Blackbird Corporate Ltd offering first class **Microsoft 365** and **SharePoint** Services to your business

Power BI Desktop Advanced

Power BI is a suite of business analytics tools to analyse data and share insights. Power BI Desktop transforms your company's data into rich visuals for you to monitor your business and get answers quickly with rich dashboards available on every device.

Target student: This course is intended for people who want to use all the capabilities of the Power BI platform to build self-serviced business intelligence solutions.

Duration: 3 Day

Timings: 09:30 – 16:30

Max number of delegates: 8

Students will learn:

- Power BI Desktop Features
- Power BI Concepts and Terms
- Manage Data Visualizations
- Manage DAX formulas
- Publish and Share Power BI

Consultancy at contractor prices | A low risk choice | Consulting as a Subscription | Agile implementation | Flexible Consulting Framework | "Speed to Value" is our USP | Greater control over suppliers | Tracked communication | Decrease in lead times | Control over the process | Greater transparency | Current workflow status

Blackbird Corporate Ltd offering first class **Microsoft 365** and **SharePoint** Services to your business

DAY TWO

Day Start 09:30	Break 11:00 to 11:20	Lunch 12:30 to 13:15	Break 14:40 to 15:00	Day End 16:30

Module 4: Querying Data

- Query Editor
- Filters and Delimiters
- Transforms
- Pivot and Group By
- Creating Custom Calculated Columns
- Creating Tables
- Adding Conditional Columns

Module 5: Data Analysis Expression (DAX)

- DAX Overview
- DAX data types and operators
- Common DAX functions
- Referencing other tables in DAX
- Using CALCULATE and RANKX, RELATED, RELATEDTABLE, LOOKUPVALUE, PREVIOUSQUARTER, FILTER, COUNTROWS, EARLIER, RELATED, DISTINCT (table), DISTINCT (column), SUMMARIZE

Module 6: Reports and Exports

- Report Elements and Options
- Working with Pages
- Adding Graphics
- Report Level Filters
- Report Themes
- Export Power BI Data to CSV
- Create a Power BI Template

Module 7: Sharing Content

- Power BI Service
- Sharing Dashboards with internal and External users
- Sharing content with Office 365 groups
- Using Publish to Web with Dashboards

Consultancy at contractor prices · A low risk choice · Consulting as a Subscription · Agile implementation · Flexible Consulting Framework · "Speed to Value" is our USP · Greater control over suppliers · Tracked communication · Decrease in lead times · Control over the process · Greater transparency · Current workflow status

Blackbird Corporate Ltd offering first class **Microsoft 365** and **SharePoint** Services to your business

Module 5: Data Analysis Expression (DAX)

DADA ENTERPRISES

Method 1: Quick measures

Many common calculations are available as *quick measures*, which write the DAX formulas for you based on your inputs in a window. These quick, powerful calculations are also great for learning DAX or seeding your own customized measures.

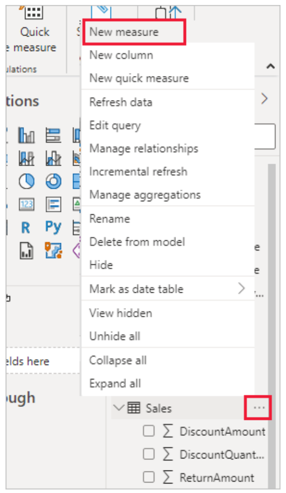
Create a quick measure using one of these methods:

- From a table in the **Fields** pane, right-click or select **More options** (**...**), and then choose **New quick measure** from the list.
- Under **Calculations** in the **Home** tab of the Power BI Desktop ribbon, select **New Quick Measure**.

