:

- The earliest date in the model, which is not in a calculated column or calculated table, is taken as the MinDate.
- The latest date in the model, which is not in a calculated column or calculated table, is taken as the MaxDate.

The date range returned is dates between the beginning of the fiscal year associated with MinDate and the end of the fiscal year associated with MaxDate.

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

Example:
In this example, the MinDate and MaxDate in the data model are July 1, 2010 and June 30, 2011.

CALENDARAUTO() will return all dates between January 1, 2010 and December 31, 2001.

CALENDARAUTO(3) will return all dates between March 1, 2010 and February28, 2012.

DATE Function (DAX)

Returns the specified date in datetime format.

Syntax: **DATE(<year>,<month>,<day>)**

DATEDIFF Function (DAX)

Returns the count of interval boundaries crossed between two sides.

Syntax: **DATEDIFF(<start_date>, <end_date>, <interval>)**

Return Value:
The count of interval boundaries crossed between two dates.

Example:
DATEDIFF(MIN(Calendar[Date]), MAX(Calendar[Date], day))

DATEDIFF(MIN(Calendar[Date]), MAX(Calendar[Date], week))

DATEDIFF(MIN(Calendar[Date]), MAX(Calendar[Date], month))

DATEDIFF(MIN(Calendar[Date]), MAX(Calendar[Date], quarter))

DATEDIFF(MIN(Calendar[Date]), MAX(Calendar[Date], year))

DATEVALUE Function (DAX)

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

Syntax: **DATEVALUE(date_text)**

EDATE Function (DAX)

Returns the date that is the indicated number of months before or after the start date. Use EDATE to calculate maturity dates or due dates that fall on the same day of the month as the date of issue.

Syntax: **EDATE(<startdate>,<months>)**

EOMONTH Function (DAX)

Returns the date in **datetime** format of the last day of the month, before or after a specified number of months. Use EOMONTH to calculate maturity dates or due dates that fall on the last day of the month.

Syntax: **EOMONTH(<start_date>,<,months>)**

TIMEVALUE Function (DAX)

Converts a time in text format to a time in datetime format.

Syntax: **TIMEVALUE(time_text)**

TODAY Function (DAX)

Returns the current date.

Syntax: **TODAY()**

CLOSINGBALANCEMONTH Function (DAX)

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

Syntax: **CLOSINGBALANCEMONTH(<expression>,<dates>[,<filter>])**

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

Return Value:
A scalar value that represents the **expression** evaluated at the last date of the month, in the current context.

CLOSINGBALANCEQUARTER Function (DAX)

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

Syntax: **CLOSINGBALANCEQUARTER(<expression>,<dates>[,<filter>])**

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

Return Value:
A scalar value that represents the **expression** evaluated at the last date of the quarter, in the current context.

CLOSINGBALANCEYEAR Function (DAX)

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

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

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

DATEADD Function (DAX)

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

Syntax: **DATEADD(<dates>,<number_of_intervals>,<interval>)**

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

Return Value:
A table containing a simple column of date values.

DATESBETWEEN Function (DAX)

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

Syntax: **DATESBETWEEN(<dates>,<start_date>,<end_date>)**

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

Return Value:
A table containing a single column of date values.

DATESINPERIOD Function (DAX)

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

Syntax: **DATESINPERIOD(<dates>,<start_date>,<number_of_intervals>,<interval>)**

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

Return Value:
A table containing a single column of date values.

DATESMTD Function (DAX)

Returns a table that contains a column of the dates for the month_to_date, in the current context.

Syntax: **DATESMTD(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Property Value/Return Value:
A table containing a single column of date values.

DATESQTD Function (DAX)

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

Syntax: **DATESQTD(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Property Value/Return Value:
A table containing a single column of date values.

DATESYTD Function (DAX)

Returns a table that contains a column of the dates for the year_to_date, in the current context.

Syntax: **DATESYTD(<dates>[,<year_end_date>])**

Parameter	Definition
dates	A column that contains dates.
year_end_date	A literal string with a date that defines the year_end_date. The default is December 31st. (Optional)

Property Value/Return Value:
A table containing a single column of date values.

ENDOFMONTH Function (DAX)

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

Syntax: **ENDOFMONTH(<dates>)**

Parameter	Definition
dates	A column that contains dates.

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

ENDOFQUARTER Function (DAX)

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

Syntax: **ENDOFQUARTER(<dates>)**

Parameter	Definition
dates	A column that contains dates.

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

ENDOFYEAR Function (DAX)

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

Syntax: **ENDOFYEAR(<dates>)[,<year_end_date>])**

Parameter	Definition
dates	A column that contains dates.
year_end_date	A literal string with a date that defines the year_end_date. The default is December 31st. (Optional)

FIRSTDATE Function (DAX)

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

Syntax: **FIRSTDATE(<daters>)**

Parameter	Definition
dates	A column that contains dates.

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

FIRSTNONBLANK Function (DAX)

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

Syntax: **FIRSTNONBLANK(<column>,<expressed>)**

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

Property Value/Return Value:
A table containing a single column and single row with the computed first value.

LASTDATE Function (DAX)

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

Syntax: **LASTDATE(<dates>)**

Parameter	Definition
dates	A column that contains dates.

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

LASTNONBLANK Function (DAX)

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

Syntax: **LASTNONBLANK(<column>,<expression>)**

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

NEXTDAY Function (DAX)

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

Syntax: **NEXTDAY(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Return Value:
A table containing a single column of date values.

NEXTMONTH Function (DAX)

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

Syntax: **NEXTMONTH(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Return Value:
A table containing a single column of date values.

NEXTQUARTER Function (DAX)

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

Syntax: **NEXTQUARTER(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Return Value:
A table containing a single column of date values.

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

NEXTYEAR Function (DAX)

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

Syntax: **NEXTYEAR(<dates>[,<year_end_date])**

Parameter	Definition
dates	A column that contains dates.
year_end_date	A literal string with a date that defines the year-end date. The default is December 31st. (Optional)

Return Value:
A table containing a single column of date values.

=CALCULATE(SUM(InternetSales_USA[SalesAmount_USD]),NEXTYEAR('DateTime'[DateKey]))

OPENINGBALANCEMONTH Function (DAX)

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

Syntax: **OPENINGBALANCEMONTH(<expression>,<dates>[,<filter>])**

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

Return Value:
A scalar value that represents the **expression** evaluated at the first date of the month in the current context.

=OPENINGBALANCEMONTH(SUMX(ProductInventory,ProductInventory[UnitCost]*ProductInventory[UnitsBalance]),DateTime[DateKey])

OPENINGBALANCEQUARTER Function (DAX)

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

Syntax: **OPENINGBALANCEQUARTER(<expression>,<dates>[,filter>])**

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

Return Value:
A scalar value that represents the **expression** evaluated at the first date of the quarter in the current context.

=OPENINGBALANCEQUARTER(SUMX(ProductInventory,ProductInventory[UnitCost]*ProductInventory[UnitsBalance]),DateTime[DateKey])

OPENINGBALANCEYEAR Function (DAX)

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

Syntax: **OPENBALANCEYEAR(<expression>,<dates>[,<filter>][,<year_end_date>])**

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

Return Value:
A scalar value that represents the **expression** evaluated at the first date of the year in the current context.

=OPENINGBALANCEYEAR(SUMX(ProductInventory,ProductInventory[UnitCost]*ProductInventory[UnitsBalance]),DateTime[DateKey])

PARALLELPERIOD Function (DAX)

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

Syntax: **PARALLELPERIOD(<dates>,<number_of_intervals>,<interval>)**

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

Return Value:
A table containing a single column of date values.

