**Statistical Functions**

1. **AVERAGE**

* **Syntax:** AVERAGE(column)
* **Why Use:** Calculates the arithmetic mean of a column of numbers.
* **How to Use:** Use when you need to find the average value of a numeric column.
* **Where to Use:** Useful in financial reports, sales data analysis, etc.
* **Example:** Calculate the average sales amount from a SalesAmount column.

AVERAGE(Sales[SalesAmount])

2. **AVERAGEA**

* **Syntax:** AVERAGEA(column)
* **Why Use:** Similar to AVERAGE, but includes text and logical values in the calculation (text = 0, TRUE = 1, FALSE = 0).
* **How to Use:** Use when your data includes non-numeric values and you want to include them in the average calculation.
* **Where to Use:** Surveys or datasets where text and logical values are significant.
* **Example:** Calculate the average score from a column where some entries might be "Pass" or "Fail".

AVERAGEA(Results[Score])

3. **AVERAGEX**

* **Syntax:** AVERAGEX(table, expression)
* **Why Use:** Calculates the average value of an expression evaluated for each row in a table.
* **How to Use:** Use when the average needs to be calculated based on a calculated column or a measure.
* **Where to Use:** Custom metrics, weighted averages, etc.
* **Example:** Calculate the average sales price per unit sold.

AVERAGEX(Sales, Sales[TotalSales] / Sales[Quantity])

4. **COUNT**

* **Syntax:** COUNT(column)
* **Why Use:** Counts the number of non-blank values in a column.
* **How to Use:** Use to count entries in a column, excluding blanks.
* **Where to Use:** Attendance, event counts, etc.
* **Example:** Count the number of products sold.

COUNT(Sales[ProductID])

5. **COUNTA**

* **Syntax:** COUNTA(column)
* **Why Use:** Counts the number of non-blank values, including text and numbers.
* **How to Use:** Use to count all non-empty entries in a column.
* **Where to Use:** Survey responses, customer feedback, etc.
* **Example:** Count the number of entries in a CustomerFeedback column.

COUNTA(Feedback[Comments])

6. **COUNTBLANK**

* **Syntax:** COUNTBLANK(column)
* **Why Use:** Counts the number of blank cells in a column.
* **How to Use:** Use to identify missing data or gaps in your dataset.
* **Where to Use:** Data quality checks, missing information analysis.
* **Example:** Count the number of blank entries in an Email column.

COUNTBLANK(Customers[Email])

7. **COUNTROWS**

* **Syntax:** COUNTROWS(table)
* **Why Use:** Counts the number of rows in a table.
* **How to Use:** Use to get the number of entries in a table or the result of a table expression.
* **Where to Use:** Inventory counts, total records, etc.
* **Example:** Count the number of orders in the Sales table.

COUNTROWS(Sales)

8. **COUNTX**

* **Syntax:** COUNTX(table, expression)
* **Why Use:** Counts the number of rows that contain a number in the result of an expression.
* **How to Use:** Use when you need to apply a filter or condition before counting.
* **Where to Use:** Conditional counting, specific criteria analysis.
* **Example:** Count the number of sales transactions above a certain threshold.

COUNTX(FILTER(Sales, Sales[SalesAmount] > 1000), Sales[SalesAmount])

9. **DISTINCTCOUNT**

* **Syntax:** DISTINCTCOUNT(column)
* **Why Use:** Counts the number of unique values in a column.
* **How to Use:** Use to find the number of distinct entries, removing duplicates.
* **Where to Use:** Unique customer counts, unique product counts, etc.
* **Example:** Count the number of unique customers.

DISTINCTCOUNT(Sales[CustomerID])

10. **DISTINCTCOUNTNOBLANK**

- \*\*Syntax:\*\* `DISTINCTCOUNTNOBLANK(column)`  
- \*\*Why Use:\*\* Counts the number of unique non-blank values in a column.  
- \*\*How to Use:\*\* Similar to `DISTINCTCOUNT`, but excludes blank values.  
- \*\*Where to Use:\*\* Unique identifiers where blanks are not considered.  
- \*\*Example:\*\* Count the number of unique products sold, excluding blanks.  
  
  ```DAX  
  DISTINCTCOUNTNOBLANK(Sales[ProductID])  
  ```

11. **MAX**

- \*\*Syntax:\*\* `MAX(column)`  
- \*\*Why Use:\*\* Returns the largest value in a column.  
- \*\*How to Use:\*\* Use when you need to find the maximum value from a dataset.  
- \*\*Where to Use:\*\* Financial analysis, sales peaks, etc.  
- \*\*Example:\*\* Find the maximum sales amount in a year.  
  
  ```DAX  
  MAX(Sales[SalesAmount])  
  ```

12. **MAXA**

- \*\*Syntax:\*\* `MAXA(column)`  
- \*\*Why Use:\*\* Returns the maximum value in a column, including text (treated as 0) and logical values.  
- \*\*How to Use:\*\* Use when you want to include non-numeric values in your analysis.  
- \*\*Where to Use:\*\* Mixed data types, survey scores, etc.  
- \*\*Example:\*\* Find the maximum score including text entries like "NA".  
  
  ```DAX  
  MAXA(Results[Score])  
  ```

13. **MAXX**

- \*\*Syntax:\*\* `MAXX(table, expression)`  
- \*\*Why Use:\*\* Returns the maximum value of an expression evaluated over a table.  
- \*\*How to Use:\*\* Use for more complex calculations or when the value to be compared is derived.  
- \*\*Where to Use:\*\* Custom calculations, advanced analytics.  
- \*\*Example:\*\* Find the maximum sales per unit price.  
  
  ```DAX  
  MAXX(Sales, Sales[TotalSales] / Sales[Quantity])  
  ```

14. **MIN**

- \*\*Syntax:\*\* `MIN(column)`  
- \*\*Why Use:\*\* Returns the smallest value in a column.  
- \*\*How to Use:\*\* Use to find the minimum value in a dataset.  
- \*\*Where to Use:\*\* Lowest scores, minimum prices, etc.  
- \*\*Example:\*\* Find the minimum sales amount in a year.  
  
