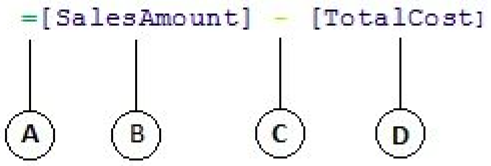
POWER BI
Workbook 3
EXPLORING DAX

INSTRUCTIONS

Create a new PBI instance, upload the SAMPLE SUPERSTORE file, implement all the queries in it and their outputs as well. Save the file and upload the file on the LMS.

I have taught you about Calculated Columns and Calculated Measures in the Class. Now, lets implement this but before that, let's revise DAX a little bit

Syntax of DAX includes the various elements that make up a formula, or more simply, how the formula is written. For example, let's look at a simple DAX formula used to create new data (values) for each row in a calculated column, named Margin, in a FactSales table: (formula text colors are for illustrative purposes only)

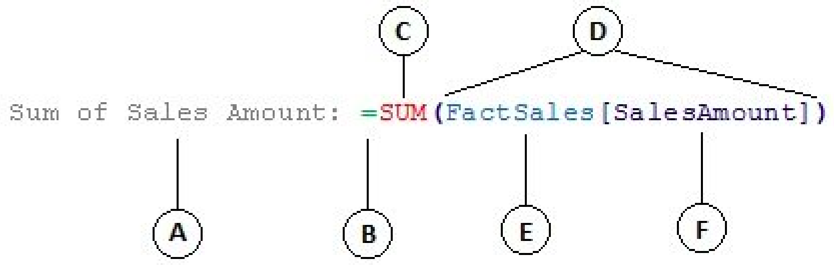


This formula's syntax includes the following elements:

1. The equals sign operator (=) indicates the beginning of the formula, and when this formula is calculated it will return a result or value. All formulas that calculate a value will begin with an equals sign.
2. The referenced column [SalesAmount] contains the values we want to subtract from. A column reference in a formula is always surrounded by brackets []. Unlike Excel formulas which reference a cell, a DAX formula always references a column.
3. The subtraction (-) mathematical operator.
4. The referenced column [TotalCost] contains the values we want to subtract from values in the [SalesAmount] column.

When trying to understand how to read a DAX formula, it is often helpful to break down each of the elements into a language you think and speak every day. For example, you can read this formula as:

In the FactSales table, for each row in the Margin calculated column, calculate (=) a value by subtracting (-) values in the [TotalCost] column from values in the [SalesAmount] column.
Let's take a look at another type of formula, one that is used in a measure:



This formula includes the following syntax elements:

1. The measure name Sum of Sales Amount. Formulas for measures can include the measure name, followed by a colon, followed by the calculation formula.
2. The equals sign operator (=) indicates the beginning of the calculation formula. When calculated, it will return a result.
3. The function SUM adds up all of the numbers in the [SalesAmount] column. You will learn more about functions later.
4. Parenthesis () surround one or more arguments. All functions require at least one argument. An argument passes a value to a function.
5. The referenced table FactSales.
6. The referenced column [SalesAmount] in the FactSales table. With this argument, the SUM function knows on which column to aggregate a SUM.

