SQL → DAX

HOW DO I CONVERT A QUERY?

OVERVIEW

SQL and DAX both handle data aggregations, but they work differently.

- **SQL is query-based**, where each query retrieves and processes data from a database.
- **DAX is context-driven**, working within a Power BI model where filters and relationships affect calculations.

Understanding differences helps avoid mistakes that lead to incorrect report results.

BASIC TRANSFORMATIONS

Feature	SQL	DAX
Aggregation	**SUM**(column)	**SUM**(column)
Grouping	**GROUP BY**	**SUMMARIZE**
Conditional Aggregation	**SUM(CASE WHEN ...)**	**CALCULATE (SUM( ),** filter)
Date-Based Aggregation	Manual date filtering **(WHERE OrderDate >= 'YYYY-MM-DD')**	**DATESYTD( ), TOTALYTD( )**
Filtering Data	**WHERE** clause	**FILTER( )** inside CALCULATE( )

Basic Aggregation

Summing Sales

Example: Calculate total sales for each product category.

SQL Approach:

```sql
SELECT ProductCategory, SUM(SalesAmount) AS TotalSales
FROM Sales
GROUP BY ProductCategory;
```

DAX Equivalent:

```dax
SUMMARIZE(
    Sales,
    Sales[ProductCategory],
    "TotalSales", SUM(Sales[SalesAmount])
)
```

In SQL, we explicitly use **GROUP BY**, while in DAX, the grouping is handled dynamically inside reports or using **SUMMARIZE( )**.

Conditional Aggregation

Filtering High-Value Sales

Example: Sum only sales where SalesAmount > 1000.

SQL Approach:

```sql
SELECT
    ProductCategory,
    SUM(CASE WHEN SalesAmount > 1000 THEN SalesAmount ELSE 0 END) AS HighValueSales
FROM Sales
GROUP BY ProductCategory;
```

DAX Equivalent:

```dax
HighValueSales =
CALCULATE(
    SUM(Sales[SalesAmount]),
    Sales[SalesAmount] > 1000
)
```

DAX uses **CALCULATE()** to apply filters dynamically, whereas SQL relies on **CASE WHEN**.

Date-Based Aggregation

Year-to-Date Sales

Example: Calculate total and Year-to-Date (YTD) sales.

SQL Approach:

```sql
SELECT
    ProductCategory,
    SUM(SalesAmount) AS TotalSales,
    SUM(CASE WHEN OrderDate >= '2023-01-01' THEN SalesAmount ELSE 0 END) AS SalesYTD
FROM Sales
GROUP BY ProductCategory;
```

DAX Equivalent:

```dax
SalesYTD =
CALCULATE(
    SUM(Sales[SalesAmount]),
    DATESYTD(Sales[OrderDate])
)
```

DAX provides built-in time intelligence functions (**DATESYTD()**), making date-based calculations more dynamic compared to hardcoded SQL conditions.

KEY TAKEAWAYS

- SQL and DAX use different methods for aggregations, but concepts like SUM, GROUP BY, and CASE WHEN have direct DAX equivalents.

- SUMMARIZE acts as GROUP BY, while CALCULATE allows filtering similar to CASE WHEN.

- DAX offers built-in time intelligence, unlike SQL, where date filters are manually defined.