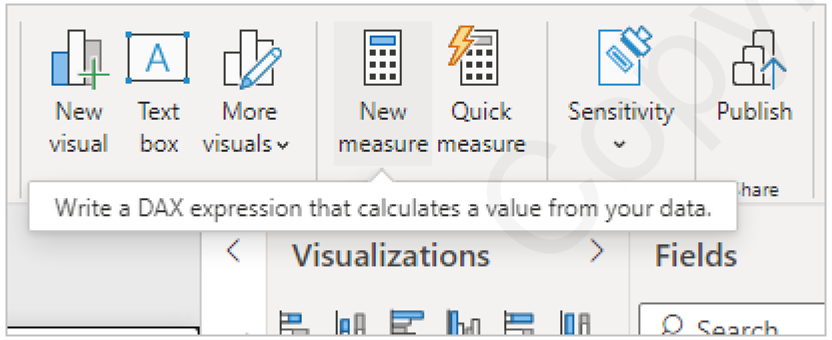
Method 2: Create a Custom Measure

To create a custom measure using one of the following method:

- From the menu that appears, choose **New measure**.
- You can also create a new measure by selecting **New Measure** in the **Calculations** group on the **Home** tab of the Power BI Desktop ribbon.

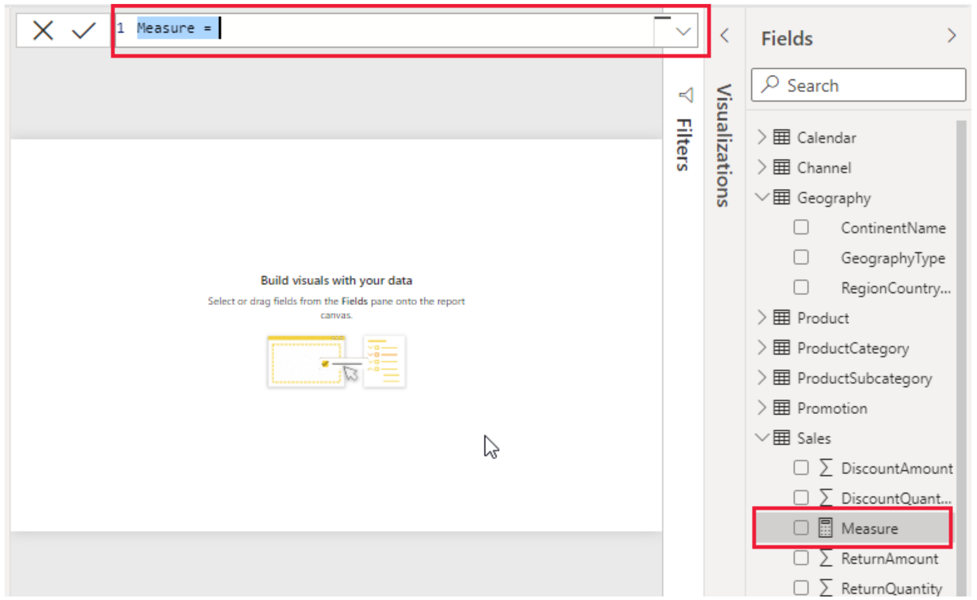


You can also create a new measure by selecting **New Measure** in the **Calculations** group on the **Home** tab of the Power BI Desktop ribbon.



By default, each new measure is named Measure.

If you don't rename it, new measures are named Measure 2, Measure 3, and so on.



Task: Create a measure using the CALCULATE formula

1. Download and open the **The Adventure Works DW 2020** Power BI Desktop file.
2. In Report view, in the field list, right-click the **Sales** table, and then select **New Measure**.
3. In the formula bar, replace **Measure** by entering a new measure name, *Previous Quarter Sales*.
4. After the equals sign, type the first few letters *CAL*, and then double-click the function you want to use. In this formula, you want to use the **CALCULATE** function.

   You'll use the CALCULATE function to filter the amounts we want to sum by an argument we pass to the CALCULATE function. This type of function is referred to as nesting functions. The CALCULATE function has at least two arguments. The first is the expression to be evaluated, and the second is a filter.