=CALCULATE(SUM(InternetSales_USD[SalesAmount_USD]),PARALLELPERIOD(DateTime[DateKey],-1,year))

PREVIOUSDAY Function (DAX)

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

Syntax: **PREVIOUSDAY(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Return Value:
A table containing a simple column of date values.

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

PREVIOUSMONTH Function (DAX)

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

Syntax: **PREVIOUSMONTH(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Return Value:
A table containing a single column of date values.

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

PREVIOUSQUARTER Function (DAX)

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

Syntax: **STARTOFQUARTER(<dates>)**

Parameter	Definition
dates	A column that contains dates.

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

=STARTOFQUARTER(DateTime[DateKey])

PREVIOUSYEAR Function (DAX)

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

Syntax: **PREVIOUSYEAR(<dates>[,<year_end_date>])**

Parameter	Definition
dates	A column that contains dates.
year_end_date	A literal string with a date that defines the year-end date. The default is December 31st. (Optional)

Return Value:
A table containing a single column of date values.

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

SAMEPERIODLASTYEAR Function (DAX)

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

Syntax: **SAMEPERIODLASTYEAR(<dates>)**

Parameter	Definition
dates	A column that contains dates.

Property Value/Return Value:
A single column table of date values.

=CALCULATE(SUM(ResellerSales_USD[SalesAmount_USD, SAMEPERIODLASTYEAR(DateTime[DateKey]))

STARTOFMONTH Function (DAX)

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

Syntax: **STARTOFMONTH(<dates>)**

Parameter	Definition
dates	A column that contains dates.

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

=STARTOFMONTH(DateTime[DateKey])

STARTOFQUARTER Function (DAX)

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

Syntax: **STARTOFQUARTER(<dates>)**

Parameter	Definition
dates	A column that contains dates.

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

=STARTOFQUARTER(DateTime[DateKey])

STARTOFYEAR Function (DAX)

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

Syntax: **STARTOFYEAR(<dates>)**

Parameter	Definition
dates	A column that contains dates.
year_end_date	A year_end_date value. (Optional)

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

=STARTOFYEAR(DateTime[DateKey])

TOTALMTD Function (DAX)

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

Syntax: **TOTALMTD(<expression>,<dates>[,<filter>])**

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

Return Value:
A scalar value that represents the **expression** evaluated for the dates in the current month-to-date, given the dates in **dates**.

=TOTALMTD(SUM(InternetSales_USD[SalesAmount_USD]),DateTime[DateKey])

TOTALQTD Function (DAX)

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

Syntax: **TOTALQTD(<expression>,<dates>[,<filter>])**

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

Return Value:
A scalar value that represents the **expression** evaluated for all dates in the current quarter to date given the dates in **dates**.

=TOTALQTD(SUM(InternetSales_USD[SalesAmount_USD]),DateTime[DateKey])

TOTALYTD Function (DAX)

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

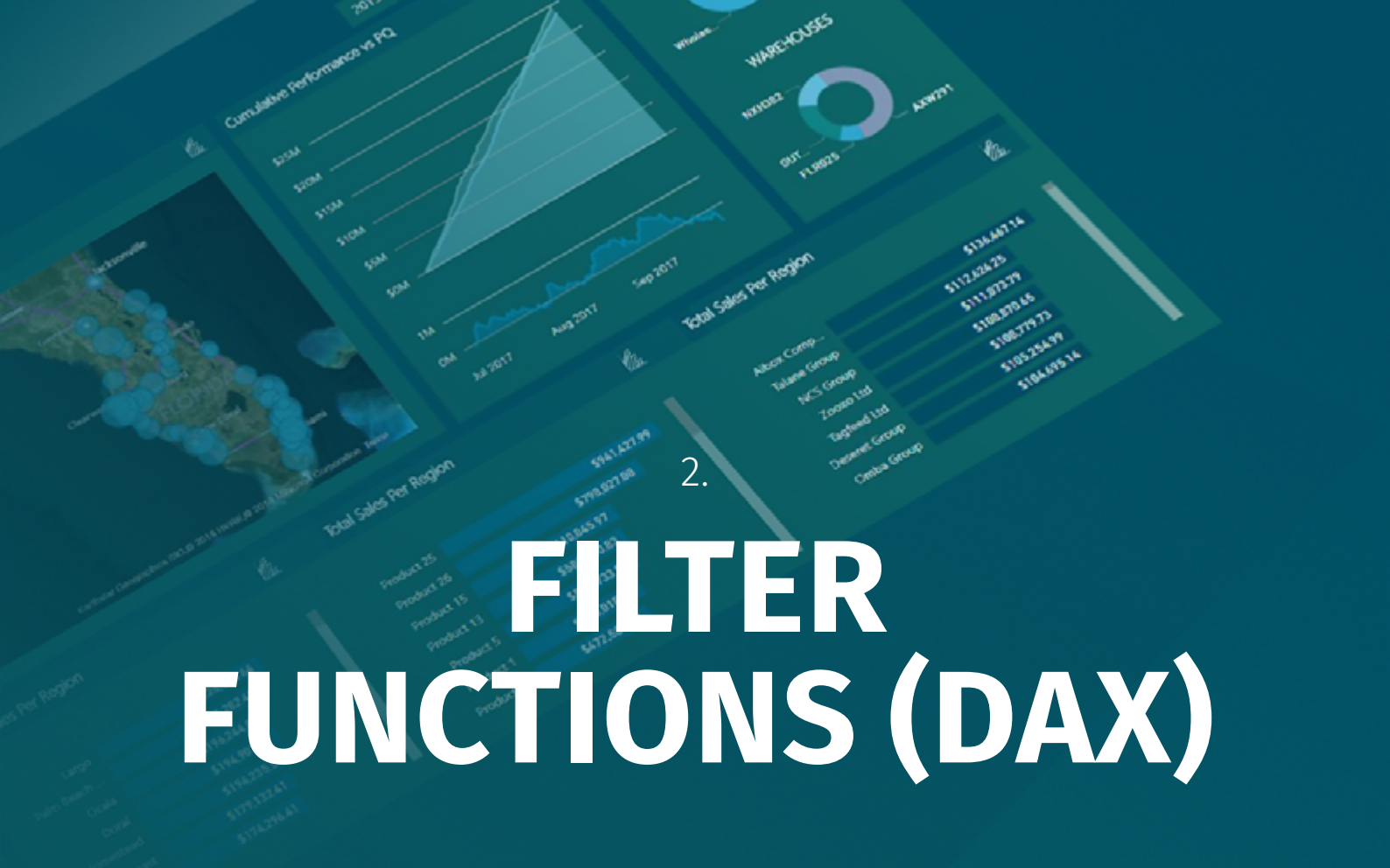
Syntax: **TOTALYTD(<expression>,<dates>[,<filter>][,<year_end_date>])**

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

Return Value:
A scalar value that represents the **expression** evaluated for the current year-to-date **dates**.

=TOTALYTD(SUM(InternetSales_USD[SalesAmount_USD]),DateTime[DateKey],ALL('DateTime'), "6/30")

=TOTALYTD(SUM(InternetSales_USD[SalesAmount_USD]),DateTime[DateKey]



2.

FILTER FUNCTIONS (DAX)

ALL Function (DAX)

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>[,...]]]})**

Parameter	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 functions.

As described in the following table, you can use the ALL and ALLEXCEPT functions in different scenarios.

Function and Usage	Definition
ALL(Table)	Remove all filters from the specified table. In effect, ALL(Table) returns all of the values in the table, removing any filters from the context that otherwise might have been applied. This function is useful when you are working with many levels of grouping, and want to create a calculation that creates a ratio of an aggregated value to the total value. The first example demonstrates this scenario.
ALL(Column[, Column[,…]])	Removes all filters from the specified columns in the table; all other filters on other columns in the table still apply. All column arguments must come from the same table. The ALL(Column) variant is useful when you want to remove the context filters for one or more specific columns and to keep all other context filters.
ALLEXCEPT(Table,Column1 [,Column2]…)	Removes all context filters in the table, except filters that are applied to the specified columns. This is a convenient shortcut for situations in which you want to remove the filters on many, but not all, columns in a table.