  ```DAX  
  MIN(Sales[SalesAmount])  
  ```

15. **MINA**

- \*\*Syntax:\*\* `MINA(column)`  
- \*\*Why Use:\*\* Returns the smallest value in a column, including text (treated as 0) and logical values.  
- \*\*How to Use:\*\* Use when analyzing mixed data types.  
- \*\*Where to Use:\*\* Data quality checks, survey responses.  
- \*\*Example:\*\* Find the minimum value in a score column that includes text like "NA".  
  
  ```DAX  
  MINA(Results[Score])  
  ```

16. **MINX**

- \*\*Syntax:\*\* `MINX(table, expression)`  
- \*\*Why Use:\*\* Returns the minimum value of an expression evaluated over a table.  
- \*\*How to Use:\*\* Use for more complex calculations where the minimum value is derived.  
- \*\*Where to Use:\*\* Financial analysis, detailed metrics.  
- \*\*Example:\*\* Find the minimum sales price per unit.  
  
  ```DAX  
  MINX(Sales, Sales[TotalSales] / Sales[Quantity])  
  ```

17. **PRODUCT**

- \*\*Syntax:\*\* `PRODUCT(column)`  
- \*\*Why Use:\*\* Returns the product of all numbers in a column.  
- \*\*How to Use:\*\* Use when multiplying values in a column.  
- \*\*Where to Use:\*\* Financial calculations, compounding growth.  
- \*\*Example:\*\* Calculate the product of growth rates.  
  
  ```DAX  
  PRODUCT(Growth[Rate])  
  ```

18. **SUM**

- \*\*Syntax:\*\* `SUM(column)`  
- \*\*Why Use:\*\* Adds all numbers in a column.  
- \*\*How to Use:\*\* Use for simple aggregation of numeric data.  
- \*\*Where to Use:\*\* Total sales, total expenses, etc.  
- \*\*Example:\*\* Calculate the total sales amount.  
  
  ```DAX  
  SUM(Sales[SalesAmount])  
  ```

19. **SUMX**

- \*\*Syntax:\*\* `SUMX(table, expression)`  
- \*\*Why Use:\*\* Returns the sum of an expression evaluated for each row in a table.  
- \*\*How to Use:\*\* Use for complex calculations involving expressions.  
- \*\*Where to Use:\*\* Custom metrics, weighted sums.  
- \*\*Example:\*\* Calculate the total sales

considering discounts.

  ```DAX  
  SUMX(Sales, Sales[SalesAmount] - Sales[Discount])  
  ```

### ****Date and Time Functions****

#### 1. ****Date and Time Functions Overview****

Date and time functions in Power BI help you perform operations like creating dates, extracting parts of dates (like year, month, day), and manipulating date values. These functions are crucial for time-based analysis, trend tracking, and creating time-related calculations.

#### 2. ****CALENDAR****

* **Syntax:** CALENDAR(start\_date, end\_date)
* **Why Use:** Creates a table of dates from start\_date to end\_date.
* **How to Use:** Use to create a custom date table, which is often necessary for proper date-based analysis.
* **Where to Use:** Time series analysis, custom fiscal calendars.
* **Example:** Create a date table from January 1, 2020, to December 31, 2020.

CALENDAR(DATE(2020, 1, 1), DATE(2020, 12, 31))

#### 3. ****CALENDARAUTO****

* **Syntax:** CALENDARAUTO([fiscal\_year\_end\_month])
* **Why Use:** Automatically generates a table with a continuous date range covering all dates in the model, optionally adjusting for fiscal year end.
* **How to Use:** Use when you want to create a date table without specifying start and end dates.
* **Where to Use:** Standard calendar setups, quick date table creation.
* **Example:** Create a date table with an automatic date range.

CALENDARAUTO()

#### 4. ****DATE****

* **Syntax:** DATE(year, month, day)
* **Why Use:** Returns the specified date in datetime format.
* **How to Use:** Use to construct a date from individual year, month, and day components.
* **Where to Use:** When you need to create dates dynamically based on other data.
* **Example:** Create a date for July 4, 2021.

DATE(2021, 7, 4)

#### 5. ****DATEDIFF****

* **Syntax:** DATEDIFF(start\_date, end\_date, interval)
* **Why Use:** Returns the difference between two dates in terms of the specified interval (day, month, year, etc.).
* **How to Use:** Use to calculate age, duration, or time between events.
* **Where to Use:** Analysis of customer age, time to complete tasks, etc.
* **Example:** Calculate the number of days between two dates.

DATEDIFF(Order[OrderDate], Order[DeliveryDate], DAY)

#### 6. ****DATEVALUE****

* **Syntax:** DATEVALUE(date\_text)
* **Why Use:** Converts a date in text format to a date value.
* **How to Use:** Use when dates are stored as text and need to be converted for calculations.
* **Where to Use:** Data cleaning, converting imported data.
* **Example:** Convert a text date "2021-07-04" to a date value.

DATEVALUE("2021-07-04")

#### 7. ****DAY****

* **Syntax:** DAY(date)
* **Why Use:** Returns the day of the month as a number from 1 to 31.
* **How to Use:** Use to extract the day component from a date.
* **Where to Use:** When analyzing data on a daily basis.
* **Example:** Get the day part of a date.

DAY(Sales[OrderDate])

#### 8. ****EDATE****

* **Syntax:** EDATE(start\_date, months)
* **Why Use:** Returns the date that is the indicated number of months before or after the start date.
* **How to Use:** Use to calculate expiration dates, future projections, etc.
* **Where to Use:** Subscription services, contract management.
* **Example:** Find the date three months after a start date.

EDATE(Contracts[StartDate], 3)

#### 9. ****EOMONTH****

* **Syntax:** EOMONTH(start\_date, months)
* **Why Use:** Returns the last day of the month before or after a specified number of months.
* **How to Use:** Use for financial reporting, month-end calculations.
* **Where to Use:** Payroll processing, financial closing.
* **Example:** Find the last day of the month three months after a start date.