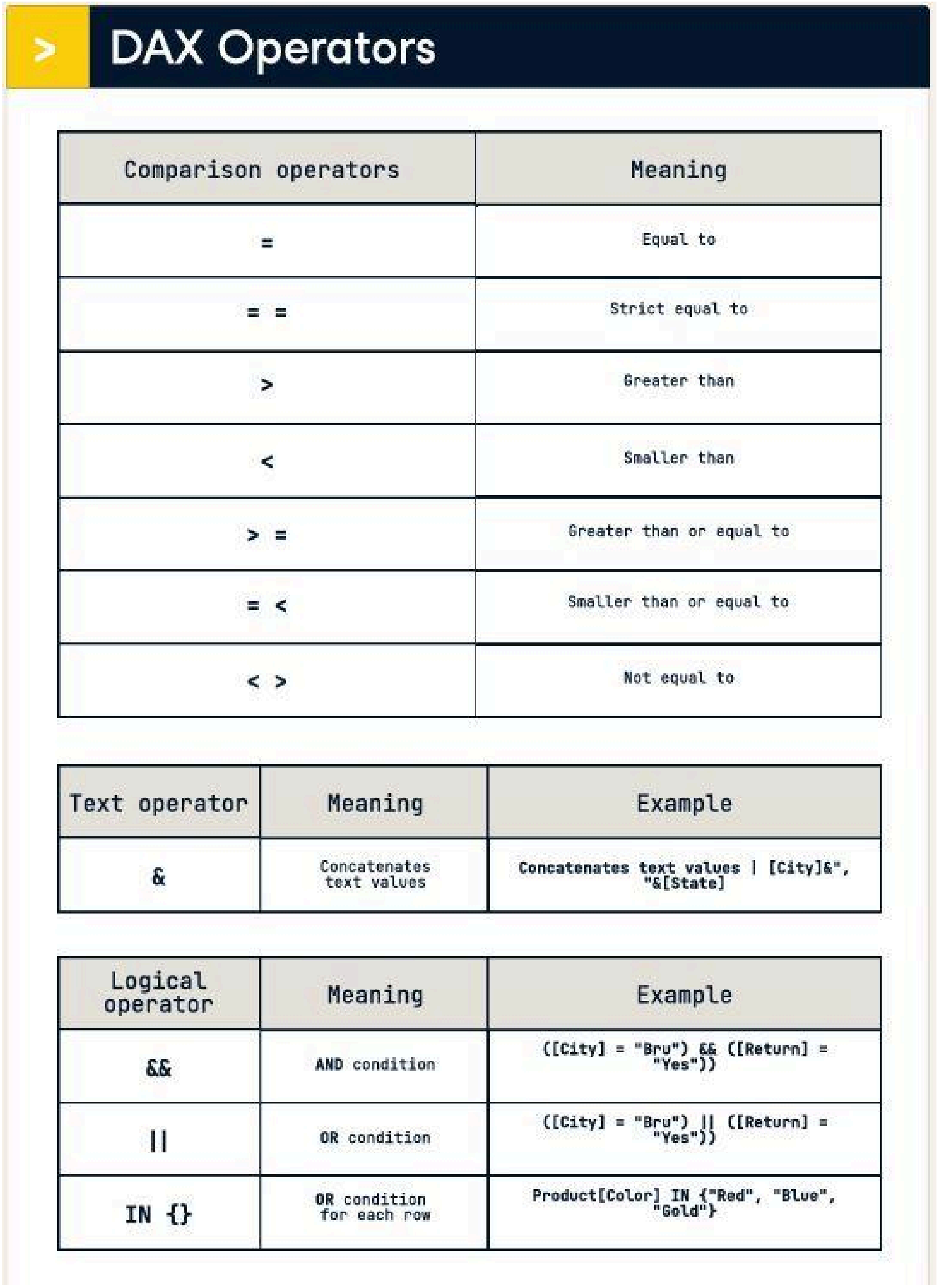
You can read this formula as:

For the measure named Sum of Sales Amount, calculate (=) the SUM of values in the [SalesAmount] column in the FactSales table.

It is very important your formulas have the correct syntax. In most cases, if the syntax is not correct, a syntax error will be returned. (just like the one we had this class)

Now, we have revised the DAX so lets go ahead and apply some of its functions, okay?
Using the **Sample Superstore** file

These are some of the operators that you need to know before moving on:



> DAX Operators

Comparison operators	Meaning
=	Equal to
= =	Strict equal to
>	Greater than
<	Smaller than
> =	Greater than or equal to
= <	Smaller than or equal to
< >	Not equal to

