// 1. Total Sales

Total Sales = SUM(orders[Sales])

// 2. Total Profit

Total Profit = SUM(orders[Profit])

// 3. Total Quantity Sold

Total Quantity = SUM(orders[Quantity])

// 4. Average Discount

Average Discount = AVERAGE(orders[Discount])

// 5. Total Discount

Total Discount = SUM(orders[Discount])

// 6. Count of Products

Product Count = COUNTROWS(orders)

// 7. Distinct Product Count

Distinct Product Count = DISTINCTCOUNT(orders[Product ID])

// 8. Profit per Product

**Profit per Product = DIVIDE(SUM(orders[Profit]), COUNTROWS(orders))**

This counts the **total number of rows** in the Orders table. Each row typically represents a unique **order** or **transaction** (not necessarily a unique product). So, if you have multiple orders for the same product, this count could include several rows for that same product.

So, the formula DIVIDE(SUM(Orders[Profit]), COUNTROWS(Orders)) calculates the **average profit per row**, which could be interpreted as "profit per transaction" or "profit per order," **not necessarily profit per product**, especially if a product appears in multiple rows (orders).

If you want to calculate **profit per unique product**, where each product is counted only once, you should use DISTINCTCOUNT to count unique products instead of COUNTROWS. Here's how you can adjust the formula:

**Profit Per Unique Product = DIVIDE(SUM(Orders[Profit]), DISTINCTCOUNT(Orders[Product ID]))**

// 9. Total Sales by Category

Sales by Category = SUMX(orders, orders[Sales])

SUMX(

<table>,

<expression>

)

SUMX is useful when the expression requires a calculation or involves **multiple columns**, for example:

SUMX is a **row-by-row** aggregation function in DAX that performs calculations over a table (or an expression that returns a table) and sums the results of an expression evaluated for each row in that table.

**Explanation**:

* This multiplies Sales[Quantity] by Sales[Price] for each row in the Sales table.
* Then, SUMX sums up the result of those row-by-row calculations.

In this case, SUMX is necessary because you are calculating the revenue for each row (Quantity × Price) before summing.

Total Revenue = SUMX(Orders, Orders[Quantity] \* Orders[Profit])

Total Sales After Discount = SUMX(orders, orders[Sales] \* (1 - orders[Discount]))

**Syntax of Calculate:**

CALCULATE(<expression>, <filter1>, <filter2>, ...)

**Key Points:**

* **Modifies the context**: CALCULATE changes the evaluation context by applying the specified filters, either overriding existing filters or adding new ones.
* **Performs aggregations**: Commonly used with functions like SUM, AVERAGE, COUNT, etc., to perform calculations under specific conditions.

// 10. Sales in Specific Region (East)

Sales in East Region = CALCULATE(SUM(orders[Sales]), orders[Region] = "East")

// 11. Sales Greater Than 1000

Sales > 1000 = CALCULATE(SUM(orders[Sales]), orders[Sales] > 1000)

// 12. Discounted Sales

Discounted Sales = CALCULATE(SUM(orders[Sales]), orders[Discount] > 0)

// 13. Profit by Category Filtered by Region (East, Furniture)

Profit in East by Category = CALCULATE(SUM(orders[Profit]), orders[Region] = "East", orders[Category] = "Furniture")

// 14. Sales of Specific Product (Product A)

Sales of Product A = CALCULATE(SUM(orders[Sales]), orders[Product Name] = "Xerox 211")

// 15. Sales for Previous Year

Sales for 2016 =

CALCULATE(

SUM(orders[Sales]),

YEAR('Orders'[Order Date]) = 2016

)

// 16. Total Sales by Region

Sales by Region = CALCULATE(SUM(orders[Sales]), ALLEXCEPT(orders, orders[Region]))

**CALCULATE:**

The CALCULATE function **modifies the context** in which a calculation (such as a sum or average) is evaluated. It applies **filters** to the calculation, changing how the data is aggregated or summarized.

The ALLEXCEPT function removes all filters **except** for the ones on specific columns that you specify. It is used to clear filters on the entire table, except for the columns explicitly stated.