EOMONTH(Sales[OrderDate], 3)

#### 10. ****HOUR****

- \*\*Syntax:\*\* `HOUR(datetime)`  
- \*\*Why Use:\*\* Returns the hour of a time value, as a number from 0 to 23.  
- \*\*How to Use:\*\* Use to extract the hour component from a datetime.  
- \*\*Where to Use:\*\* Time-based analysis, shift planning.  
- \*\*Example:\*\* Get the hour part of a timestamp.  
  
  ```DAX  
  HOUR(Sales[OrderTime])  
  ```

#### 11. ****MINUTE****

- \*\*Syntax:\*\* `MINUTE(datetime)`  
- \*\*Why Use:\*\* Returns the minute of a time value, as a number from 0 to 59.  
- \*\*How to Use:\*\* Use to extract the minute component from a datetime.  
- \*\*Where to Use:\*\* Detailed time analysis, process timings.  
- \*\*Example:\*\* Get the minute part of a timestamp.  
  
  ```DAX  
  MINUTE(Events[EventTime])  
  ```

#### 12. ****MONTH****

- \*\*Syntax:\*\* `MONTH(date)`  
- \*\*Why Use:\*\* Returns the month as a number from 1 (January) to 12 (December).  
- \*\*How to Use:\*\* Use to extract the month component from a date.  
- \*\*Where to Use:\*\* Seasonal analysis, monthly trends.  
- \*\*Example:\*\* Get the month part of a date.  
  
  ```DAX  
  MONTH(Sales[OrderDate])  
  ```

#### 13. ****NETWORKDAYS****

- \*\*Syntax:\*\* `NETWORKDAYS(start\_date, end\_date, [holidays])`  
- \*\*Why Use:\*\* Returns the number of whole working days between two dates, excluding weekends and specified holidays.  
- \*\*How to Use:\*\* Use for calculating business days, project timelines.  
- \*\*Where to Use:\*\* Project management, employee scheduling.  
- \*\*Example:\*\* Calculate the number of working days between two dates.  
  
  ```DAX  
  NETWORKDAYS(Start[Date], End[Date], Holidays[Date])  
  ```

#### 14. ****NOW****

- \*\*Syntax:\*\* `NOW()`  
- \*\*Why Use:\*\* Returns the current date and time.  
- \*\*How to Use:\*\* Use when real-time data is needed in your analysis.  
- \*\*Where to Use:\*\* Real-time dashboards, tracking systems.  
- \*\*Example:\*\* Get the current date and time.  
  
  ```DAX  
  NOW()  
  ```

#### 15. ****QUARTER****

- \*\*Syntax:\*\* `QUARTER(date)`  
- \*\*Why Use:\*\* Returns the quarter of the year as a number from 1 to 4.  
- \*\*How to Use:\*\* Use to extract the quarter from a date for quarterly analysis.  
- \*\*Where to Use:\*\* Quarterly reporting, trend analysis.  
- \*\*Example:\*\* Get the quarter part of a date.  
  
  ```DAX  
  QUARTER(Sales[OrderDate])  
  ```

#### 16. ****SECOND****

- \*\*Syntax:\*\* `SECOND(datetime)`  
- \*\*Why Use:\*\* Returns the seconds of a time value, as a number from 0 to 59.  
- \*\*How to Use:\*\* Use to extract the second component from a datetime.  
- \*\*Where to Use:\*\* High-resolution time analysis, logging systems.  
- \*\*Example:\*\* Get the second part of a timestamp.  
  
  ```DAX  
  SECOND(Events[EventTime])  
  ```

#### 17. ****TIME****

- \*\*Syntax:\*\* `TIME(hour, minute, second)`  
- \*\*Why Use:\*\* Returns the time as a datetime value.  
- \*\*How to Use:\*\* Use to construct a time from individual hour, minute, and second components.  
- \*\*Where to Use:\*\* When you need to create time values dynamically.  
- \*\*Example:\*\* Create a time for 2:30:00 PM.  
  
  ```DAX  
  TIME(14, 30, 0)  
  ```

#### 18. ****TIMEVALUE****

- \*\*Syntax:\*\* `TIMEVALUE(time\_text)`  
- \*\*Why Use:\*\* Converts a time in text format to a time value.  
- \*\*How to Use:\*\* Use when times are stored as text and need to be converted for calculations.  
- \*\*Where to Use:\*\* Data cleaning, converting imported data.  
- \*\*Example:\*\* Convert a text time "14:30" to a time value.  
  
  ```DAX  
  TIMEVALUE("14:30")  
  ```

#### 19. ****TODAY****

- \*\*Syntax:\*\* `TODAY()`  
- \*\*Why Use:\*\* Returns the current date.  
- \*\*How to Use:\*\* Use when you need to work with the current date.  
- \*\*Where to Use:\*\* Daily reporting, dynamic date comparisons.  
- \*\*

Example:\*\* Get today's date.

  ```DAX  
  TODAY()  
  ```

#### 20. ****WEEKDAY****

- \*\*Syntax:\*\* `WEEKDAY(date, [return\_type])`  
- \*\*Why Use:\*\* Returns the day of the week as a number from 1 (Sunday) to 7 (Saturday), with optional return types.  
- \*\*How to Use:\*\* Use to extract the weekday from a date.  
- \*\*Where to Use:\*\* Weekday-based analysis, staffing schedules.  
- \*\*Example:\*\* Get the weekday of a date, where 1 = Sunday.  
  
  ```DAX  
  WEEKDAY(Sales[OrderDate])  
  ```

#### 21. ****WEEKNUM****

- \*\*Syntax:\*\* `WEEKNUM(date, [return\_type])`  
- \*\*Why Use:\*\* Returns the week number for the year of a date.  
- \*\*How to Use:\*\* Use to extract the week number for weekly analysis.  
- \*\*Where to Use:\*\* Weekly reporting, production schedules.  
- \*\*Example:\*\* Get the week number of a date.  
  
