SQL to DAX Query to DAX Measure

SQL and DAX serve similar purposes in querying and manipulating data, but they have different syntax and concepts. Here are some equivalents between SQL and DAX. By identifying the key calculations and ensuring the result is scalar, you can effectively convert DAX queries to measures.

SQL Query

1. Query all rows and columns from a table

```
SELECT *
FROM FactSales;
```

2. Query only 100 rows from a table

```
SELECT *
FROM FactSales
LIMIT 100;
```

3. Sort the results by SalesKey in Ascending order

```
SELECT *
FROM FactSales
ORDER BY SalesKey ASC;
```

4. Query only a few columns from a table

```
SELECT
  BrandName,
  ProductName
FROM DimProduct;
```

DAX Query

1. Query all rows and columns from a table

```
EVALUATE
CALCULATETABLE ( FactSales )
```

2. Query only 100 rows from a table

```
EVALUATE
TOPN ( 100, FactSales )
```

3. Sort the results by SalesKey in Ascending order

```
EVALUATE
CALCULATETABLE ( FactSales )
ORDER BY FactSales[SalesKey] ASC
```

4. Query only a few columns from a table

```
EVALUATE
SELECTCOLUMNS (
    DimProduct,
    DimProduct[BrandName],
    DimProduct[ProductName]
 )
```

DAX Measure

A measure is a calculation used to aggregate data dynamically. Unlike SQL and DAX queries, which can return multiple rows and columns, a measure provides a single scalar value. Measures in Power BI require an aggregation function, such as SUM, AVERAGE, COUNT, or others, to perform their calculations and provide meaningful insights based on the report's context or visualization.

SQL to DAX Query to DAX Measure

SQL Query

5. **Query unique rows from a table**

```sql
SELECT DISTINCT *
FROM DimProduct;
```

6. **Query unique rows from a column of a table**

```sql
SELECT
  DISTINCT BrandName
FROM DimProduct;
```

7. **Query a table and filter row with a condition**

```sql
SELECT *
FROM DimProduct
WHERE
  BrandName = "Contoso";
```

8. **Query a table and filter row with a wildcard**

```sql
SELECT *
FROM DimProduct
WHERE BrandName LIKE "Con%";
```

DAX Query

5. **Query unique rows from a table**

```
EVALUATE
DISTINCT ( DimProduct )
```

6. **Query unique rows from a column of a table**

```
EVALUATE
DISTINCT ( DimProduct[BrandName] )
```

7. **Query a table and filter row with a condition**

```
EVALUATE
FILTER (
    DimProduct,
    DimProduct[BrandName] =
"Contoso"
)
```

8. **Query a table and filter row with a wildcard**

```
EVALUATE
FILTER (
    DimProduct,
    CONTAINSSTRING(
DimProduct[BrandName], "Con?" )
)
```

DAX Measure

A measure is a calculation used to aggregate data dynamically. Unlike SQL and DAX queries, which can return multiple rows and columns, a measure provides a single scalar value. Measures in Power BI require an aggregation function, such as SUM, AVERAGE, COUNT, or others, to perform their calculations and provide meaningful insights based on the report's context or visualization.

SQL to DAX Query to DAX Measure

SQL Query

9. **Query a table having missing values**

```
SELECT *
FROM DimGeography
WHERE CityName IS NOT NULL;
```

10. **Query a table with values from a list**

```
SELECT *
FROM FactSales
WHERE SalesKey IN (100, 200,
300);
```

11. **Count the number of rows in a table**

```
SELECT
  COUNT(*) AS Total_Rows
FROM FactSales;
```

12. **Count the unique values in a column**

```
SELECT
  COUNT(DISTINCT BrandName) AS
Total_Brands
FROM DimProduct;
```

DAX Query

9. **Query a table having missing values**

```
EVALUATE
FILTER ( DimGeography,
    NOT ISBLANK (
DimGeography[CityName] ))
```

10. **Query a table with values from a list**

```
EVALUATE
FILTER ( FactSales,
    FactSales[SalesKey]
IN {100, 200, 300})
```

11. **Count the number of rows in a table**

```
EVALUATE
ROW ( "Total_Rows", COUNTROWS (
FactSales ) )
```

12. **Count the total values in a column**

```
EVALUATE
ROW ( "Total_Brands",
    DISTINCTCOUNT (
DimProduct[BrandName] )
)
```

DAX Measure

- **Count the number of rows in a table**

```
Total Rows Sales Table =
COUNTROWS ( FactSales )
```

- **Count the total values in a column**

```
Total_Brands =
DISTINCTCOUNT (
DimProduct[BrandName] )
```

SQL to DAX Query to DAX Measure

SQL Query

13. **Aggregate values in a column with a condition**

```
SELECT
  SUM(SalesAmount) AS
2008_SalesAmount
FROM FactSales
WHERE YEAR(DateKey) = 2008;
```

14. **Cross-join multiple tables**

```
SELECT
  g.CityName,
  st.SalesTerritoryCountry
FROM
  DimGeography g
  CROSS JOIN DimSalesTerritory st;
```

DAX Query

13. **Aggregate values in a column with a condition**

```
EVALUATE
ROW ( "2008_SalesAmount",
  CALCULATE (
    SUM ( FactSales[SalesAmount]
),
    KEEPFILTERS( YEAR(
FactSales[DateKey] ) = 2008 )
     )
)
```

14. **Cross-join multiple tables**

```
EVALUATE
CROSSJOIN (
    SELECTCOLUMNS (
DimGeography,
DimGeography[CityName] ),
    SELECTCOLUMNS (
DimSalesTerritory,
DimSalesTerritory[SalesTerritory
Country] )
)
```