ALLEXCEPT Function (DAX)

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

Syntax: **ALLEXCEPT(<table>,<column>[,<column>[,...]])**

Parameter	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 with the ALLEXCEPT function.

Return Value:
A table with all filters removed, except for the filters on the specified columns.

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

ALLSELECTED Function (DAX)

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

Parameter	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.

CALCULATE Function (DAX)

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

Syntax: **CALCULATE(<expression>,<filter1>,<filter2>...)**

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

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

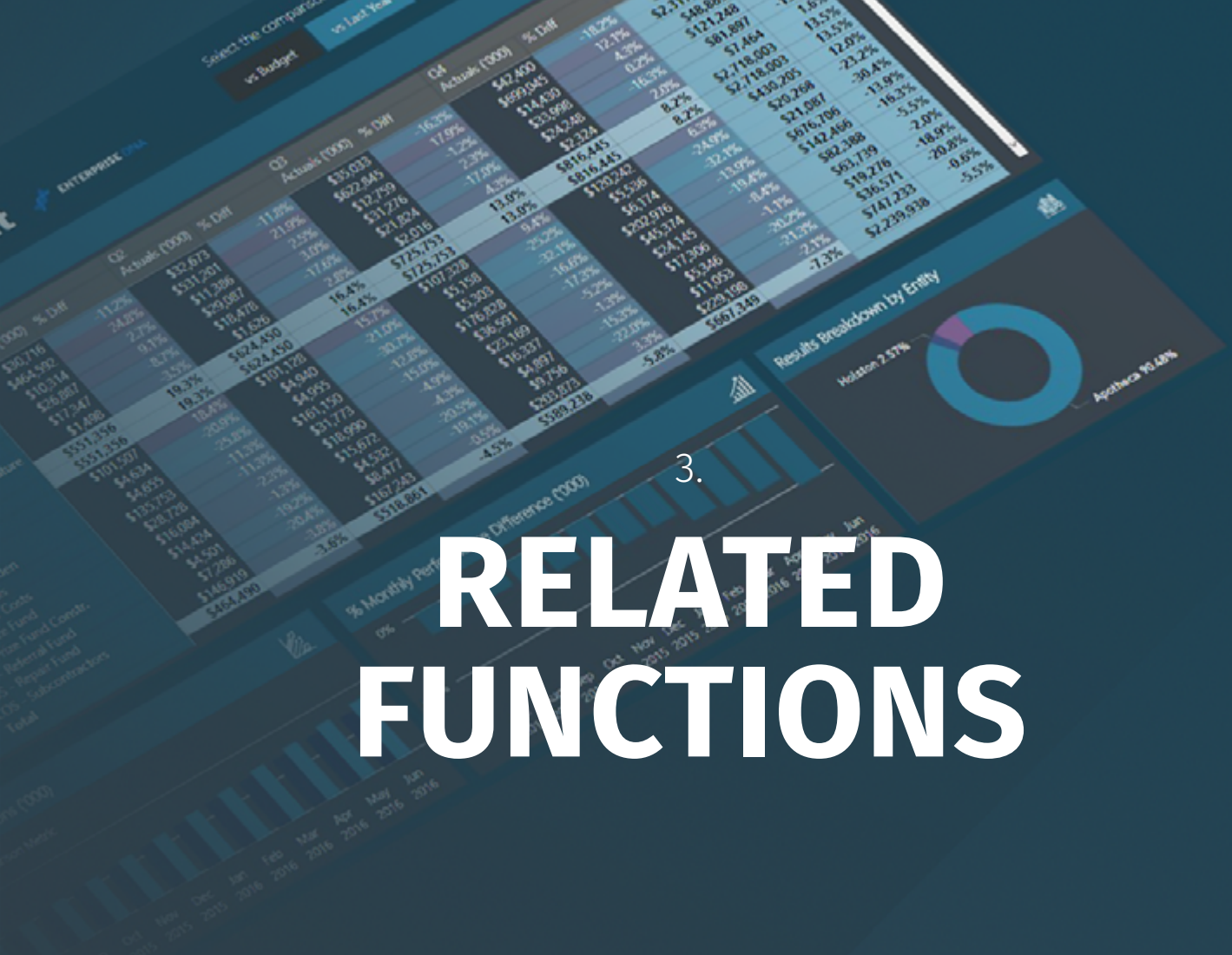
The following restrictions apply to Boolean expressions that are used as arguments.

- 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.

Remarks:

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



3.

RELATED FUNCTIONS

Whereas the CALCULATE function requires as its first argument an expression that returns a single value, the CALCULATETABLE function takes a table of values.

CALCULATETABLE Function (DAX)

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

Syntax: **CALCULATETABLE(<expression>,<filter1>,<filter2>,...)**

Parameter	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.

Remarks:
The CALCULATETABLE function changes the context in which the date is filtered, and evaluates the expression in the new context that you specify. For each column used in a filter argument, any existing filters on that column are removed, and the filter used in the filter argument is applied instead. This function is a synonym for the RELATEDTABLE function.

DISTINCT Function (DAX)

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: **DISTINCT(<column>)**

Parameter	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.

EARLIER Function (DAX)

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

EARLIER is used for nested calculations where you want to use a certain value as an input and produce calculations based on the 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>)**

Parameter	Definition
column	A column or expression that resolves to a column.
num	A positive number to the outer evaluation pass. (Optional) 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.

FILTER Function (DAX)

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

Syntax: **FILTER(<table>,<filter>)**

Parameter	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.

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

HASONEFILTER Function (DAX)

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

Syntax: **HASONEFILTER(<columnName>)**

Parameter	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**.

Remarks:
This function is similar to HASONEVALUE() with the difference that HASONEVALUE() works based on cross-filters, while HASONEFILTER() works by a direct filter.

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())

HASONEVALUE Function (DAX)

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

Syntax: **HTML**
HASONEVALUE(<columnName>)

Parameter	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**.

ISCROSSFILTERED Function (DAX)

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

Syntax: **ISCROSSFILTERED(<columnName>)**

Parameter	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**.

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

The related function ISFILTERED Function (DAX) returns TRUE when *columnName* is filtered directly.

ISFILTERED Function (DAX)

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>)**

Parameter	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.

Remarks:
columnName is said to be filtered directly when the filter or filters apply over the column; a column is said to 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 Function (DAX)

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

Syntax: **KEEPFILTERS(<expression>)**

Parameter	Definition
expression	Any expression.

Return Value:
No return value.

Remarks:
You use KEEPFILTERS within the context CALCULATE and CALCULATETABLE functions, to override the standard behavior of those functions.

By default, filter arguments in functions, such as CALCULATE, are used as the context for evaluating the expression and, as such, filter arguments for CALCULATE replace all existing filters over the same columns. The new context effected by the filter argument for CALCULATE affects only existing filters on columns mentioned as part of the filter argument. Filters on columns other than those mentioned in the arguments of CALCULATE, or other related functions remain in effect and unaltered.

The KEEPFILTERS function allows you to modify this behavior. When you use KEEPFILTERS, any existing filters in the current context are compared with the columns in the filter arguments and the intersection of those arguments is used as the context for evaluating the expression. The net effect over any one column is that both sets of arguments apply; both the filter arguments used in CALCULATE and the filters in the arguments of the KEEPFILTER function. In other words, whereas CALCULATE filters replace the current context. KEEPFILTERS adds filters to the current context.