  ```DAX  
  WEEKNUM(Sales[OrderDate])  
  ```

#### 22. ****YEAR****

- \*\*Syntax:\*\* `YEAR(date)`  
- \*\*Why Use:\*\* Returns the year component of a date.  
- \*\*How to Use:\*\* Use to extract the year from a date for yearly analysis.  
- \*\*Where to Use:\*\* Annual reporting, year-over-year comparisons.  
- \*\*Example:\*\* Get the year part of a date.  
  
  ```DAX  
  YEAR(Sales[OrderDate])  
  ```

### ****Filter Functions****

#### 1. ****ALL****

* **Syntax:** ALL([table|column])
* **Why Use:** Removes all filters from the specified table or column(s). It's commonly used to create measures that need to ignore filters, such as in calculating overall totals.
* **How to Use:** Use to reset filters for certain calculations, such as calculating percentages of totals.
* **Where to Use:** Calculations involving overall context, like percentage of total or grand totals.
* **Example:** Calculate the percentage of sales per product category compared to the overall sales.

TotalSales = SUM(Sales[SalesAmount])  
SalesPct = SUM(Sales[SalesAmount]) / CALCULATE(SUM(Sales[SalesAmount]), ALL(Sales))

#### 2. ****ALLEXCEPT****

* **Syntax:** ALLEXCEPT(table, column[, column2, ...])
* **Why Use:** Removes all filters from a table except for the specified columns. It's useful when you want to keep certain filters while removing others.
* **How to Use:** Use to preserve specific filters while clearing others in complex calculations.
* **Where to Use:** Maintaining context for specific dimensions while ignoring others.
* **Example:** Calculate sales while ignoring filters on all columns except ProductCategory.

TotalSales = SUM(Sales[SalesAmount])  
SalesByCategory = CALCULATE(TotalSales, ALLEXCEPT(Sales, Sales[ProductCategory]))

#### 3. ****ALLSELECTED****

* **Syntax:** ALLSELECTED([table|column])
* **Why Use:** Returns all the values in the specified column(s) or table, ignoring filters that have been applied to the table, but honoring all filters applied by the visual or calculation.
* **How to Use:** Use when you need to consider user-applied filters within visuals, but not the default filters.
* **Where to Use:** Dynamic calculations within visuals, interactive reports.
* **Example:** Calculate the percentage of total sales within the context of selected filters.

TotalSales = SUM(Sales[SalesAmount])  
SalesPct = SUM(Sales[SalesAmount]) / CALCULATE(SUM(Sales[SalesAmount]), ALLSELECTED(Sales))

#### 4. ****CALCULATE****

* **Syntax:** CALCULATE(expression, [filter1], [filter2], ...)
* **Why Use:** Evaluates an expression in a modified filter context. It's the most powerful DAX function for applying filters and context modifications.
* **How to Use:** Use to modify the context of calculations, applying specific filters or altering existing ones.
* **Where to Use:** Conditional calculations, complex metrics.
* **Example:** Calculate total sales for a specific region.

SalesForRegion = CALCULATE(SUM(Sales[SalesAmount]), Sales[Region] = "East")

#### 5. ****CALCULATETABLE****

* **Syntax:** CALCULATETABLE(table, [filter1], [filter2], ...)
* **Why Use:** Returns a table with modified filters applied. Similar to CALCULATE, but specifically for tables.
* **How to Use:** Use to create tables that filter data based on specific conditions.
* **Where to Use:** Dynamic table creation, filtered datasets.
* **Example:** Create a table of sales data for a specific year.

Sales2019 = CALCULATETABLE(Sales, YEAR(Sales[OrderDate]) = 2019)

#### 6. ****FILTER****

* **Syntax:** FILTER(table, expression)
* **Why Use:** Returns a table that contains a subset of the original table based on the specified expression.
* **How to Use:** Use to apply row-level filters to a table, generating a new table.
* **Where to Use:** Custom subsets of data, row-level security, detailed analysis.
* **Example:** Filter sales data to include only transactions greater than $1000.

HighValueSales = FILTER(Sales, Sales[SalesAmount] > 1000)

#### ****CONTAINS****

* **Syntax:** CONTAINS(table, column, value)
* **Why Use:** Checks if a specific value exists in a column of a table and returns TRUE or FALSE.
* **How to Use:** Use to validate the presence of data, implement conditional logic, or manage relationships.
* **Where to Use:** Data validation, conditional calculations.
* **Example:** Check if the 'Products' table contains a product with the name "Widget".

HAS\_WIDGET = CONTAINS(Products, Products[ProductName], "Widget")

#### 2. ****ISBLANK****

* **Syntax:** ISBLANK(value)
* **Why Use:** Checks if a value is blank (i.e., null) and returns TRUE if it is, FALSE otherwise.
* **How to Use:** Use to handle missing data or to implement conditional logic based on data presence.
* **Where to Use:** Data cleaning, error handling.
* **Example:** Check if the 'Discount' column in the 'Sales' table has any blank values.

IS\_BLANK\_DISCOUNT = ISBLANK(Sales[Discount])

#### 3. ****ISERROR****

* **Syntax:** ISERROR(value)
* **Why Use:** Checks if an expression results in an error and returns TRUE if it does, FALSE otherwise.
* **How to Use:** Use to handle errors gracefully in your calculations.
* **Where to Use:** Error checking, robust calculation development.
* **Example:** Check if a division results in an error (e.g., division by zero).

HAS\_ERROR = ISERROR(Sales[Revenue] / Sales[UnitsSold])

### ****Logical Functions****

#### 1. ****AND****

* **Syntax:** AND(expression1, expression2)
* **Why Use:** Returns TRUE if both arguments are TRUE; otherwise, returns FALSE.
* **How to Use:** Use to combine multiple conditions that must all be met.
* **Where to Use:** Filtering data, conditional formatting, calculated columns or measures.
* **Example:** Check if sales are greater than $1000 and the region is "East".

IS\_HIGH\_SALE\_EAST = AND(Sales[SalesAmount] > 1000, Sales[Region] = "East")

#### 2. ****FALSE****

* **Syntax:** FALSE()
* **Why Use:** Returns the logical value FALSE.
* **How to Use:** Use to set a false condition explicitly in logical tests.
* **Where to Use:** Initialization, default values, conditional logic.
* **Example:** Use as a default condition in a formula.