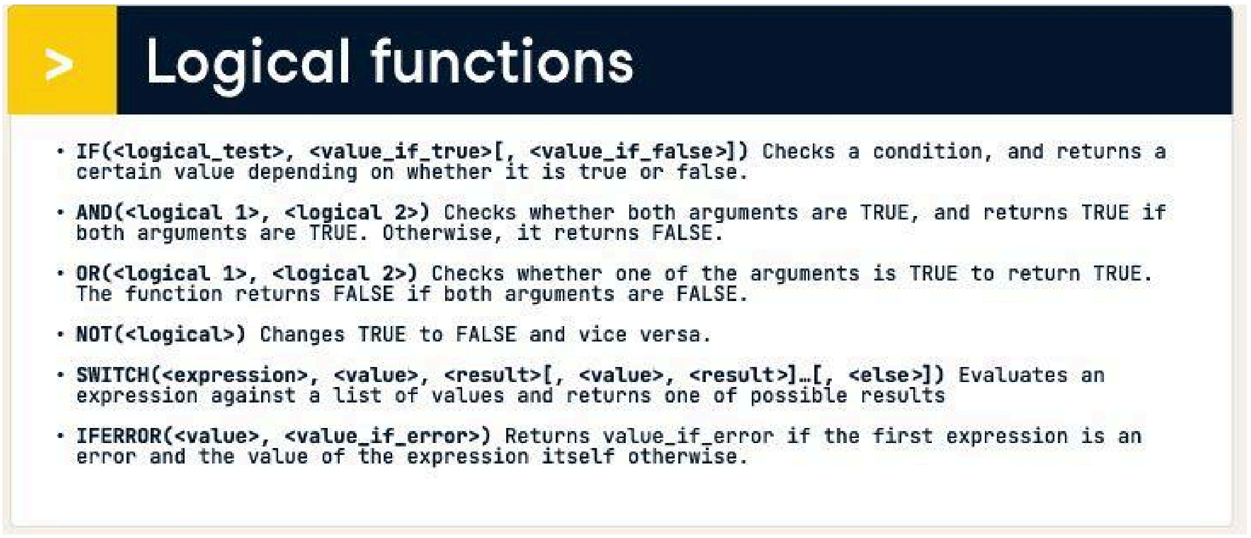
Text operator	Meaning	Example
&	Concatenates text values	Concatenates text values | [City]&", "&[State]

Logical operator	Meaning	Example
&&	AND condition	([City] = "Bru") && ([Return] = "Yes"))
||	OR condition	([City] = "Bru") || ([Return] = "Yes"))
IN {}	OR condition for each row	Product[Color] IN {"Red", "Blue", "Gold"}

TASKS

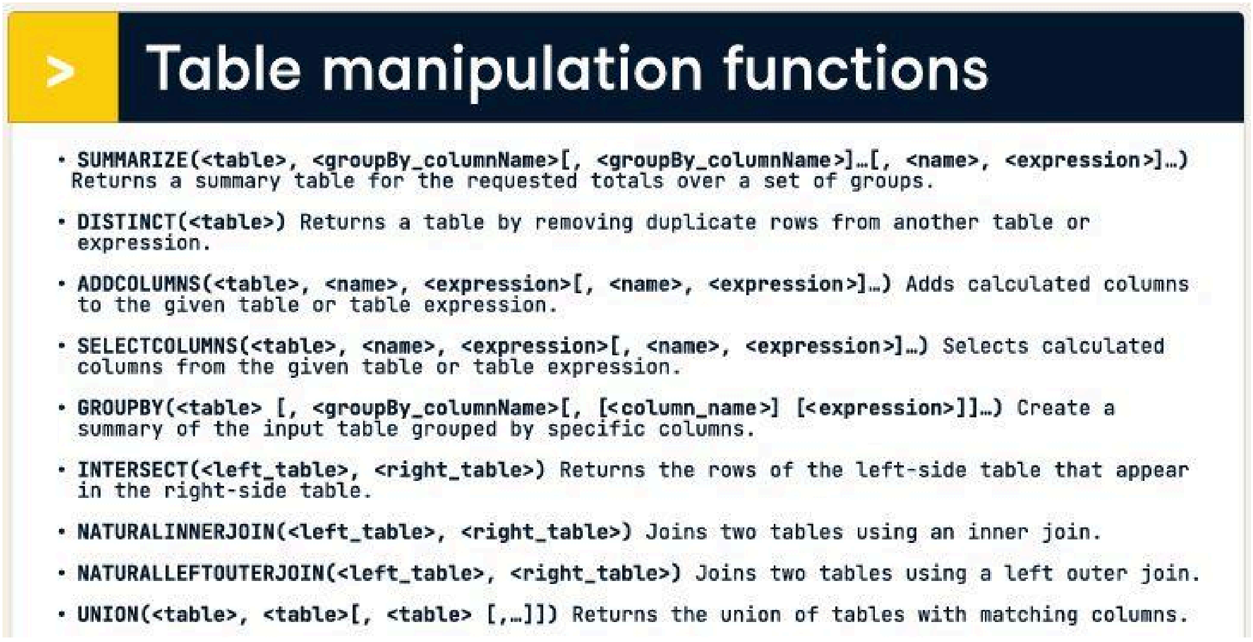
NOTE: You are to explore yourself which function is used to get the answer for each task.