RELATED Function (DAX)

Returns a related value from another table.

Syntax: **RELATED(<column>)**

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

Return Value:
A single value that is related to the current row.

Remarks:
The RELATED function requires that a relationship exists between the current table and the table with related information. You specify the column that contains the data that you want, and the function follows an existing many-to-one relationship to fetch the value from the specified column in the related table. If a relationship does not exist, you must create a relationship.

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

Note: The RELATED function needs a row context; therefore, it can only be used in calculated column expression, where the current row context is unambiguous, or as a nested function in an expression that uses a table scanning function. A table scanning function, such as SUMX, gets the value of the current row value and then scans another table for instances of that value.

Example:
=SUMX(FILTER('InternetSales_USD',RELATED('SalesTerritown'[SalesTerritoryCountry])<>" United States"),'InternetSales_USD'[SalesAmount_USD])

The following table shows only totals for each region, to prove that the filter expression in the measure. Non-USA Internet Sales works as intended.

RELATEDTABLE Function (DAX)

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

Syntax: **RELATEDTABLE(<tableName>)**

Parameter	Definition
tableName	The name of an existing table using standard DAX syntax. It cannot be an expression.

Return Value:
A table of values.

Remarks:
The RELATEDTABLE function changes the context in which the data is filtered, and evaluates the expression in the new context that you specify.

This function is a shortcut for CALCULATETABLE function with no logical expression.

Example:
=SUMX(RELATEDTABLE('InternetSales_USD'),[SalesAmount_USD])

USERELATIONSHIP Function (DAX)

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

Syntax: **USERELATIONSHIP(<columnName1>,<columnName2>)**

Parameter	Definition
columnName1	

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

VALUES Function (DAX)

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

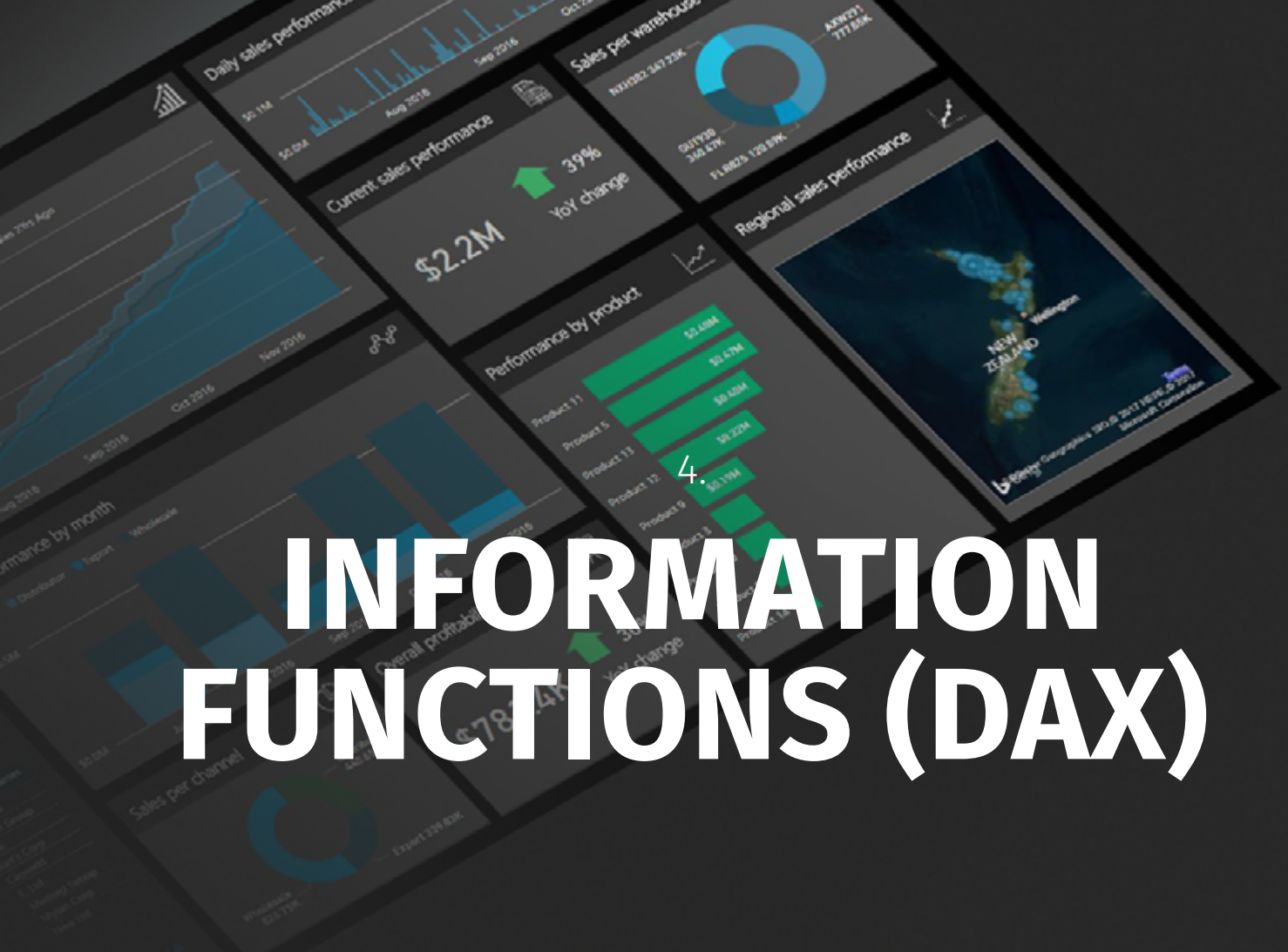
Syntax: **VALUES(<TableNameOrColumnName>)**

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

Return Value:
A column of unique values.

Example:
The following formula counts the number of unique invoices (sales orders), and produces the following results when used in a report that includes the Product Category Names.

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



4.

INFORMATION FUNCTIONS (DAX)

CONTAINS Function (DAX)

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

Syntax: **CONTAINS(<tables>),<columnName>,<value>[,columnName>,<value>]...)**

Parameter	Definition
table	Any DAX expression that returns a table of data.
columnName	The name of an existing column, using standard DAX syntax. It cannot be an expression.
value	Any DAX expression that returns a single scalar value that is to be sought in columnName. The expression is to be evaluated exactly once and before it is passed to the argument list.

Return Value:
A value of **TRUE** if each specificed value can be found in the corresponding *columnName*, or are contained, in those columns; otherwise the function returns **FALSE**.

Example:
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,[Productivity],214,[CustomerKey],11185)

ISBLANK Function (DAX)

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

Syntax: **ISBLANK(<value>)**

Parameter	Definition
value	The value or expression you want to test.

Return Value:
A Boolean value of TRUE if the value is blank; otherwise FALSE.

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.

Sales to Previous Year Ratio:

=IF(ISBLANK('CalculatedMeasures'[PreviousYearTotalSales]),BLANK(),('CalculatedMeasures'[TotalSales'CalculatedMeasures'[PreviousYearTotalSales])/'CalculatedMeasures'[PreviousYearTotalSales])

ISERROR Function (DAX)

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

Syntax: **ISERROR(<value>)**

Parameter	Definition
value	The value you want to test.

Return Value:
A Boolean value of TRUE if the value is an error; otherwise FALSE.

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 ero. 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]))

ISNUMBER Function (DAX)

Checks whether a value is a number and returns TRUE or FALSE.

Syntax: **ISNUMBER(<value>)**

Parameter	Definition
value	The value you want to test.

Property Value/Return Value:
TRUE if the value is numeric; otherwise FALSE.

