**POWER-BI DAX ASSIGNMENT**

1. **What the DAX function is performing the operation.**

1. COUNT

Counts the number of non-blank rows in a column.

Typically used to count the number of rows with valid (non-null) entries in a dataset.

2. DISTINCTCOUNT

Counts the distinct (unique) values in a column.

Useful for identifying the number of unique entries, such as distinct customers, products, or transactions.

3. SUM

Sums the values in a column.

Used to add up numerical values such as sales amounts, quantities, or profits.

4. SUMX

Iterates over a table and evaluates an expression for each row, summing the results.

Enables summation of complex expressions across rows, such as calculating total profits for each transaction.

5. AVERAGE

Calculates the average of a column.

Used to find the mean value, such as average sales or average quantities.

6. MIN

Returns the smallest value from a column.

Useful for identifying the minimum numerical or date value, such as the lowest sales amount or earliest date.

7. MAX

Returns the largest value from a column.

Helps identify the maximum value in a dataset, such as highest sales or latest date.

8. SUMMARIZE

Returns a summary table for the requested columns and aggregated data.

Used to group data and apply aggregations like SUM or COUNT.

9. CALCULATE

Modifies the context of a calculation with filters.

Used to calculate results under specific conditions, such as sales for a specific product category.

10. IF

Returns one value if a condition is true, and another if it's false.

Enables conditional calculations, such as classifying sales as "High" or "Low."

11. IFERROR

Returns a specified result if an expression generates an error.

Ensures calculations return valid results even in case of errors (e.g., division by zero).

12. ISBLANK

Checks if a value is blank (null).

Useful for detecting missing or null values in a dataset.

13. EOMONTH

Returns the last date of the month for a specified date.

Commonly used for financial reporting and analysis where month-end dates are required.

14. DATEDIFF

Returns the difference between two dates in the specified interval (days, months, years).

Used to calculate the time difference between two dates, such as between order and shipment dates.

15. DATEADD

Shifts dates by a specified number of intervals (days, months, or years).

Used to create dynamic date ranges, such as moving back or forward a year to compare data.

16. RELATED

Returns a related value from another table.

Purpose: Helps retrieve values from related tables using established relationships.

17. FILTER

Returns a table that has been filtered based on specific criteria.

Purpose: Used to narrow down data and apply calculations only on the filtered results.

18. SWITCH

Evaluates an expression against a list of values and returns the corresponding result.

Provides a simplified way to test multiple conditions, like a case statement.

19. DATESINPERIOD

Returns a range of dates that fall within a specified period from a start date.

Useful for working with time periods, such as calculating sales within the last month or quarter.

20. SELECTEDVALUE

Returns the value selected in a filter or slicer, or blank if multiple values are selected.

Often used in dynamic reports where user input determines the displayed result.

21. ALL

Removes filters from a column or table.

Used to return all values, ignoring any filters applied.

1. VALUES

Retrieves distinct values from a specific column or table, considering the filter context. If there are no rows in the input, it returns an empty table, and if there are no filters, it returns all distinct values.

23. RANKX

Returns the rank of a value in a list of values.

Used to rank items such as customers by sales performance.

24. OFFSET

Shifts the rows in a table by a specified number.

Used to compare values over different periods, such as month-over-month comparisons.

25. USERPRINCIPALNAME

Returns the user principal name for the current user.

Often used in security and personalization scenarios in Power BI.

26. USERELATIONSHIP

Specifies which relationship to use between two tables when multiple relationships exist.

Useful for activating inactive relationships when necessary in calculations.

1. **DAX functions usage**

1. COUNT

Usage: Primarily used in measures to count the number of entries in a specified column. For example, to count the number of sales transactions in a sales table.

2. DISTINCTCOUNT

Usage: Utilized to find the number of unique items in a column. Commonly used in dashboards to track unique customers or products sold.

3. SUM

Usage: Used to calculate total values in numerical columns, such as total revenue or expenses, often in reports and dashboards.

4. SUMX

Usage: Employed for iterating over a table and performing row-wise calculations. Useful when the sum needs to be based on complex calculations rather than direct column values.

5. AVERAGE

Usage: Applied to determine the mean of a set of values, such as average sales per month, and is often displayed in KPI reports.

6. MIN

Usage: Utilized to find the smallest value in a column, useful for identifying the lowest sales or earliest order dates.

7. MAX

Usage: Used to find the maximum value in a column, such as the highest sales figure or the latest transaction date.

8. SUMMARIZE

Usage: Used to create summary tables for analysis, enabling aggregation of data by specific columns, like summarizing sales by region or product category.

9. CALCULATE

Usage: Frequently used in measures to modify the filter context and perform calculations under specific conditions, like calculating sales for a specific region.

10. IF

Usage: Employed to create conditional calculations. Commonly used in financial reports to categorize values based on thresholds.

11. IFERROR

Usage: Used to manage errors in calculations by providing an alternative result, ensuring that reports remain user-friendly and informative.