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

5. After the opening parenthesis *(* for the **CALCULATE** function, type *SUM* followed by the following DAX formula below:

```
BB_Total sales on last selected date =
CALCULATE (
   SUM ( ‘Sales’[Sales Amount] ),
   'Sales'[OrderDateKey] = MAX ( 'Sales'[OrderDateKey] )
)
```

6. Select the checkmark ✓ in the formula bar or press Enter to validate the formula and add it to the Sales table.

Task: Create a measure using the CALCULATE formula

1. Download and open the **The Adventure Works DW 2020** Power BI Desktop file.
2. In Report view, in the field list, right-click the **Sales** table, and then select **New Measure**.
3. In the formula bar, replace **Measure** by entering a new measure name, *Previous Quarter Sales*.
4. After the equals sign, type the first few letters *CAL*, and then double-click the function you want to use. In this formula, you want to use the **CALCULATE** function.
   You'll use the CALCULATE function to filter the amounts we want to sum by an argument we pass to the CALCULATE function. This type of function is referred to as nesting functions. The CALCULATE function has at least two arguments. The first is the expression to be evaluated, and the second is a filter.
5. After the opening parenthesis *(* for the **CALCULATE** function, type *SUM* followed by another opening parenthesis *(*.
   Next, we'll pass an argument to the SUM function.
6. Begin typing *Sal*, and then select **Sales[SalesAmount]**, followed by a closing parenthesis *)*.
   This step creates the first expression argument for our CALCULATE function.
7. Type a comma (,) followed by a space to specify the first filter, and then type *PREVIOUSQUARTER*.
   You'll use the PREVIOUSQUARTER time intelligence function to filter SUM results by the previous quarter.
8. After the opening parenthesis *(* for the PREVIOUSQUARTER function, type *Date[DateKey]*.
   The PREVIOUSQUARTER function has one argument, a column containing a contiguous range of dates. In our case, that's the DateKey column in the Date table.
9. Close both the arguments being passed to the PREVIOUSQUARTER function and the CALCULATE function by typing two closing parenthesis *))*.
   Your formula should now look like this:
   BB_Previous Quarter Sales = CALCULATE(SUM('Sales_Amount_2023'[Sales Amount]), PREVIOUSQUARTER('Date'[DateKey]))

   Select the checkmark ✓ in the formula bar or press Enter to validate the formula and add it to the Sales table.

Task: DAX formula using FILTER function

You can use FILTER to reduce the number of rows in the table that you are working with and use only specific data in calculations. FILTER is not used independently, but as a function that is embedded in other functions that require a table as an argument.

Basic Syntax

FILTER(<table>,<filter>)

Example

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

Using the FILTER function nested in other calculations.

The following example first filters the table, Sales, on the expression, 'Sales[SalesTerritoryID] = 5`, and then returns the sum of all values in the Sales Amount column. In other words, the expression returns the sum of Sales Amount for only the specified sales area.

Worked Example

BB_Sum of South East USA = SUMX(FILTER('Sales', Sales[SalesTerritoryKey]=5),[Sales Amount])

Task: DAX formula using EARLIER function in CALCULATED COLUMN

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. To illustrate the use of EARLIER, it is necessary to build a scenario that calculates a rank value and then uses that rank value in other calculations.

Basic Syntax

EARLIER(<column>, <number>)

Example

The following example is based on this simple table, **ProductSubcategory Sample.xls file**, which shows the total sales for each ProductSubcategory.

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

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

The DAX formula essentially creates a dynamic range of the number of rows within the SubCategory Table where the Current Rows' TotalSubCategorySales is the greater. It will then Filter this range, and the Count the number of rows between the Current Row being evaluated and the top row of that range.

a) So, the **EARLIER** function in RED gets the value of *TotalSubcategorySales* for the current row in the table, whereas the function in GREEN denotes the number of rows where the value of *TotalSubcategorySales* is higher.
b) The **FILTER** function *in evaluating from row to row*, then return a dynamic table range where all rows have a value of *TotalSubcategorySales* larger than Current Row being evaluated.
c) The **COUNTROWS** function then counts the rows of the filtered table (in this dynamic range) and assigns that value to the new calculated column in the current row **plus 1**.
d) Adding a **'+1'** is needed to prevent the top ranked value from become a Blank, because at the Top Row the number of rows where the value of *TotalSubcategorySales* being higher is '0' so a +1 is needed.

Task: DAX formula using RELATED function

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.