Example:
The following three samples show the behavior of ISNUMBER:

//RETURNS: Is number=
IF(ISNUMBER(0),"Is number", "Is Not number")

//RETURNS: Is number
=IF(ISNUMBR(3.1E-1),"Is number, "Is Not number")

//RETURNS: Is Not number
=IF(ISNUMBR("123"),"Is number", Is Not number')

ISTEXT Function (DAX)

Checks if a value is text and returns TRUE or FALSE.

Syntax: **ISTEXT(<value>)**

Parameter	Definition
value	The value you want to check.

Property Value/Return Value:
TRUE if the value is text; otherwise FALSE.

Example:
The following examples show the behavior of the ISTEXT function:

```
//RETURNS: Is Text
=IF(ISTEXT("text"), "Is Text", "Is Non-Text")
```

LOOKUPVALUE Function (DAX)

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

Syntax: **LOOKUPVALUE(<result_columnName>,<search_columnName>,<search_value>[,<search_columnName>,<search_value>]...)**

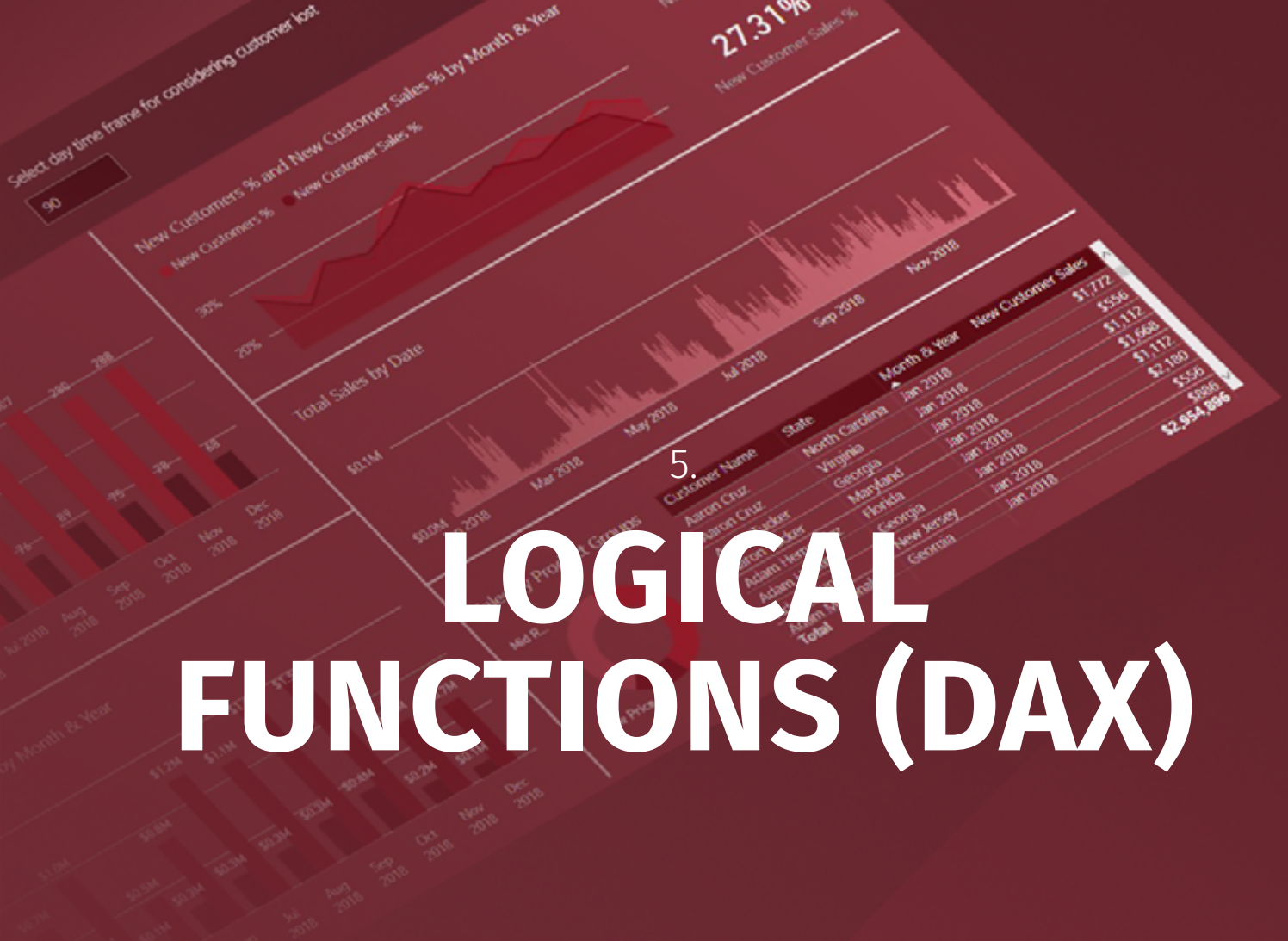
Parameter	Definition
result_columnName	The name of an existing column that contains the value you want to return. The column must be name using standard DAX syntax, usually fully qualified. It cannot be an expression.
search_columnName	The name of an existing column in the same table as result_columnName, or in a related table, over which the lookup is performed. The column must be named using standard DAX syntax, usually fully qualified. It cannot be an expression.
search_value	A scalar expression that does not refer to any column in the same table being searched.

Return Value:
The value of *result_column* at the row where allpairs of *search_column* and *search_value* have a match. If there is no match that satisfies all the search values, a BLANK is returned. In other words, the function will not return a lookup value if only some of the criteria match.

If multiple rows match the search values and, in all cases, *result_column* values are identical then that value is returned. However, if *result_column* returns different values an error is returned.

Example:
The following example returns the SafetyStockLevel for the bike model "Mountain-400-W Silver,46":

```
=LOOKUPVALUE(Product[SafetyStockLevel],[ProductName],"Mountain-400-W Silver,46")
```



5.

LOGICAL FUNCTIONS (DAX)

IF FUNCTION (DAX)

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)**

Parameter	Definition
logical_test	Any value or expression that can be evaluated to TRUE or FALSE.
value_if_true	The value that is returned if the logical test is TRUE. If omitted, TRUE is returned.
value_if_false	The value that is returned if the logical test is FALSE. If omitted, FALSE is returned.

Return Value:
Any type of value that can be returned by an expression.

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

IFERROR Function (DAX)

Evaluates an expression and returns a specified value if the expression returns an error; otherwise returns the value of the express itself.

Syntax: **IFERROR(value,value_if_error)**

Parameter	Definition
value	Any value or expression.
value_if_error	Any value or expression.

Return Value:
A scalar of the same type as **value**.

Example:
=IFERROR(25/0,9999)

SWITCH Function (DAX)

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

Syntax: **SWITCH(<expression>,<value>,<result>[,<value>,<result>]...[,<else>])**

Parameter	Definition
expression	Any DAX expression that returns a single scalar value where the expression is to be evaluated multiple times (for each row/context).
value	A constant value to be matched with the results of expression.
result	Any scalar expression to be evaluated if the results of expression match the corresponding value.
else	Any scalar expression to be evaluated if the result of expression does not match any of the value arguments.

Return Value:
A scalar value coming from one of the *result* expressions, if there was a match with *value*, or from the *else* expression, if there was no match with any *value*.