DAX Measure

- **Aggregate values in a column with a condition**

```
2008_SalesAmount =
CALCULATE (
    SUM ( FactSales[SalesAmount] ),
    KEEPFILTERS ( YEAR (
FactSales[DateKey] ) = 2008)
)
```

SQL to DAX Query to DAX Measure

SQL Query

15. **Query multiple tables with inner join**

```
SELECT
  c.ProductCategoryName,
  sc.ProductSubCategoryName,
  p.ProductName
FROM DimProduct p
  INNER JOIN DimProductSubcategory sc ON p.ProductSubCategoryKey = sc.ProductSubCategoryKey
  INNER JOIN DimProductCategory c ON sc.ProductCategoryKey = c.ProductCategoryKey;
```

16. **Query multiple tables with left join**

```
SELECT
  g.GeographyKey,
  g.CityName,
  g.RegionCountryName,
  st.SalesTerritoryCountry
FROM
  DimGeography g
  LEFT OUTER JOIN DimSalesTerritory st ON g.GeographyKey = st.GeographyKey;
```

DAX Query

15. **Query multiple tables with inner join**

```
EVALUATE
SELECTCOLUMNS ( DimProduct,
    "ProductCategoryName", RELATED ( DimProductCategory[ProductCategoryName] ),
    "ProductSubCategoryName", RELATED ( DimProductSubcategory[ProductSubcategoryName] ),
    DimProduct[ProductName]
)
```

16. **Query multiple tables with left join**

```
EVALUATE
NATURALLEFTOUTERJOIN (
  SELECTCOLUMNS ( DimGeography,
     "GeographyKey", CONVERT ( DimGeography[GeographyKey], INTEGER ),
     DimGeography[CityName],
     DimGeography[RegionCountryName] ),
  SELECTCOLUMNS ( DimSalesTerritory,
     "GeographyKey", CONVERT ( DimSalesTerritory[GeographyKey], INTEGER ),
     DimSalesTerritory[SalesTerritoryCountry]
    )
)
```

DAX Measure

We can effectively convert a DAX query into a measure by identifying the main calculation, ensuring the result is a scalar value, and using appropriate DAX functions. This allows the measure to be dynamically calculated based on the report's filter context.

The following is an example of converting a DAX query into a DAX measure.

```
Count Unique Products =
VAR _Product =
    SELECTCOLUMNS (
        DimProduct,
        "ProductCategoryName", RELATED ( DimProductCategory[ProductCategoryName] ),
        "ProductSubCategoryName", RELATED ( DimProductSubcategory[ProductSubcategoryName] ),
        DimProduct[ProductName]
    )
RETURN
    COUNTX (
        SUMMARIZE ( _Product, DimProduct[ProductName] ),
        DimProduct[ProductName]
    )
```

SQL to DAX Query to DAX Measure

SQL Query

17. **Combine two tables while removing duplicates**

```
SELECT
  RegionCountryName AS Country
FROM
  DimGeography
UNION
SELECT
  SalesTerritoryCountry AS Country
FROM
  DimSalesTerritory;
```

18. **Show results that appear in the first table, but not in the second**

```
SELECT
  RegionCountryName
FROM
  DimGeography
EXCEPT
SELECT
  SalesTerritoryCountry
FROM
  DimSalesTerritory;
```

DAX Query

17. **Combine two tables while removing duplicates**

```
EVALUATE
DISTINCT (
    UNION (
        SELECTCOLUMNS( DimGeography, "Country", DimGeography[RegionCountryName] ),
        SELECTCOLUMNS( DimSalesTerritory, "Country", DimSalesTerritory[SalesTerritoryCountry] )
    )
)
```

18. **Show results that appear in the first table, but not in the second**

```
EVALUATE
DISTINCT (
    EXCEPT (
        VALUES ( DimGeography[RegionCountryName] ),
        VALUES ( DimSalesTerritory[SalesTerritoryCountry] )
    )
)
```

DAX Measure

Here's another example of how a DAX query can be converted into a measure. In this example, we create a list of countries that are not present in the DimSalesTerritory table.

```
• List of Countries not Present in DimSalesTerritory Table =
VAR _Country =
    DISTINCT (
        EXCEPT (
            VALUES ( DimGeography[RegionCountryName] ),
            VALUES ( DimSalesTerritory[SalesTerritoryCountry] )
        )
    )
RETURN
    CONCATENATEX ( _Country, [RegionCountryName], ", " )
```

SQL to DAX Query to DAX Measure

SQL Query

19. **Group by a column and perform an aggregation**

```sql
SELECT
  p.BrandName,
  SUM(s.SalesAmount) AS
Total_Sales
FROM FactSales s
JOIN DimProduct p ON s.ProductKey
= p.ProductKey
GROUP BY p.BrandName;
```

DAX Query

19. **Group by a column and perform an aggregation**

```
EVALUATE
GROUPBY (
    FactSales,
    DimProduct[BrandName],
    "TotalSales", SUMX (
CURRENTGROUP (),
FactSales[SalesAmount] )
)
```

DAX Measure

- A measure in Power BI is a dynamic calculation whose result depends on the current filter context. The dimension values in a visual act as filters. Power BI breaks down the measure calculation based on the dimension values. For each unique value of the dimension (e.g., each brand), the measure is recalculated within that subset of data.

  In our case, we want to group the total sales by each `BrandName`.

  We can create the following simple measure and plot against the `BrandName`.

  ```
  Total Sales =
  SUM ( FactSales[SalesAmount] )
  ```

  In visual, Power BI filters the `FactSales` table to include only the rows where Brand matches the current row in the visual and `SUM(FactSales[SalesAmount])` calculation is performed on this filtered subset.