Basic Syntax

RELATED(<column>)

So, in the the **The Adventure Works DW 2020** Power BI Desktop file you can use an existing relationship between Sales and SalesTerritory and explicitly state that the country must be different from the United States. To do so, create a filter expression like the following:

Worked Example

FILTER(‘Sales', RELATED('SalesTerritory'[SalesTerritoryCountry])<>"United States")

This expression uses the RELATED function to lookup the country value in the SalesTerritory table, starting with the value of the key column, SalesTerritoryKey, in the Sales table. The result of the lookup is used by the filter function to determine if the Sales row is filtered or not.

BB_Related_Measure = SUMX(FILTER(Sales, RELATED('Sales Territory'[Country])<>"United States"),'Sales'[Sales Amount])

Task: DAX formula using LOOKUPVALUE function in a CALCULATED COLUMN

Returns the value for the row that meets all criteria specified by one or more search conditions.

Basic Syntax

```
LOOKUPVALUE(
    <result_columnName>,
    <search_columnName>,
    <search_value>
    [, <search2_columnName>, <search2_value>]…
    [, <alternateResult>]
)
```

Worked Example

In the **The Adventure Works DW 2020** Power BI Desktop file, the following **calculated column** defined in the Sales table uses the LOOKUPVALUE function to return channel values from the Sales Order table.

```
BB_CHANNEL = LOOKUPVALUE('Sales Order'[Channel],'Sales Order'[SalesOrderLineKey],[SalesOrderLineKey])
```

However, in this case, because there is a relationship between the Sales Order and Sales tables, it's more efficient to use the RELATED function.

```
BB_RELATED_Column = RELATED('Sales Order'[Channel])
```

Task: CALCULATED COLUMN using RELATEDTABLE function in DAX formula

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.

Basic Syntax

RELATEDTABLE(<tableName>)

So, in the the **The Adventure Works DW 2020** Power BI Desktop, the following example uses the RELATEDTABLE function to create a **calculated column** with the Sales in the Product table:

Worked Example

BB_RELATEDTABLE_Measure = SUMX(RELATEDTABLE('Sales'),[Sales Amount])

RANKX formula explained

DAX Expression:

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

Parameters

table

Any DAX expression that returns a table of data over which the expression is evaluated.

expression

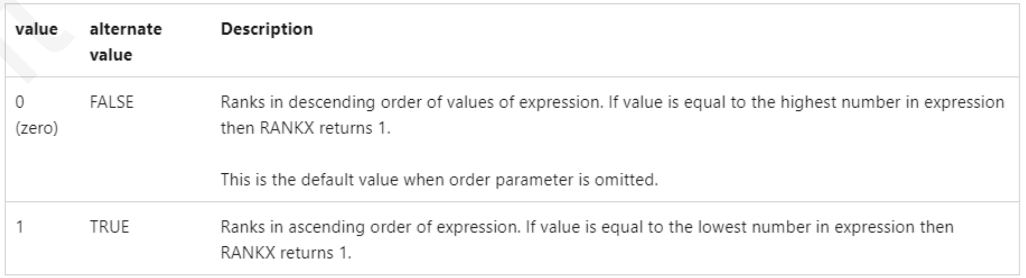
Any DAX expression that returns a single scalar value. The expression is evaluated for each row of *table*, to generate all possible values for ranking. See the remarks section to understand the function behaviour when *expression* evaluates to BLANK.

value

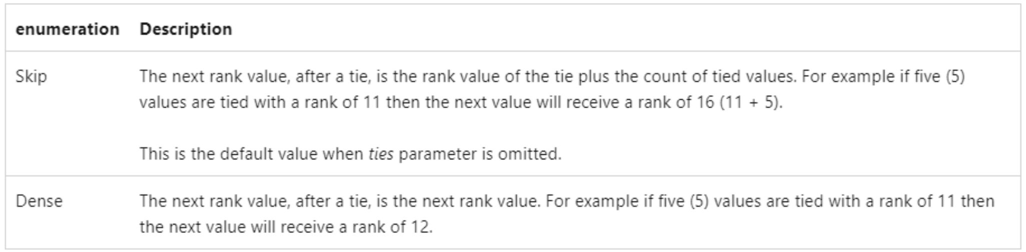
(Optional) Any DAX expression that returns a single scalar value whose rank is to be found. See the remarks section to understand the function's behaviour when *value* is not found in the expression. When the *value* parameter is omitted, the value of expression at the current row is used instead.

order

(Optional) A value that specifies how to rank *value*, low to high or high to low:



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. This is the default value when order parameter is omitted.
1	TRUE	Ranks in ascending order of expression. If value is equal to the lowest number in expression then RANKX returns 1.