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")

TRUE Function (DAX)

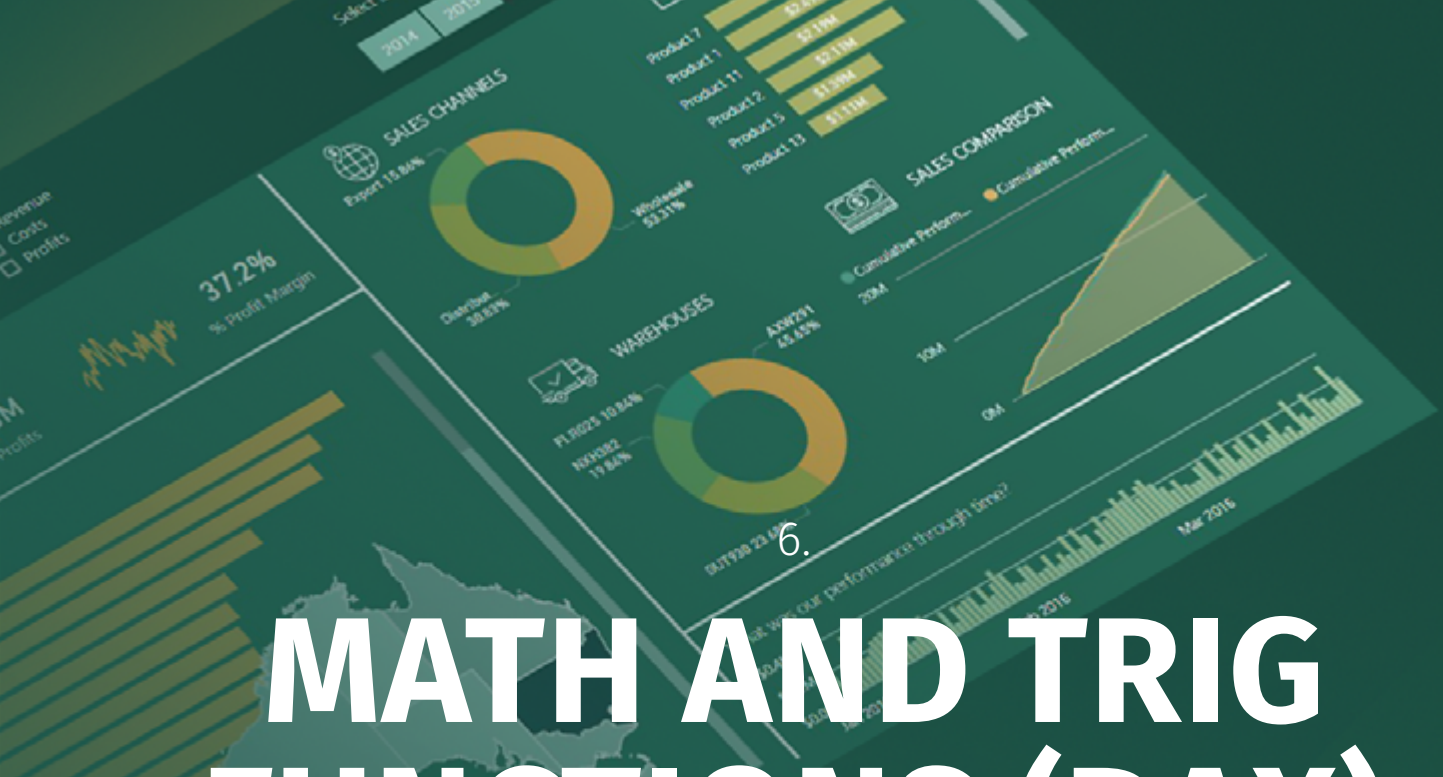
Returns the logical value TRUE.

Syntax: **TRUE()**

Return Value:
Always TRUE.

Remarks:
The word TRUE is also interpreted as the logical value TRUE.

Example:
=IF(SUM('InternetSales_USD'[SalesAmount_USD])>2000000,TRUE(),false())



6.

MATH AND TRIG FUNCTIONS (DAX)

DIVIDE Function (DAX)

Performs division and returns alternate result or BLANK() on division by 0.

Syntax: **DIVIDE(<numerator>,<denominator>[,<alternateresult>])**

Parameter	Definition
numerator	The dividend or number to divide.
denominator	The divisor or number to divide by.
alternateresult	The value returned when division by zero results in an error. When not provided, the default value is BLANK(). (Optional)

Return Value:
A decimal number.

Remarks:
Alternate result on divide by 0 must be a constant.

Example:
The following example returns 2.5:

=DIVIDE(5.2)

The following example returns BLANK:

=DIVIDE(5.0)

The following example returns 1:

=DIVIDE(5.0.1)

RAND Function (DAX)

Returns a random number greater than or equal to 0 or less than 1, evenly distributed. The number that is returned changes each time the cell containing this function is recalculated.

Syntax: **RAND()**

Return Value:
A decimal number.

Example:
To generate a random real number between two other numbers, you can use a formula like the following:

=RAND()*(int1-int2)+int1

RANDBETWEEN Function (DAX)

Returns a random number in the range between two numbers you specify.

Syntax: **RANDBETWEEN(<bottom>,<top>)**

Parameter	Definition
bottom	The smallest integer the function will return.
top	The largest integer the function will return.

Return Value:
A whole number.

Example:
The following formula returns a random number between 1 and 10.

=RANDBETWEEN(1,10)

SUMX Function (DAX)

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

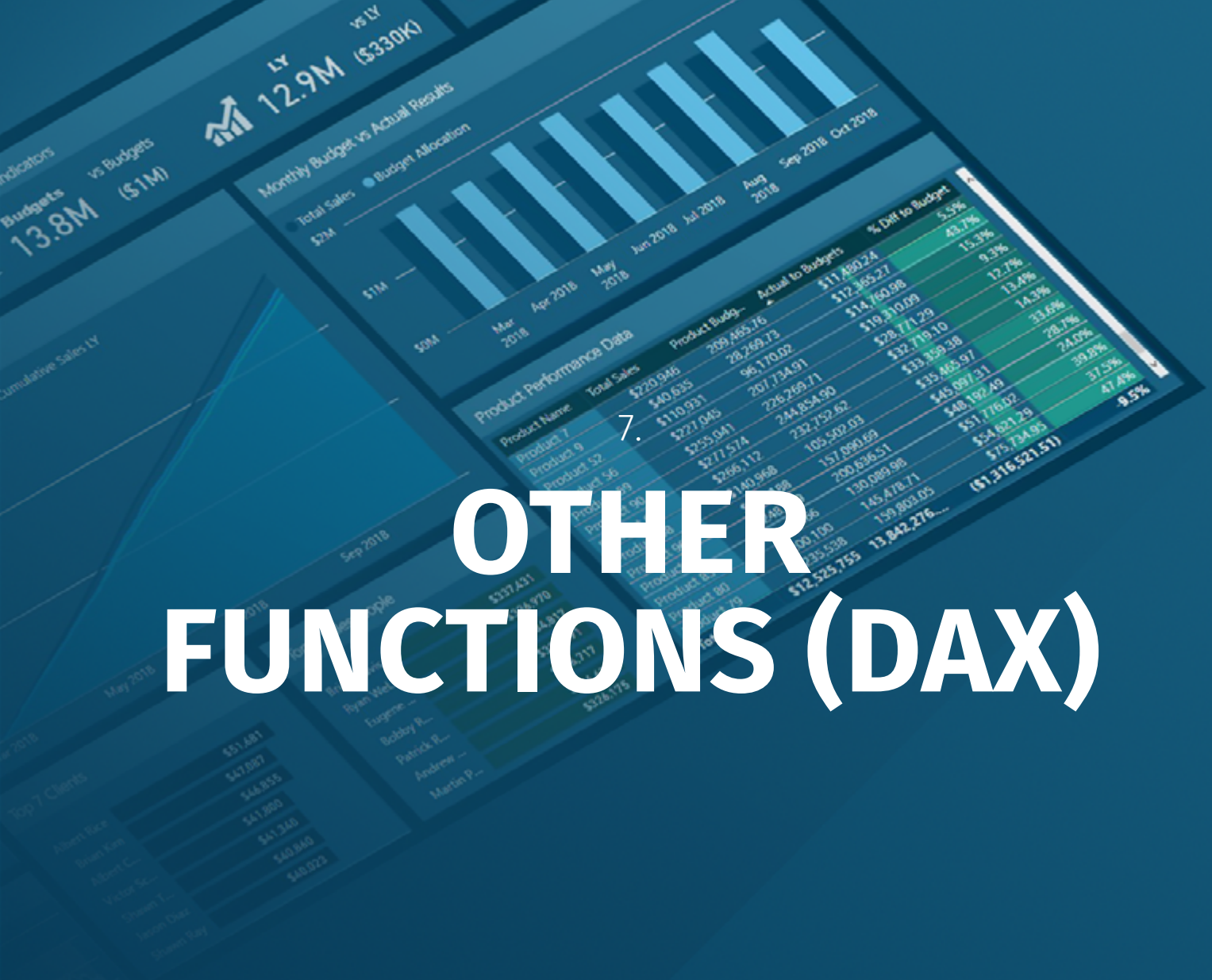
Syntax: **SUMX(<table>,<expression>)**

Parameter	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.