DEFAULT\_CONDITION = FALSE()

#### 3. ****IF****

* **Syntax:** IF(condition, result\_true, [result\_false])
* **Why Use:** Returns one value if the condition evaluates to TRUE, and another value if FALSE.
* **How to Use:** Use to apply conditional logic in measures or calculated columns.
* **Where to Use:** Conditional calculations, data categorization.
* **Example:** Apply a discount if sales amount exceeds $1000.

DiscountedPrice = IF(Sales[SalesAmount] > 1000, Sales[SalesAmount] \* 0.9, Sales[SalesAmount])

#### 4. ****IFERROR****

* **Syntax:** IFERROR(value, value\_if\_error)
* **Why Use:** Returns a specified value if the first argument results in an error, otherwise returns the first argument.
* **How to Use:** Use to handle potential errors in expressions, providing a fallback value.
* **Where to Use:** Error handling, ensuring data quality.
* **Example:** Handle division by zero errors gracefully.

SafeDivision = IFERROR(Sales[Revenue] / Sales[UnitsSold], 0)

#### 5. ****NOT****

* **Syntax:** NOT(logical)
* **Why Use:** Reverses the logical value of its argument, returning TRUE for FALSE inputs and vice versa.
* **How to Use:** Use to invert a condition's result.
* **Where to Use:** Conditional logic, data filtering.
* **Example:** Identify sales that are not from the "East" region.

IS\_NOT\_EAST = NOT(Sales[Region] = "East")

#### 6. ****OR****

* **Syntax:** OR(expression1, expression2)
* **Why Use:** Returns TRUE if at least one of the arguments is TRUE; otherwise, returns FALSE.
* **How to Use:** Use to combine conditions where any of them can be met.
* **Where to Use:** Filtering data, creating flags for conditions.
* **Example:** Check if sales are either greater than $1000 or from the "East" region.

IS\_HIGH\_SALE\_OR\_EAST = OR(Sales[SalesAmount] > 1000, Sales[Region] = "East")

#### 7. ****TRUE****

* **Syntax:** TRUE()
* **Why Use:** Returns the logical value TRUE.
* **How to Use:** Use to set a true condition explicitly in logical tests.
* **Where to Use:** Initialization, default values, conditional logic.
* **Example:** Use as a default condition in a formula.

DEFAULT\_CONDITION = TRUE()

### ****Text Functions****

#### 1. ****CONCATENATE****

* **Syntax:** CONCATENATE(text1, text2)
* **Why Use:** Joins two text strings into one.
* **How to Use:** Use to combine values from different columns into one string.
* **Where to Use:** Creating labels, combining names or descriptions.
* **Example:** Concatenate first and last names.

FullName = CONCATENATE(Customer[FirstName], " ", Customer[LastName])

#### 2. ****CONCATENATEX****

* **Syntax:** CONCATENATEX(table, expression, [delimiter])
* **Why Use:** Joins text strings from a table or expression into a single string, separated by a specified delimiter.
* **How to Use:** Use to aggregate text data across multiple rows.
* **Where to Use:** Creating lists, summarizing text information.
* **Example:** Create a comma-separated list of product names.

ProductList = CONCATENATEX(Products, Products[ProductName], ", ")

#### 3. ****FORMAT****

* **Syntax:** FORMAT(value, format\_string)
* **Why Use:** Formats a value as a text string according to the specified format.
* **How to Use:** Use to format numbers, dates, or other values for display.
* **Where to Use:** Data presentation, report aesthetics.
* **Example:** Format a date as "Month Day, Year".

FormattedDate = FORMAT(Sales[OrderDate], "MMMM DD, YYYY")

#### 4. ****LEFT****

* **Syntax:** LEFT(text, num\_chars)
* **Why Use:** Returns the leftmost characters from a text string.
* **How to Use:** Use to extract a portion of a text string.
* **Where to Use:** Parsing text data, data extraction.
* **Example:** Extract the first three letters of a product code.

ProductPrefix = LEFT(Product[ProductCode], 3)

#### 5. ****LEN****

* **Syntax:** LEN(text)
* **Why Use:** Returns the number of characters in a text string.
* **How to Use:** Use to measure the length of a string, often for data validation.
* **Where to Use:** Data validation, text processing.
* **Example:** Get the length of a product description.

DescriptionLength = LEN(Product[Description])

#### 6. ****LOWER****

* **Syntax:** LOWER(text)
* **Why Use:** Converts all characters in a text string to lowercase.
* **How to Use:** Use to standardize text data.
* **Where to Use:** Data normalization, text comparison.
* **Example:** Convert a product name to lowercase.

LowerCaseName = LOWER(Product[ProductName])

#### 7. ****MID****

* **Syntax:** MID(text, start\_num, num\_chars)
* **Why Use:** Returns a specific number of characters from a text string, starting at the specified position.
* **How to Use:** Use to extract a substring from a larger text string.
* **Where to Use:** Parsing text data, data extraction.
* **Example:** Extract the middle part of a product code.

ProductSegment = MID(Product[ProductCode], 4, 3)

#### 8. ****REPLACE****

* **Syntax:** REPLACE(old\_text, start\_num, num\_chars, new\_text)
* **Why Use:** Replaces part of a text string with a different text string.
* **How to Use:** Use to update or correct text data.
* **Where to Use:** Data correction, text manipulation.
* **Example:** Replace a portion of a text with a new value.

UpdatedCode = REPLACE(Product[ProductCode], 1, 3, "NEW")

#### 9. ****RIGHT****

* **Syntax:** RIGHT(text, num\_chars)
* **Why Use:** Returns the rightmost characters from a text string.
* **How to Use:** Use to extract a portion of a text string from the end.
* **Where to Use:** Parsing text data, data extraction.
* **Example:** Extract the last two characters of a product code.