1. Add all the numbers in ‘Quantity Ordered New’
2. Return the average of ‘Sales’
3. Calculate the geometric mean of ‘Profit’
4. Return all the non-blank values of the column ‘Discount’
5. Return the number of total rows in the Returns table
6. Return the ranking of each Sales and make sure it does not skip a rank.
7. Return all the names of cities except Bowling Green
8. Write a DAX expression to create a calculated table showing only orders from the West region.
9. Create a measure called High-Value East Coast Orders that calculates the total sales of orders from the East region with a Sales value greater than $1000. (Hint: Combine CALCULATE, FILTER, and SUM functions)
10. Create a calculated table showing customers who have placed more than 10 orders and have a total Sales exceeding $5000. (Hint: Use FILTER and COUNTROWS functions to filter customers based on the number of orders, and then use CALCULATE and SUM to filter based on total sales.)
11. Return column statistics of every column in the Users table.
12. Calculate the average Sales for all products, ignoring any filters applied on the Product Sub-Category column.(Hint: Use the ALL function to remove filters from the Product Sub-Category column while calculating the average Sales)
13. Develop a measure named Total Profit Margin in Power BI to calculate the overall profit margin as a percentage of total sales. Provide the DAX expression for this measure. (Hint: Use the DIVIDE and SUM functions)
14. In Power BI, create a calculated column named Order Processing Time to determine the number of days it takes to ship an order. Provide the DAX expression used for this calculation. (Hint: Use the DATEDIFF function)
15. Add a new column to the Orders table that categorizes orders as 'High Profit' if the profit is above $100, and 'Low Profit' otherwise. (Use ADDCOLUMNS and IF)