Example:
The following example first filters the table, InternetSales, on the expression, ShippingTerritory10 = 5, and then returns the sum of all values in the column, Frieght. In other words, the expression returns the sum of freight charges for only the specified sales area.

=SUMX(FILTER(InternetSales,InternetSales[SalesTerritoryID]=5),[Freight])

If you do not need to filter the column, use the SUM function. The SUM function is similar to the Excel function of the same name, except it takes a column as a reference.



7.

OTHER FUNCTIONS (DAX)

EXCEPT Function (DAX)

Returns the rows of one table which do not appear in another table.

Syntax: **EXCEPT(<table_expression1>,<table_expression2>)**

Parameter	Definition
table_expression	Any DAX expression that returns a table.

Return Value:
A table that contains the rows of one table minus all the rows of another table.

GROUPBY Function (DAX)

Syntax: **GROUPBY(<table>,[<groupBy_columnName1>],[<name>,<expression>]...)**

Return Value:
A table with the selected columns for the groupBy_columnName arguments and the groupedBy columns designated by the name arguments.

Example:
Assume a data model has four tables: Sales, Customer, Product, Georgraphy, where Sales is on the "many" side of a relationship to each of the other three tables.

GROUPBY(Sales,Geography[Country],Product[Category],"TotalSales",SUMX(CURRENTGROUP(),Sales[Price}*Sales[Qty]))

INTERSECT Function (DAX)

Returns the row intersection of two tables, retaining duplicates.

Syntax: **INTERSECT(<table_expression1>,<table_expression2>)**

Parameter	Definition
table_expression	Any DAX expression that returns a table.

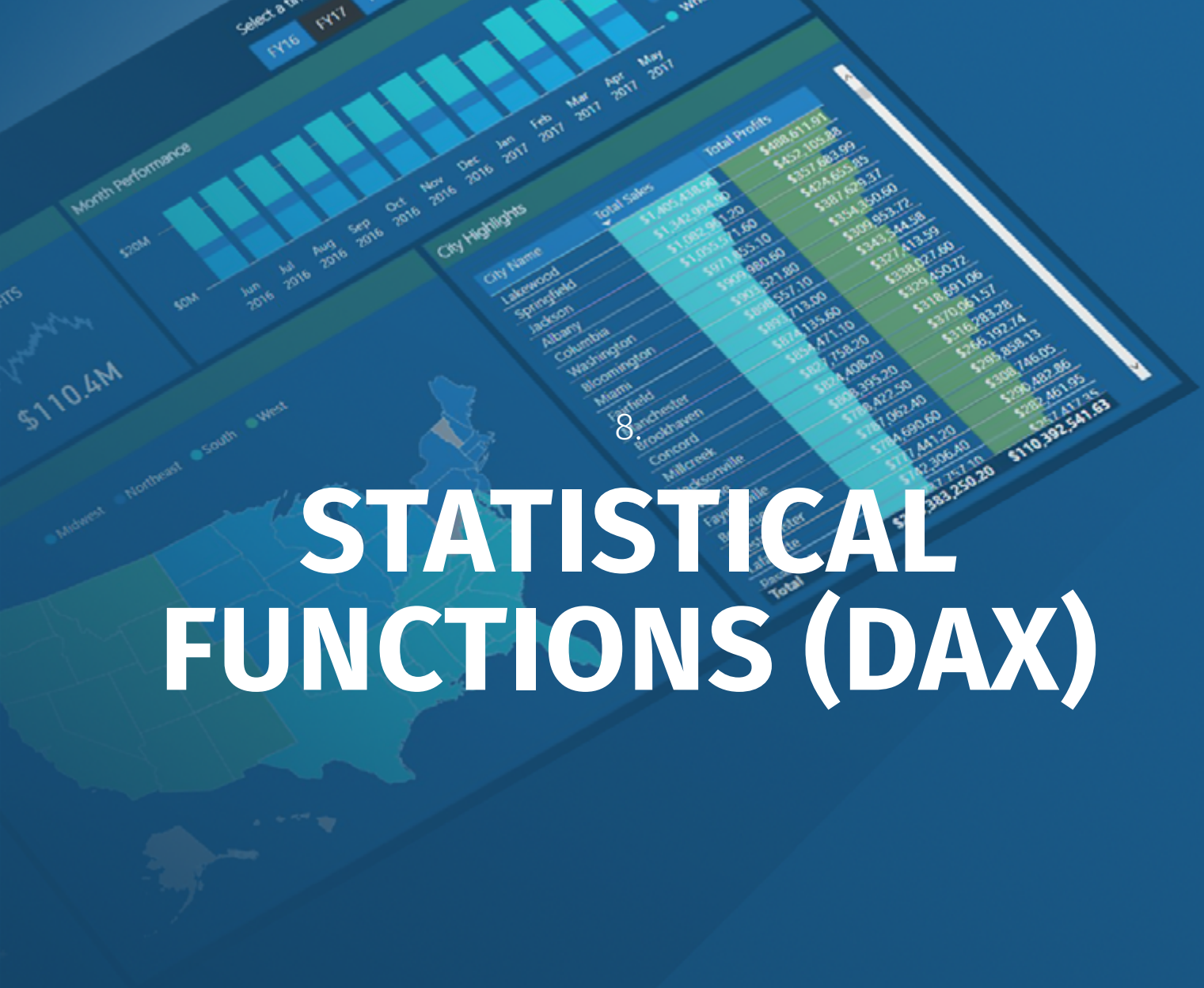
Return Value:
A table that contains all the rows in table_expression1 that are also in table_expression2.

SUMMARIZECOLUMNS Function (DAX)

Returns a summary table over a set of groups.

Syntax: **SUMMARIZECOLUMNS(<groupBy_columnName>[,<groupBy_columnName>]... ,[<filterTable>]...[,<name>,<expression>]...)**

Return Value:
A table which includes combinations of values from the supplied columns, based on the grouping specified. Only rows for which at least one of the supplied expressions return a non-blank value are included in the table returned. If all expressions evaluate to BLANK/NULL for a row, that row is not included in the table returned.



8.

STATISTICAL FUNCTIONS (DAX)

ADDCOLUMNS Function (DAX)

Adds calculated columns to the given table or table expression.

Syntax: **ADDCOLUMNS(<table>,<name>,<expression>[,<name>,<expression>]...)**

Parameter	Definition
table	Any DAX expression that returns a table of data.
name	The name given to the column enclosed in double quotes.
expression	Any DAX expression that returns a scalar expression evaluated for each row of table.