ProductSuffix = RIGHT(Product[ProductCode], 2)

#### 10. ****SUBSTITUTE****

- \*\*Syntax:\*\* `SUBSTITUTE(text, old\_text, new\_text, [instance\_num])`  
- \*\*Why Use:\*\* Replaces occurrences of a specified old text with

### ****Text Functions (continued)****

#### 10. ****SUBSTITUTE****

* **Syntax:** SUBSTITUTE(text, old\_text, new\_text, [instance\_num])
* **Why Use:** Replaces occurrences of a specified substring within a text string with another substring.
* **How to Use:** Use to correct or modify specific parts of a text string.
* **Where to Use:** Data cleansing, text updates.
* **Example:** Replace all occurrences of "Old" with "New" in a product description.

UpdatedDescription = SUBSTITUTE(Product[Description], "Old", "New")

#### 11. ****TRIM****

* **Syntax:** TRIM(text)
* **Why Use:** Removes all leading and trailing spaces from a text string.
* **How to Use:** Use to clean up text data by eliminating unnecessary spaces.
* **Where to Use:** Data cleaning, formatting.
* **Example:** Remove extra spaces from a product name.

CleanName = TRIM(Product[ProductName])

#### 12. ****UPPER****

* **Syntax:** UPPER(text)
* **Why Use:** Converts all characters in a text string to uppercase.
* **How to Use:** Use to standardize text data for uniformity.
* **Where to Use:** Data normalization, text formatting.
* **Example:** Convert a product name to uppercase.

UpperCaseName = UPPER(Product[ProductName])

#### 13. ****VALUE****

* **Syntax:** VALUE(text)
* **Why Use:** Converts a text string that represents a number into a numeric value.
* **How to Use:** Use to perform mathematical operations on text-based numbers.
* **Where to Use:** Converting formatted text numbers into usable numeric values.
* **Example:** Convert a text representation of a number to an actual numeric value.

NumericValue = VALUE(Product[TextNumber])

### ****Time Intelligence Functions****

#### 1. ****DATEADD****

* **Syntax:** DATEADD(date, number\_of\_intervals, interval)
* **Why Use:** Shifts a date by a specified number of intervals, such as days, months, or years.
* **How to Use:** Use to calculate dates that are a certain number of intervals before or after a given date.
* **Where to Use:** Date calculations, trend analysis.
* **Example:** Get the date 30 days before today.

Date30DaysAgo = DATEADD(TODAY(), -30, DAY)

#### 2. ****DATESMTD****

* **Syntax:** DATESMTD(date)
* **Why Use6:** Returns a table of dates for the current month up to the date specified.
* **How to Use:** Use to calculate metrics for the month-to-date period.
* **Where to Use:** Monthly reporting, trend analysis.
* **Example:** Calculate sales for the month-to-date.

MTD\_Sales = CALCULATE(SUM(Sales[SalesAmount]), DATESMTD(Sales[OrderDate]))

#### 3. ****DATESQTD****

* **Syntax:** DATESQTD(date)
* **Why Use:** Returns a table of dates for the current quarter up to the date specified.
* **How to Use:** Use to calculate metrics for the quarter-to-date period.
* **Where to Use:** Quarterly reporting, performance tracking.
* **Example:** Calculate sales for the quarter-to-date.

QTD\_Sales = CALCULATE(SUM(Sales[SalesAmount]), DATESQTD(Sales[OrderDate]))

#### 4. ****DATESYTD****

* **Syntax:** DATESYTD(date)
* **Why Use:** Returns a table of dates for the current year up to the date specified.
* **How to Use:** Use to calculate metrics for the year-to-date period.
* **Where to Use:** Yearly reporting, annual performance analysis.
* **Example:** Calculate sales for the year-to-date.

YTD\_Sales = CALCULATE(SUM(Sales[SalesAmount]), DATESYTD(Sales[OrderDate]))

#### 5. ****SAMEPERIODLASTYEAR****

* **Syntax:** SAMEPERIODLASTYEAR(date)
* **Why Use:** Returns a table of dates for the same period in the previous year.
* **How to Use:** Use to compare current period data with the same period from the previous year.
* **Where to Use:** Year-over-year analysis, trend comparison.
* **Example:** Compare sales with the same period last year.

SalesLastYear = CALCULATE(SUM(Sales[SalesAmount]), SAMEPERIODLASTYEAR(Sales[OrderDate]))

#### 6. ****TOTALMTD****

* **Syntax:** TOTALMTD(expression, date, [filter])
* **Why Use:** Calculates the total of an expression for the month-to-date.
* **How to Use:** Use to aggregate data for the month up to the current date.
* **Where to Use:** Monthly totals, performance tracking.
* **Example:** Calculate total sales for the month-to-date.

TotalMTD\_Sales = TOTALMTD(SUM(Sales[SalesAmount]), Sales[OrderDate])

#### 7. ****TOTALQTD****

* **Syntax:** TOTALQTD(expression, date, [filter])
* **Why Use:** Calculates the total of an expression for the quarter-to-date.
* **How to Use:** Use to aggregate data for the quarter up to the current date.
* **Where to Use:** Quarterly totals, performance tracking.
* **Example:** Calculate total sales for the quarter-to-date.

TotalQTD\_Sales = TOTALQTD(SUM(Sales[SalesAmount]), Sales[OrderDate])

#### 8. ****TOTALYTD****

* **Syntax:** TOTALYTD(expression, date, [filter], [year\_end\_date])
* **Why Use:** Calculates the total of an expression for the year-to-date.
* **How to Use:** Use to aggregate data for the year up to the current date.
* **Where to Use:** Yearly totals, annual performance tracking.
* **Example:** Calculate total sales for the year-to-date.

TotalYTD\_Sales = TOTALYTD(SUM(Sales[SalesAmount]), Sales[OrderDate])

RANKX, SELECTEDVALUE, SUMMARIZE, and TOPN in the same format as your example.

### ****1. RANKX****

* **Syntax:**

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

* **Why Use:** Assigns a **ranking** to each row based on a given expression.
* **How to Use:** Used to rank sales, profits, or any metric within a dataset.
* **Where to Use:** Leaderboards, top customer/product analysis.
* **Example:** Rank customers by total sales.

