**The Searched CASE Statement**

The Searched **CASE Statement** is SQL's way of doing **"if this, then that, otherwise this"** logic across your data.1

It checks a series of true/false conditions one after the other.2 As soon as it finds the first condition that is true, it stops and returns the value you specified.3 It's great for creating new columns that categorize or label your existing data based on complex rules.4

**Syntax**

SELECT column1,

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

ELSE default\_result

END AS new\_column\_name

FROM table\_name;

**Basic Example Query**

Let's categorize the 'Price' of products in a 'Products' table:

SELECT Name, Price,

CASE

WHEN Price > 100 THEN 'Expensive'

WHEN Price > 50 THEN 'Moderate'

ELSE 'Cheap'

END AS PriceCategory

FROM Products;

**The Simple CASE Statement**

The **Simple CASE Statement** is a cleaner, shorter version of the CASE logic that works only for **direct, exact comparisons** of a single value.

Instead of writing a full condition (WHEN [column] = 'A' THEN...), you list the column name right after CASE, and then only need to list the value you are checking for in the WHEN clauses. It's best used when you are just swapping out one specific value for another (like converting a status code to a full word).

**Syntax**

SELECT column1,

CASE expression\_to\_check

WHEN value1 THEN result1

WHEN value2 THEN result2

ELSE default\_result

END AS new\_column\_name

FROM table\_name;

**Basic Example Query**

Let's convert a 'StatusCode' (1, 2, or 3) into a word in an 'Orders' table:

SELECT OrderID,

CASE Status

WHEN 1 THEN 'Pending'

WHEN 2 THEN 'Shipped'

WHEN 3 THEN 'Delivered'

ELSE 'Unknown'

END AS OrderStatus

FROM Orders;

**Using CASE with Aggregate Functions**

Using **CASE with Aggregate Functions** is a clever trick to perform **multiple, specific counts or sums** in a single query, often showing the results side-by-side (like a pivot table).

For example, instead of running three separate queries to count 'High', 'Medium', and 'Low' values, you put a CASE statement *inside* a COUNT() function.5 The CASE assigns a value (like the number 1) only when its condition is met, and a NULL otherwise.6 Since COUNT() ignores NULLs, it only counts the rows that meet your specific condition.

**Key Logic**

To count a category: COUNT(CASE WHEN condition THEN 1 END)

To sum a value conditionally: SUM(CASE WHEN condition THEN column\_to\_sum ELSE 0 END)

**Basic Example Query**

To count the number of 'Shipped' orders and the number of 'Delivered' orders in a single row from an 'Orders' table:

SELECT

SUM(CASE WHEN Status = 'Shipped' THEN 1 ELSE 0 END) AS TotalShipped,

SUM(CASE WHEN Status = 'Delivered' THEN 1 ELSE 0 END) AS TotalDelivered

FROM Orders;