12. ISBLANK

Usage: Applied to check for empty values in columns, often used to handle missing data in calculations and reports.

13. EOMONTH

Usage: Commonly used in time-based analysis to get the last date of the month, especially in financial reporting for month-end calculations.

14. DATEDIFF

Usage: Used to calculate the time difference between two dates in reports, often employed to measure lead times or response times.

15. DATEADD

Usage: Frequently used in time intelligence calculations to shift dates for comparative analysis, such as year-over-year sales growth.

16. RELATED

Usage: Used in calculated columns to fetch values from related tables, enhancing the richness of data in reports by linking different entities.

17. FILTER

Usage: Employed to filter tables based on conditions, allowing for targeted analysis, like filtering sales records to a specific year.

18. SWITCH

Usage: Used to simplify multiple conditions, replacing nested IF statements, commonly seen in user-defined categorizations in reports.

19. DATESINPERIOD

Usage: Utilized to define dynamic date ranges for analysis, helping to create reports that track performance over specific periods.

20. SELECTEDVALUE

Usage: Commonly used in scenarios where user input via filters is necessary, allowing the report to reflect selected data dynamically.

21. ALL

Usage: Used to remove filters from a column or table for overall calculations, such as calculating total sales regardless of filters applied.

1. VALUES

Usage: When you want to retrieve the unique values of a specific field from your data. For instance, if you are dealing with a table of customer orders, you can use VALUES to list all distinct customer IDs or product categories.

23. RANKX

Usage: Employed to rank items based on a specified measure, useful for performance analysis such as ranking sales representatives.

24. OFFSET

Usage: Used to compare data across different periods, like comparing current month sales against the previous month.

25. USERPRINCIPALNAME

Usage: Utilized for personalizing reports based on user identity, ensuring that users see data relevant to them.

26. USERELATIONSHIP

Usage: Used when there are multiple relationships between tables to specify which relationship to use for calculations.

1. **A scenario where and how it is used.**

1. COUNT

Scenario: In a sales dashboard, you might want to count the total number of orders placed. Using COUNT(Order[OrderID]) helps to understand the volume of sales.

2. DISTINCTCOUNT

Scenario: In a customer analysis report, you could use DISTINCTCOUNT(Customer[CustomerID]) to determine how many unique customers made purchases in a specific period, helping in customer segmentation.

3. SUM

Scenario: A financial report may require the total revenue generated in the last quarter. You can use SUM(Sales[TotalRevenue]) to provide this total, aiding in financial forecasting.

4. SUMX

Scenario: In a performance analysis report, you might want to calculate total sales revenue considering discounts. Using SUMX(Sales, Sales[Quantity] \* Sales[Price] \* (1 - Sales[Discount])) allows for a detailed revenue calculation.

5. AVERAGE

Scenario: For a monthly performance review, you may need to calculate the average sales per day. AVERAGE(Sales[TotalSales]) gives insights into daily sales performance.

6. MIN

Scenario: In a project management report, you might want to find the earliest start date of all projects. Using MIN(Project[StartDate]) helps identify the first project kickoff date.

7. MAX

Scenario: In a product performance review, finding the highest sales value can be crucial. MAX(Sales[TotalSales]) identifies the product that achieved the highest sales.

8. SUMMARIZE

Scenario: In a summary report for management, you might want to aggregate sales by product category. SUMMARIZE(Sales, Products[Category], "TotalSales", SUM(Sales[TotalRevenue])) creates a summary table for easy analysis.

9. CALCULATE

Scenario: If you want to calculate sales figures for a specific region, you can use CALCULATE(SUM(Sales[TotalSales]), Sales[Region] = "North"), enabling targeted analysis of regional performance.

10. IF

Scenario: To categorize products based on sales performance, you might use IF(Sales[TotalSales] > 10000, "High", "Low"), helping to quickly identify top-performing products.

11. IFERROR

Scenario: In a financial report, you might encounter division errors. Using IFERROR(Sales[TotalSales]/Sales[TotalOrders], 0) allows you to handle potential errors gracefully.

12. ISBLANK

Scenario: If you want to identify customers without a registered email, ISBLANK(Customer[Email]) can help in auditing and enhancing customer records.

13. EOMONTH

Scenario: In a financial report, you might need to find the end of the current month to evaluate month-end closing. EOMONTH(TODAY(), 0) gives you the end date for closing procedures.

14. DATEDIFF

Scenario: For tracking project timelines, you might want to calculate the number of days between the project start and end dates. DATEDIFF(Project[StartDate], Project[EndDate], DAY) provides the total duration.

15. DATEADD

Scenario: To analyze year-over-year sales growth, you might use DATEADD(Sales[Date], -1, YEAR) to shift your date context back by one year for comparative analysis.

16. RELATED

Scenario: In a sales report, you might want to include customer information, such as RELATED(Customer[CustomerName]), which retrieves the name of the customer associated with a sale.

17. FILTER

Scenario: In a sales performance dashboard, you might need to filter for sales in a specific year. Using FILTER(Sales, Sales[Year] = 2023) allows you to analyze only the relevant data.