CustomerRank = RANKX(ALL(Sales[CustomerName]), SUM(Sales[SalesAmount]),, DESC, DENSE)

Customer Rank =

RANKX(

    ALL(Orders[Customer Name]),

    CALCULATE(SUM(Orders[Sales])),

    , DESC, DENSE

)

· SUM(Orders[Sales]) is evaluated in the **current row context**, which means it doesn't properly calculate total sales per customer.

· This may return incorrect or unexpected rankings.

### ****Why Use**** CALCULATE ****in**** RANKX****?****

RANKX evaluates each row **individually**, and we need to ensure that the **total sales per customer** are correctly calculated for ranking.

* If we use SUM(Orders[Sales]) directly, **it might not respect the current row context** inside RANKX.
* CALCULATE(SUM(Orders[Sales])) ensures that we are correctly summing sales **for each customer** while ranking them.

Product Rank by Category =

RANKX(

FILTER(ALL('Superstore Sales'), 'Superstore Sales'[Category] = MAX('Superstore Sales'[Category])),

CALCULATE(SUM('Superstore Sales'[Sales])),

, DESC, DENSE

)

**Region wies rank**

Region Rank = RANKX( ALL('Superstore Sales'[Region]), CALCULATE(SUM('Superstore Sales'[Profit])), , DESC, DENSE )

:

## ****🔹 Why Use**** ALL(Orders[Customer Name]) ****in**** RANKX****?****

ALL(Orders[Customer Name]) **removes any filters applied to the Customer Name column**, ensuring that all customers are ranked based on total sales, not just the ones currently visible in a filter or slicer.

### ****🔍 What Happens If We Don't Use**** ALL****?****

❌ **Without** ALL **(Incorrect Ranking in a Filtered Context)**

DAX

CopyEdit

Customer Rank =

RANKX(

Orders,

CALCULATE(SUM(Orders[Sales])),

, DESC, DENSE

)

🚨 **Problem:**

* If a user selects **only "Region = East"** in a slicer, RANKX will only rank **customers in the East region** rather than all customers.
* The ranking becomes **local to the filtered dataset** instead of considering all customers.

✅ **With** ALL **(Correct Global Ranking)**

DAX

CopyEdit

Customer Rank =

RANKX(

ALL(Orders[Customer Name]),

CALCULATE(SUM(Orders[Sales])),

, DESC, DENSE

)

💡 **Why It Works:**

* ALL(Orders[Customer Name]) **ignores slicers and filters** on the Customer Name column, ensuring that **all customers are ranked globally, not just the visible ones**.
* The ranking remains **consistent even when filtering by region, category, etc.**

### ****📌 Real-Life Example for Students****

Imagine a class of **100 students**, and you're ranking them based on their total marks.

* **Without** ALL → If you filter only "Math Students," their rankings will be based only on **Math students**, not the entire class.
* **With** ALL → Rankings are based on **all 100 students, even if you filter only Math students**.

### ****🔹 What is**** SELECTEDVALUE ****in DAX?****

SELECTEDVALUE is used to **return a single selected value from a column**. If multiple values are selected, it returns an **alternate result (optional)** or BLANK() by default.

### ****📌 Syntax:****

SELECTEDVALUE(<Column>, [AlternateResult])

* <Column> → The column from which to get the selected value.
* [AlternateResult] (optional) → The value to return if multiple values are selected (default is BLANK()).

## ****1️⃣ Simple Example: Getting the Selected Region****

If there is a **slicer** for Orders[Region], we can create a measure:

Selected Region = SELECTEDVALUE(Orders[Region], "Multiple Regions Selected")

✅ **How it Works:**

* If the user selects **one region** (e.g., "East"), it returns "East".
* If the user selects **multiple regions**, it returns "Multiple Regions Selected".
* If nothing is selected, it returns BLANK().

## ****2️⃣ Using**** SELECTEDVALUE ****in Measures (Dynamic Title)****

To create a **dynamic title for a report**, use:

Report Title = "Sales Report for " & SELECTEDVALUE(Orders[Region], "All Regions")

✅ **Now the title changes dynamically** when a region is selected.

## ****3️⃣ Using**** SELECTEDVALUE ****with**** CALCULATE

To filter data based on a **selected value**, use:

Sales for Selected Category =

CALCULATE(

SUM(Orders[Sales]),

Orders[Category] = SELECTEDVALUE(Orders[Category])

)

✅ **Now, when a user selects a category in a slicer, this measure shows total sales for that category.**

## ****4️⃣ When**** SELECTEDVALUE ****Returns**** BLANK()****?****

If nothing or multiple values are selected, SELECTEDVALUE(Orders[Region]) returns BLANK().  
**Solution:** Use IF to handle blank values:

Selected Region Fixed =

IF(

ISBLANK(SELECTEDVALUE(Orders[Region])),

"No Selection",

SELECTEDVALUE(Orders[Region])

)

✅ Now, if nothing is selected, it returns "No Selection".

## ****📌 When to Use**** SELECTEDVALUE****?****

| **Scenario** | **Use SELECTEDVALUE?** | **Alternative If Not** |
| --- | --- | --- |
| **Single-value slicer selection** | ✅ Yes | Use VALUES if expecting multiple values |
| **Dynamic titles based on selection** | ✅ Yes | Use IF(ISFILTERED(...)) for more control |
| **Filter calculations based on selection** | ✅ Yes | Use CALCULATE(FILTER(...)) for multiple values |

🚀 **Would you like a Power BI example with** SELECTEDVALUE**?** 😊

* **Why Use:** Returns the **selected** value from a slicer or single-value column.
* **How to Use:** Used to dynamically filter measures based on user selection.
* **Where to Use:** When working with slicers and dynamic calculations.
* **Example:** Get the **selected region** from a slicer.