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 rank 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.

Task: Create a CALCULATED COLUMN using the RANKX formula in DAX formula

Basic Syntax

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

- If *expression* or *value* evaluates to BLANK it is treated as a 0 (zero) for all expressions that result in a number, or as an empty text for all text expressions.
- If *value* is not among all possible values of *expression* then RANKX temporarily adds *value* to the values from *expression* and re-evaluates RANKX to determine the proper rank of *value*.

So, in the the **The Adventure Works DW 2020** Power BI Desktop, the following example uses the RELATEDTABLE function to create a **calculated column** with the Sales in the Product table:

Worked Example

The following calculated column in the Products table calculates the sales ranking for each product in the Sales Table.

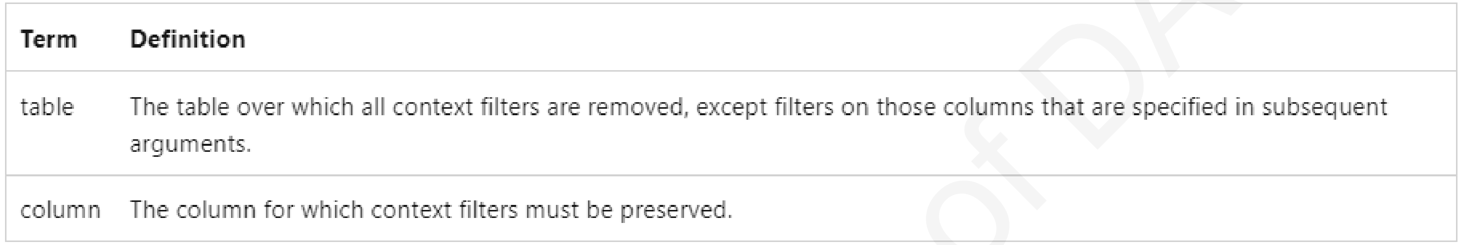
BB_RANKX_Measure = RANKX(ALL('Product'), SUMX(RELATEDTABLE(Sales), [Sales Amount]))

Task: DAX formula using ALLEXCEPT function

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

<u>Basic Syntax</u>

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



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.

Where:

- The first argument to the ALLEXCEPT function must be a reference to a base table.
- All subsequent arguments must be references to base columns and state the filters on specific columns to be preserved.

<u>Worked Example</u>

The following measure formula sums ‘Sales Amount’ Column from the Sales Table and uses the ALLEXCEPT function to remove any context filters that have been applied (e.g., on a slicer) except for those that have been applied to the ‘Fiscal Year’ Column in the Date Table.

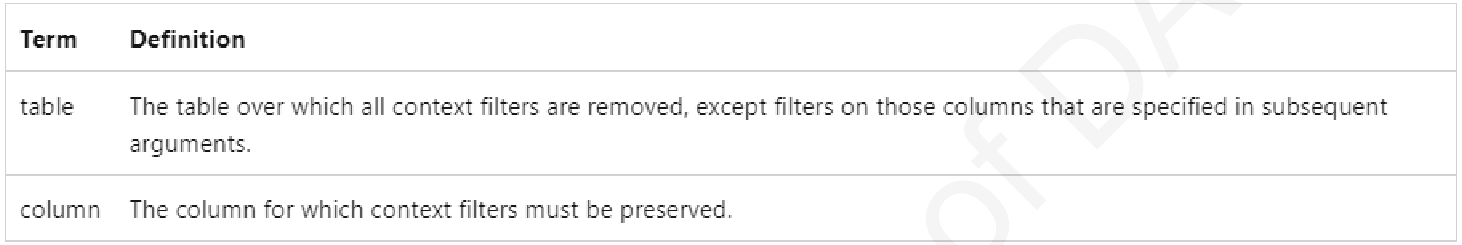
BB_ALLEXCEPT Measure = CALCULATE(SUM(Sales[Sales Amount]), ALLEXCEPT('Date','Date'[Fiscal Year]))

Task: DAX formula using ALLEXCEPT function

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

Basic Syntax

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



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.