18. SWITCH

Scenario: To categorize sales performance based on ranges, you might use SWITCH(TRUE(), Sales[TotalSales] > 10000, "High", Sales[TotalSales] > 5000, "Medium", "Low"), simplifying categorization.

19. DATESINPERIOD

Scenario: In a financial report, if you want to analyze sales over the last three months, you can use DATESINPERIOD(Sales[Date], LASTDATE(Sales[Date]), -3, MONTH) to define the timeframe.

20. SELECTEDVALUE

Scenario: When creating a report where users select a product category, SELECTEDVALUE(Products[Category]) allows the report to reflect only the selected category, providing a personalized experience.

21. ALL

Scenario: To calculate the percentage of total sales regardless of filters, you might use CALCULATE(SUM(Sales[TotalSales]), ALL(Sales)) to remove filters and provide a comprehensive view.

1. VALUE

Scenario: · You can use the VALUES function to pull in distinct values from one table that can be used in related calculations in another. When you want to work with distinct values in a report or calculation.

23. RANKX

Scenario: In a sales performance report, you might want to rank sales representatives based on their total sales. RANKX(ALL(SalesRep), SalesRep[TotalSales]) helps in identifying top performers.

24. OFFSET

Scenario: To compare current month sales to the previous month, you might implement an offset calculation using Sales[TotalSales] - OFFSET(Sales[TotalSales], -1, MONTH) to find the difference.

25. USERPRINCIPALNAME

Scenario: To personalize reports, you might want to show data relevant to the logged-in user. USERPRINCIPALNAME() helps to filter data based on the user's identity.

26. USERELATIONSHIP

Scenario: In a scenario with multiple relationships, you might want to calculate sales based on a specific relationship. CALCULATE(SUM(Sales[TotalSales]), USERELATIONSHIP(Sales[CustomerID], Customer[CustomerID])) allows targeted calculations.

1. **DAX expression for the above scenario.**

1. COUNT

TotalOrders = COUNT(Order[OrderID])

2. DISTINCTCOUNT

UniqueCustomers = DISTINCTCOUNT(Customer[CustomerID])

3. SUM

TotalRevenue = SUM(Sales[TotalRevenue])

4. SUMX

TotalRevenueAfterDiscount = SUMX(Sales, Sales[Quantity] \* Sales[Price] \* (1 - Sales[Discount]))

5. AVERAGE

AverageDailySales = AVERAGE(Sales[TotalSales])

6. MIN

EarliestStartDate = MIN(Project[StartDate])

7. MAX

HighestSales = MAX(Sales[TotalSales])

8. SUMMARIZE

SalesSummary = SUMMARIZE(Sales, Products[Category], "TotalSales", SUM(Sales[TotalRevenue]))

9. CALCULATE

NorthRegionSales = CALCULATE(SUM(Sales[TotalSales]), Sales[Region] = "North")

10. IF

SalesCategory = IF(Sales[TotalSales] > 10000, "High", "Low")

11. IFERROR

SafeSalesPerOrder = IFERROR(Sales[TotalSales] / Sales[TotalOrders], 0)

12. ISBLANK

MissingEmailCount=COUNTROWS(FILTER(Customer,ISBLANK(Customer[Email])))

13. EOMONTH

MonthEndDate = EOMONTH(TODAY(), 0)

14. DATEDIFF

ProjectDuration = DATEDIFF(Project[StartDate], Project[EndDate], DAY)

15. DATEADD

LastYearSales = CALCULATE(SUM(Sales[TotalSales]), DATEADD(Sales[Date], -1, YEAR))

16. RELATED

CustomerName = RELATED(Customer[CustomerName])

17. FILTER

SalesIn2023 = FILTER(Sales, Sales[Year] = 2023)

18. SWITCH

SalesPerformance = SWITCH(TRUE(),

Sales[TotalSales] > 10000, "High",

Sales[TotalSales] > 5000, "Medium",

"Low")

19. DATESINPERIOD

LastThreeMonthsSales=CALCULATE(SUM(Sales[TotalSales]), DATESINPERIOD (Sales[Date], LASTDATE(Sales[Date]), -3, MONTH))

20. SELECTEDVALUE

SelectedCategory = SELECTEDVALUE(Products[Category])

21. ALL

TotalSalesAll = CALCULATE(SUM(Sales[TotalSales]), ALL(Sales))

1. VALUE

ProductList = VALUES(Sales[ProductName])

23. RANKX

SalesRank = RANKX(ALL(SalesRep), SalesRep[TotalSales])

24. OFFSET

SalesDifference = Sales[TotalSales] - OFFSET(Sales[TotalSales], -1, MONTH)

25. USERPRINCIPALNAME

CurrentUser = USERPRINCIPALNAME()

26. USERELATIONSHIP

TargetedSales=CALCULATE(SUM(Sales[TotalSales]),USERELATIONSHIP (Sales[CustomerID], Customer[CustomerID]))