Selected\_Region = SELECTEDVALUE(Sales[Region], "No Region Selected")

## # ****Example: Show Total Sales for the Selected Region****

To filter total sales based on **the region selected in a slicer**, use:

DAX

CopyEdit

Sales by Selected Region =

CALCULATE(

SUM('Orders'[Sales]),

'Orders'[Region] = SELECTEDVALUE('Orders'[Region])

)

🔹 **How It Works:**

* If a region is selected (e.g., "West"), it calculates sales only for that region.
* If no region is selected, the measure **returns blank** (to avoid this, use ALL as a fallback).

✅ **Use Case:** Add this measure to a **Table or Card Visual** to show sales dynamically.

====

### ****3. SUMMARIZE --return table****

* **Syntax:**

SUMMARIZE(table, groupBy\_column, [groupBy\_column2, …], [name1, expression1], …)

* SUMMARIZE groups data based on specified columns and allows aggregations like **SUM, COUNT, AVERAGE**, etc.
* **Why Use:** Creates a **grouped summary table** with aggregated values.
* **How to Use:** Used to summarize data with calculated columns.
* **Where to Use:** Custom aggregation tables, drill-through reports.
* **Example:** Create a summary table of **Total Sales per Category**.

SalesSummary = SUMMARIZE(Sales, Sales[Category], "Total Sales", SUM(Sales[SalesAmount]))

### ****Syntax:****

DAX

CopyEdit

SUMMARIZE(

<Table>,

<GroupBy\_Column1>, <GroupBy\_Column2>, ...,

"<New\_Column\_Name>", <Aggregation\_Expression>

)

* <Table>: The dataset (e.g., 'Orders' table).
* <GroupBy\_Column>: Columns to group by (e.g., Category, Region).
* "<New\_Column\_Name>": Name for the aggregated result.
* <Aggregation\_Expression>: DAX function (e.g., SUM(Orders[Sales])).

## ****2️⃣ Example: Summarize Sales by Category****

If you want to summarize **total sales per category**, use:

DAX

CopyEdit

Sales Summary =

SUMMARIZE(

Orders,

Orders[Category],

"Total Sales", SUM(Orders[Sales])

)

🔹 **How It Works:**

* Groups data by **Category**.
* Adds a new column "Total Sales" that calculates SUM(Orders[Sales]).

✅ **Use Case:**

* Use this in **Table Visual** to display total sales per category.

### ****4. TOPN****

* **Syntax:**

TOPN(n, table, expression, [order])

* **Why Use:** Returns the **top N rows** based on a given expression.
* **How to Use:** Used to find top-performing products, customers, or regions.
* **Where to Use:** Leaderboards, top sales/product analysis.
* **Example:** Get the **Top 5 Products by Sales**.

Top5\_Products = TOPN(5, Sales, SUM(Sales[SalesAmount]), DESC)

### ****Using**** TOPN ****in Power BI (DAX)****

The TOPN function in DAX is used to **return the top N rows** of a table based on a specific column's values. It's useful for creating **leaderboards, top-performing products, top customers, etc.**

## ****1️⃣ Syntax of**** TOPN

TOPN(<N>, <Table>, <OrderBy\_Expression>, [Order])

* <N> → The number of top records to return.
* <Table> → The table to retrieve data from.
* <OrderBy\_Expression> → The column used for sorting (e.g., SUM(Orders[Sales])).
* [Order] → Optional, **ASC** (ascending) or **DESC** (descending). Default is DESC.

## ****2️⃣ Using**** TOPN ****in a Measure (Dynamic Top N)****

If you want to get the **Top 5 Customers by Sales**, use:

Top 5 Customers =

TOPN(

5,

SUMMARIZE(Orders, Orders[Customer Name], "Total Sales", SUM(Orders[Sales])),

[Total Sales], DESC

)

🔹 **How It Works:**

* SUMMARIZE(Orders, Orders[Customer Name], "Total Sales", SUM(Orders[Sales])) creates a table with total sales per customer.
* TOPN(5, ..., [Total Sales], DESC) returns the **top 5 customers** based on total sales.

✅ **Use Case:**

* Use this in a **Table Visual** to display the **top 5 customers dynamically**.

## ****3️⃣ Using**** TOPN ****in a Calculated Table****

If you want a **static table** of the **Top 10 Products by Profit**, use:

Top 10 Products Table =

TOPN(

10,

SUMMARIZE(Orders, Orders[Product Name], "Total Profit", SUM(Orders[Profit])),

[Total Profit], DESC

)

✅ **Now this table will always contain the top 10 products based on profit.**

## ****4️⃣ Using**** TOPN ****with a Slicer (Dynamic Top N)****

If you have a slicer where users can select how many top products to display:

### ****Step 1: Create a Parameter for N****

1. **Go to "Modeling" → "New Parameter"**
2. Name it **"Select Top N"**
3. Set a range (e.g., 1 to 20)
4. This will create a table called Select Top N with a column Select Top N[Top N].

### ****Step 2: Create a Dynamic Measure****

Top N Products by Sales =

VAR N = SELECTEDVALUE('Select Top N'[Top N], 5)

RETURN

TOPN(

N,

SUMMARIZE(Orders, Orders[Product Name], "Total Sales", SUM(Orders[Sales])),

[Total Sales], DESC

)

✅ **Now users can control the number of top products dynamically!**

## ****5️⃣ Using**** TOPN ****in RANKX****

If you want to show **only the top 5 products but rank all products**, use:

Product Rank =

RANKX(ALL(Orders[Product Name]), SUM(Orders[Sales]), , DESC, DENSE)

Then, filter your table to show only rows where Product Rank ≤ 5.

## ****📌 Summary: When to Use**** TOPN****?****

| **Feature** | **Can Use TOPN?** | **Alternative If Not** |
| --- | --- | --- |
| **Measure** | ✅ Yes (Dynamic Top N) | Use CALCULATE + RANKX |
| **Calculated Table** | ✅ Yes (Static Top N) | Use SUMMARIZE |
| **Calculated Column** | ❌ No | Use RANKX instead |

🚀 **Would you like a Power BI implementation example with visuals?**

https://radacad.com/write-conditional-statement-using-switch-in-dax-and-power-bi