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

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,"InternetSales",SUMX(RELATEDTABLE(InternetSales_USD),InternetSales_USD[SalesAmount_USD]),"ResellerSales",SUMX(RELATEDTABLE(ResellerSales_USD[SalesAmount_USD]))

AVERAGE Function (DAX)

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

Syntax: **AVERAGE(<column>)**

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

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

=AVERAGE(InternetSales[ExtendedSalesAmount])

AVERAGEX Function (DAX)

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

Syntax: **AVERAGEX(<table>,<expression>)**

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

Return Value:
A decimal number.

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])

COUNT Function (DAX)

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

Syntax: **COUNT(<column>)**

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

Return Value:
A whole number.

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 Function (DAX)

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 non-blank values, including text, dates, and logical values.

Syntax: **COUNTA(<column>)**

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

Return Value:
A whole number.

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 Function (DAX)

The COUNTAX function counts non-blank 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 non-blank result.

Syntax: **COUNTAX(<table>,<expression>)**

Parameter	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 whole number.

Example:
The following example counts the number of non-blank 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 Function (DAX)

Counts the number of blank cells in a column.

Syntax: **COUNTBLANK(<column>)**

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

Return Value:
A whole number. If no rows are found that meet the condition, blanks are returned.

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.

COUNTROWS Function (DAX)

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

Syntax: **COUNTROWS(<table>)**

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

Return Value:
A whole number.

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

=COUNTROWS('Orders')

=COUNTROWS(RELATEDTABLE(ResellerSales))

COUNTX Function (DAX)

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>)**

Parameter	Definition
table	The table containing the rows to be counted.
expression	An expression that returns the set of values that contains the values you want to count.

Return Value:
An integer.

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

=COUNTX(Product,[ListPrice])

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

CROSSJOIN Function (DAX)

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>]...)**

Parameter	Definition
table	Any DAX expression thar returns a table of data.

Return Value:
A table that contains the Cartesian product of all rows from all tables in the arguments.

Example:
CROSSJOIN(Colors,Stationery)

DATATABLE Function

Provides a mechanism for declaring an inline set of data values.

Syntax: **DATATABLE(ColumnName1,DataType1,ColumnName2,DataType2... ,{{Value1,Value2...},{ValueN,ValueN+1...}...})**

Return Value:
A table declaring an inline set of values.

Example:
=DataTable("Name",STRING,"Region",STRING,("User1","East"),(User2","East"),("User3","West"),("User4","West"),("User4","East")))

DISTINCTCOUNT Function (DAX)

The DISTINCTCOUNT function counts the numer of different cells in a column of numbers.

Syntax: **DISTINCTCOUNT(<column>)**

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

Return Value:
The number of distinct values in *column*.

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

GENERATE Function (DAX)

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>)**

Parameter	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*.

Example:
GENERATE(SUMMARIZE(SalesTerritory,SalesTerritory[SalesTerritoryGroup]),SUMMARIZE(ProductCategory,[ProductCategoryName],"ResellerSales",SUMX(RELATEDTABLE(ResellerSales_USD),ResellerSales_USD[SalesAmount_USD])

GENERATEALL Function (DAX)

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>)**

Parameter	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*.

Example:
GENERATEALL(SUMMARIZE(SalesTerritory,SalesTerritown[SalesTerritoryGroup]),SUMMARIZE(ProductCategory,[ProductCategoryName],"ResellerSales",SUMX(RELATEDTABLE(ResellerSales_USD),ResellerSales_USD[SalesAmount_USD])))

MAX Function (DAX)

Returns the largest numeric value in a column.

Syntax: **MAX(<column>)**

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

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

=MAX(InternetSales[ExtendedAmount])

MAXX Function (DAX)

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

Syntax: **MAXX(<table>,<expression>)**

Parameter	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.

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(<table>,<expression>)MAXX(InternetSales, InternetSales[TaxAmt]+InternetSales[Freight])

MIN Function (DAX)

Returns the smallest numeric value in a column. Ignores logical values and text.

Syntax: **MIN(<column>)**

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

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

=MIN([ResellerMargin])

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([TransactionsDate])

MINX Function (DAX)

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

Syntax: **MINX(<table>,>expression>)**

Parameter	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.

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])

RANKX Function (DAX)

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>]]])**

Parameter	Definition
table	Any DAX expression that returns a table of data over which the expression is evaluated.
expression	Any DAX expression that returns a single scalar value. The expression is evaluated for each row of table, to generate all possible values for ranking. See the remarks section to understand the function behavior when expression evaluates to BLANK.
value	Any DAX expression that returns a single scalar value whose rank is to be found. See the remarks section to understand the function's behavior when value is not found in the expression. (Optional) When the value parameter is omitted, the value of expression at the current row is used instead.
order	A value that specifies how to rank value, low to high or high to low. (Optional)

Value	Alternate Value	Description
0 (zero)	FALSE	Ranks in descending order of values of expression. If value is equal to the highest number in expression then RANKX returns 1.
1	TRUE	Ranks in ascending order of expression. If value is equal to the lowest number in expression then RANKX returns 1.

Parameter	Definition
ties	An enumeration that defines how to determine ranking when there are ties. (Optional)

Enumeration	Description
skip	The next rank value, after a tie, is the rank value of the tie plus the count of tied values. For example, if five (5) values are tied with a rate of 11, then the next value will receive a rank of 16 (11+5). This is the default value when ties parameter is omitted.
dense	The next rank value, after a tie, is the next rank value. For example, if five (5) values are tied with a rank of 11, then the next value will receive a rank of 12.

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]))

ROW Function (DAX)

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

Syntax: **ROW(<name>,<expression>[[,<name>,<expression>]...])**

Parameter	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.

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

ROW("InternetTotalSales(USD)",SUM(InternetSales_USD[SalesAmount_USD]),"ResellersTotalSales(USD)",SUM(ResellerSales_USD[SalesAmount_USD]))

SELECTCOLUMNS Function (DAX)

Adds calculated columns to the given table or table expression.

Syntax: **SELECTCOLUMNS(<table>,<name>,<scalar_expression>[,<name>,<scalar_expression>]...)**

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

Return Value:
A table with the same number of rows as the table specified as the first argument. The returned table has one column for each pair of <name>, <scalar_expression> arguments, and each expression is evaluated in the context of a row from the specified <table> argument.

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

SUMMARIZE Function (DAX)

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

Syntax: **SUMMARIZE(<table>,<groupBy_columnName>[,<groupBy_columnName>]... [,<name>,<expression>]...)**

Parameter	Definition
table	Any DAX expression that returns a table of data.
groupBy_columnName	The qualified name of an existing column to be used to create summary groups based on the values found in it. This parameter cannot be an expression. (Optional)
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 selection columns for the *groupBy_columnName* arguments and the summarized columns designed by the name arguments.

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],"SalesAmount(USD)",SUM(ResellerSales_USD[SalesAmount_USD]),"DiscountAmount(USD)",SUM(ResellerSales_USD[DiscountAmount]))

9.

TEXT FUNCTIONS (DAX)

BLANK Function (DAX)

Returns a blank.

Syntax: **BLANK()**

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

FIND Function (DAX)

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>]])**

Parameter	Definition
find_text	The text you want to find. Use double quotes (empty text) to match the first character in within_text. You can use wildcard characters—the question mark (?) and asterisk (*)—in find_text. A question mark matches any single character, an asterisk matches any sequence of characters. If you want to find an actual question mark or asterisk, type a tilde (~) before the character.
within_text	The text containing the text you want to find.
start_num	The character at which to start the search; if omitted, start_num = 1. The first character in within_text is character number 1. (Optional)
NotFoundValue	The value that should be returned when the operation does not find a matching substring, typically 0, -1, or BLANK().

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")

ENTERPRISE DNA

+64 21 164 5508 | info@enterprisedna.co
www.enterprisedna.co