Where:

- The first argument to the ALLEXCEPT function must be a reference to a base table.
- All subsequent arguments must be references to base columns and state the filters on specific columns to be preserved.

Worked Example1

The following measure formula Sums 'Sales Amount' Column from the Sales Table and uses the ALLEXCEPT function to remove any context filters that have been applied (e.g., on a slicer) except for those that have been applied to the 'Fiscal Year' Column in the Date Table.

BB_ALLEXCEPT Measure1 = CALCULATE(SUM(Sales[Sales Amount]), ALLEXCEPT('Date','Date'[Fiscal Year]))

Worked Example2

The following measure formula Counts the number of rows in the Sales Table and uses the ALLEXCEPT function to remove any context filters that have been applied except for those that have been applied to the 'OrderDateKey' Column in the Sales Table.

BB_ALLEXCEPT Measure2 = CALCULATE (COUNTROWS(Sales), ALLEXCEPT (Sales,Sales[OrderDateKey]))

Task: DAX formula using DISTINCT (column) function

Returns a one-column table that contains the distinct values from the specified column.

Basic Syntax

DISTINCT(<column>)

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

Worked Example1

The following formula counts the number of unique customers who have generated orders in the Sales Table.

BB_DISTINCT Measure = COUNTROWS(DISTINCT(Sales[CustomerKey]))

Task: DAX formula using DISTINCT (table) function

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

Basic Syntax

DISTINCT(<table>)

Worked Example1

The following formula returns a **result table** by removing duplicate rows from the Sales Table.

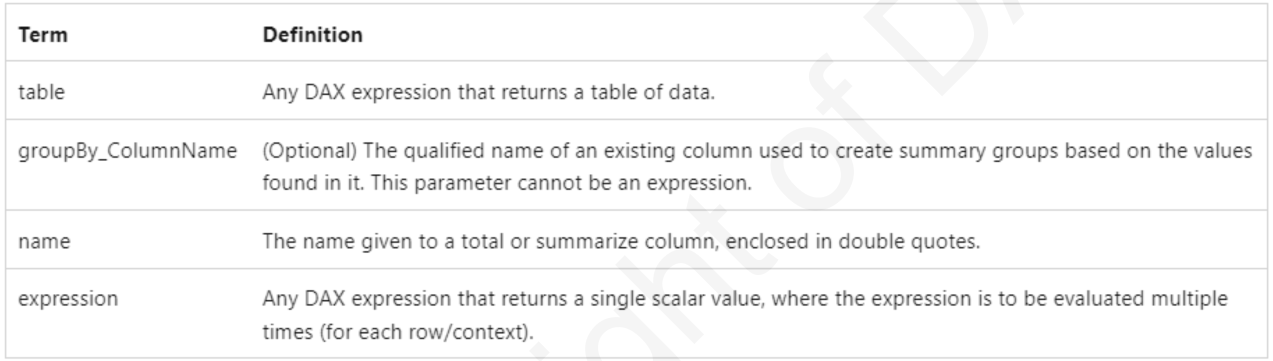
BB_DISTINCT Table = DISTINCT(Sales)

Task: DAX formula using SUMMARIZE function

Returns a summary table for the requested totals over a set of groups. A table with the selected columns for the groupBy_columnName arguments and the summarized columns designed by the name arguments.

<u>Basic Syntax</u>

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



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

<u>Worked Example</u>

The following example returns a summary of the Sales Table grouped around the Fiscal Year (in the Date Table) and the product category name, (from the Product Table) in a **result table** allows you to do analysis (aggregate sales & quantity) over the sales by year and product category.

```
BB_SUMMARIZE Table = SUMMARIZE(Sales
    , 'Date'[Fiscal Year]
    , 'Product'[Subcategory]
    , "Aggregate Sales Amount", SUM(Sales[Sales Amount])
    , "Aggregate Quantity", SUM(Sales[Order Quantity])
    )
```