**UNDERSTANDING DAX IN POWER BI**

1. **Write a DAX formula to create a measure that calculates the total sales amount from the 'order\_details' table, where the 'quantity' column is multiplied by the 'price' column from the 'pizzas' table.**

Ans: Total Sales Amount = SUMX('order\_details', 'order\_details'[quantity] \* RELATED('pizzas'[price]))

**2. Create a measure to calculate the average order value from the 'order\_details' and 'orders' tables.**

Ans : Average Order Value = AVERAGEX( 'orders', CALCULATE( SUMX('order\_details', 'order\_details'[quantity] \* RELATED('pizzas'[price]))))

**3. How would you create a measure to calculate the total number of pizzas sold from the 'order\_details' table?**

Ans : Total Pizzas Sold = SUM('order\_details'[quantity])

**5. Describe filter context and explain how it affects the calculation of a measure that sums the total quantity of pizzas ordered in the 'order\_details' table for a specific date range in the 'orders' table.**

Ans : Filter context in Power BI refers to the set of filters that are applied to data when a calculation is made. These filters can come from various sources, such as slicers, visuals, or other filters in the report.

When calculating a measure that sums the total quantity of pizzas ordered in the 'order\_details' table for a specific date range in the 'orders' table, the filter context affects which rows from these tables are included in the calculation.

For example, if a date range filter is applied from the 'orders' table, only the rows from the 'orders' table that fall within this date range are considered. Consequently, the related rows in the 'order\_details' table are also filtered accordingly. The measure then sums the 'quantity' column from these filtered rows, resulting in the total quantity of pizzas ordered within the specified date range.

**6. Explain how row context is applied when creating a calculated column that multiplies 'quantity' by 'price' in the 'order\_details' table.**

Ans : Row context in Power BI refers to the context of a single row when performing calculations in calculated columns or calculated tables. When creating a calculated column, each row is evaluated individually.

To create a calculated column that multiplies 'quantity' by 'price' in the 'order\_details' table, the calculation is performed row by row. For each row in the 'order\_details' table, Power BI applies the row context to access the 'quantity' and 'price' values of that specific row and performs the multiplication.

TotalPrice = order\_details[quantity] \* order\_details[price]

In this formula, for each row, the 'quantity' and 'price' values are retrieved, multiplied, and the result is stored in the 'TotalPrice' column. This operation is repeated for each row in the 'order\_details' table.

**7. How does the CALCULATE function change the filter context of a measure? Provide an example using the dataset.**

**Ans :** The CALCULATE function in Power BI modifies the filter context of a measure by applying additional filters or altering existing ones. It evaluates an expression in a context that is modified by the specified filters.

**Example**

To calculate the total sales for a specific pizza type

**TotalSalesForVegetarian= CALCULATE(SUMX(order\_details,order\_details[quantity] \* order\_details[price]), pizza[pizza\_type] = "Vegeterian")**

Here, CALCULATE changes the filter context to include only rows where the pizza type is "Vegeterian".

**8. Describe a scenario where you would use the ALLEXCEPT function in your DAX calculations**.

Ans: The ALLEXCEPT function in DAX is typically used in calculations where you need to remove all filters except for specific ones. This function is particularly useful in scenarios where you want to maintain certain filters while ignoring others. Here are a few common types of calculations where ALLEXCEPT can be applied:

### SCENERIO : ****Calculating Totals While Keeping Specific Filters****

You can use ALLEXCEPT to calculate totals or sums while keeping filters on specific columns, such as categories or product types. This is useful when you want to see the overall performance of a category regardless of other filters applied in the report.

For instance

Calculate the total sales for each pizza type, ignoring other filters

TotalSalesByPizzaType = CALCULATE( [TotalSales], ALLEXCEPT(pizza\_type, pizza\_type[type\_name]))

9**. Name and describe the purpose of at least three common DAX functions that you would use to perform calculations or data manipulations on the 'orders' and 'order\_details' tables.**

Ans : Here are three common DAX functions that can be used for calculations or data manipulations on the 'orders' and 'order\_details' tables:

1. SUMX : To calculate the sum of an expression evaluated over a table.

2. CALCULATE: To modify the filter context of an expression.

3. RELATED : To fetch related values from another table.

**10. How does the RELATED function work and when would you use it with the 'order\_details' and 'pizzas' tables?**

Ans: The RELATED function helps to pull information from one table into another when there is a relationship between them. For example, if you have an order\_details table with a list of pizzas ordered and a pizzas table that has the names of those pizzas, you can use RELATED to get the pizza names into the order\_details table. This function looks up the pizza name in the pizzas table based on the pizza\_id and brings that name into the order\_details table. You'd use it when you need to add more details from a related table to make your data more complete and understandable.

11. Write a DAX formula to create a measure that counts the number of distinct orders in the 'orders' table.

Ans: Distinct Order Count = DISTINCTCOUNT('orders'[order\_id])

**12. What is the difference between the SUM and SUMX functions in DAX? Provide an example of each using the dataset.**

Ans : The `SUM` function adds up values in a single column, while `SUMX` performs a calculation on each row and then sums the results.

Example with `order\_details` Table

SUM Example: TotalQuantity = SUM(order\_details[quantity])

SUMX Example: TotalSales = SUMX(order\_details, order\_details[quantity] \* order\_details[price])

**13. What are time intelligence functions in DAX and why are they important?**

Ans: Time intelligence functions in DAX allow you to analyze data based on dates, such as calculating year-to-date totals or comparing periods. They are important for understanding trends over time and making time-based comparisons.

Examples include TOTALYTD, SAMEPERIODLASTYEAR, and DATEADD.

**14. Explain how the TOTALYTD function works and provide a DAX example using it to calculate the year-to-date total sales amount from the 'order\_details' and 'orders' tables.**

Ans: The `TOTALYTD` function calculates the total for a measure from the start of the year up to a specified date. It helps in summarizing year-to-date figures.

Example

To calculate the year-to-date total sales amount:

YTDTotalSales = TOTALYTD( [TotalSales], orders[order\_date] )

In this example, `TOTALYTD` sums the `TotalSales` measure from the beginning of the year up to the date specified in `orders[order\_date]`.