// 17. Total Profit for High Sales Products

Profit High Sales = CALCULATE(SUM(orders[Profit]), orders[Sales] > 1000)

// 18. Total Sales with Discounts

Sales with Discounts = CALCULATE(SUM(orders[Sales]), orders[Discount] > 0)

// 19. Profit Margin

Profit Margin = DIVIDE(SUM(orders[Profit]), SUM(orders[Sales]))

//20. Count of States with Sales > 1000

States with Sales > 1000 = COUNTROWS(FILTER(orders, orders[Sales] > 1000))

Same DAX without filter:

States1 with Sales > 1000 =

CALCULATE(

COUNTROWS(orders),

orders[Sales] > 1000

)

**// 21. Rank Cities by Sales**

City Rank by Sales =

RANKX(

ALL(orders[City]), // This removes any filters on the City column

CALCULATE(SUM(orders[Sales])), // This sums up the sales for each city

, // Optional: You can specify an order argument here

DESC // Rank in descending order

)

The third parameter (which is left blank: ,) is **optional**. This parameter allows you to specify a specific value to be ranked against the other values in the column. In your case, it's not used because you are ranking all cities independently, not a specific value.

City1 Rank by Sales =

RANKX(

ALL(orders[City]), // This removes any filters on the City column

CALCULATE(SUM(orders[Sales])), // This sums up the sales for each city

CALCULATE(SUM(orders[Sales]), orders[City] = "New York City"), // This ranks New York's sales

DESC // Rank in descending order

)

States with Sales > 1000 =

COUNTROWS(

FILTER(

SUMMARIZE(orders, orders[State], "Total Sales", SUM(orders[Sales])),

[Total Sales] > 1000

)

)

// 22. Average Profit per Sale

Syntax:

AVERAGEX(<table>, <expression>)

Average Profit per Sale = AVERAGEX(orders, orders[Profit])

Or

Average Profit per Sale = AVERAGE(orders[Profit])

**Which to Choose?**

* **Use AVERAGE** when you are just averaging a column of values, like in this case with orders[Profit]. It’s simpler and more efficient than AVERAGEX for direct column averages.
* **Use AVERAGEX** when you need to iterate over a table and calculate more complex expressions for each row (like dividing, multiplying, or calculating conditions for each row).

// 23. Average Sales per Product

Avg Sales per Product = AVERAGEX(VALUES(orders[Product Name]), SUM(orders[Sales]))

// 24. Sales to Quantity Ratio

Sales to Quantity Ratio = DIVIDE(SUM(orders[Sales]), SUM(orders[Quantity]))

// 25. Average Sales by Region

Average Sales by Region = AVERAGEX(SUMMARIZE(orders, orders[Region], "Region Sales", SUM(orders[Sales])), [Region Sales])

**Overall Logic**

* The **SUMMARIZE** function creates a temporary table that summarizes total sales per region.
* The **AVERAGEX** function then takes this temporary table and calculates the average of the total sales for all regions.

SUMMARIZE(

<table>,

<groupBy\_column1>,

<groupBy\_column2>, ...,

[<name1>, <expression1>], [<name2>, <expression2>], ...

)

SUMMARIZE is used to create a temporary table that groups the data by a specified column (in this case, Region) and adds a calculated column (in this case, Region Sales).

orders[Region]: This is the column by which the table is grouped. The result will show one row per region.

"Region Sales": This is the name of the new calculated column.

SUM(orders[Sales]): For each region, this part calculates the **total sales**.

**Explanation**

1. **SUMMARIZE(orders, orders[State], "Total Sales", SUM(orders[Sales]))**:
   * This part creates a temporary table that summarizes total sales for each unique state.
   * It groups by orders[State] and calculates the total sales for each state.
2. **FILTER(..., [Total Sales] > 1000)**:
   * This filters the summarized table to include only states where total sales exceed 1000.
3. **COUNTROWS(...)**:
   * Finally, it counts the number of rows in the filtered table, which represents the unique states with sales greater than 1000.