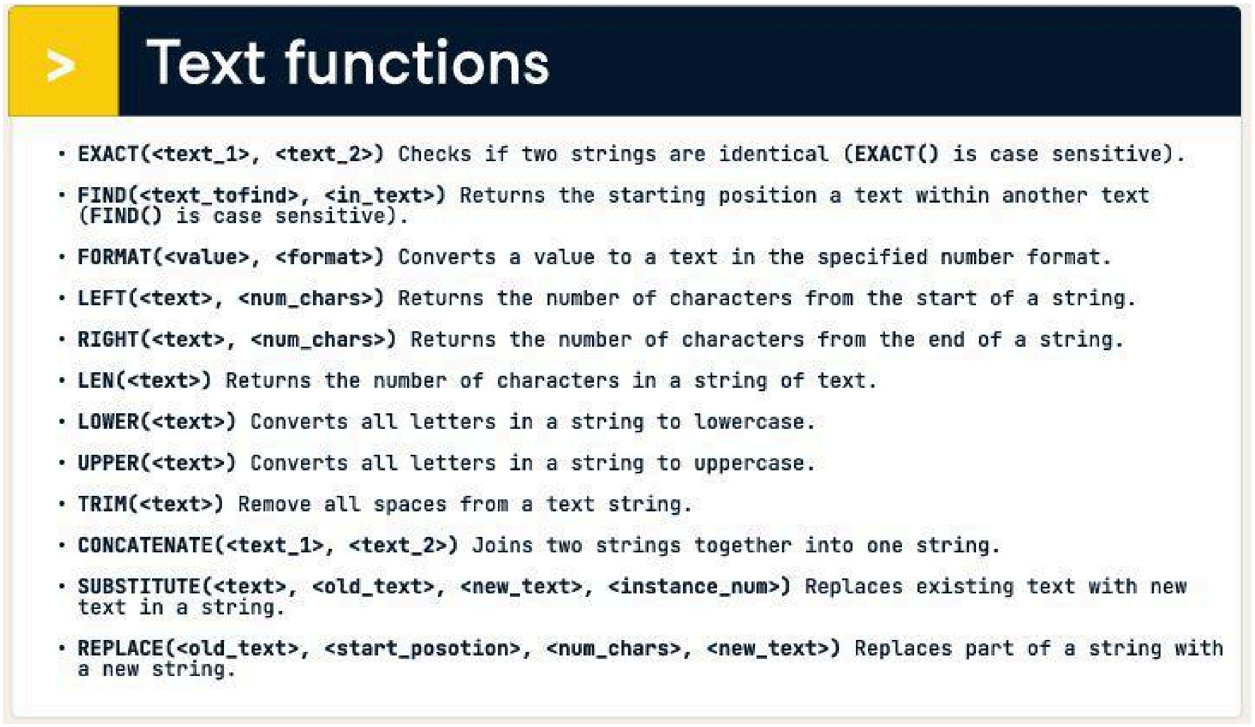
> Logical functions

- **IF(<logical_test>, <value_if_true>[, <value_if_false>])** Checks a condition, and returns a certain value depending on whether it is true or false.
- **AND(<logical 1>, <logical 2>)** Checks whether both arguments are TRUE, and returns TRUE if both arguments are TRUE. Otherwise, it returns FALSE.
- **OR(<logical 1>, <logical 2>)** Checks whether one of the arguments is TRUE to return TRUE. The function returns FALSE if both arguments are FALSE.
- **NOT(<logical>)** Changes TRUE to FALSE and vice versa.
- **SWITCH(<expression>, <value>, <result>[, <value>, <result>]…[, <else>])** Evaluates an expression against a list of values and returns one of possible results
- **IFERROR(<value>, <value_if_error>)** Returns value_if_error if the first expression is an error and the value of the expression itself otherwise.



> Table manipulation functions

- **SUMMARIZE(<table>, <groupBy_columnName>[, <groupBy_columnName>]…[, <name>, <expression>]…)** Returns a summary table for the requested totals over a set of groups.
- **DISTINCT(<table>)** Returns a table by removing duplicate rows from another table or expression.
- **ADDCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]…)** Adds calculated columns to the given table or table expression.
- **SELECTCOLUMNS(<table>, <name>, <expression>[, <name>, <expression>]…)** Selects calculated columns from the given table or table expression.
- **GROUPBY(<table> [, <groupBy_columnName>[, [<column_name>] [<expression>]]…)** Create a summary of the input table grouped by specific columns.
- **INTERSECT(<left_table>, <right_table>)** Returns the rows of the left-side table that appear in the right-side table.
- **NATURALINNERJOIN(<left_table>, <right_table>)** Joins two tables using an inner join.
- **NATURALLEFTOUTERJOIN(<left_table>, <right_table>)** Joins two tables using a left outer join.
- **UNION(<table>, <table>[, <table> [,…]])** Returns the union of tables with matching columns.



Text functions

- EXACT(<text_1>, <text_2>) Checks if two strings are identical (EXACT() is case sensitive).
- FIND(<text_tofind>, <in_text>) Returns the starting position a text within another text (FIND() is case sensitive).
- FORMAT(<value>, <format>) Converts a value to a text in the specified number format.
- LEFT(<text>, <num_chars>) Returns the number of characters from the start of a string.
- RIGHT(<text>, <num_chars>) Returns the number of characters from the end of a string.
- LEN(<text>) Returns the number of characters in a string of text.
- LOWER(<text>) Converts all letters in a string to lowercase.
- UPPER(<text>) Converts all letters in a string to uppercase.
- TRIM(<text>) Remove all spaces from a text string.
- CONCATENATE(<text_1>, <text_2>) Joins two strings together into one string.
- SUBSTITUTE(<text>, <old_text>, <new_text>, <instance_num>) Replaces existing text with new text in a string.
- REPLACE(<old_text>, <start_posotion>, <num_chars>, <new_text>) Replaces part of a string with a new string.

Please view these functions as well. You'll be able to identify what most of them do. If there are some issues or confusion, use the following resources to understand them.

1. Chat GPT
2. Gemini
3. DAX Documentation
4. Files that have been shared with you **(PRACTICE ON THEM)**