**Day 5-6: Conditional Logic & Complex Problems**

**Subtopic 1: The Searched CASE Statement**

* **Definition:** The CASE statement is SQL's way of handling if/then/else logic. The searched CASE statement evaluates a list of conditions and returns a value when the first condition is met.1 If no conditions are met, it returns the value in the ELSE clause, or NULL if there is no ELSE.
* **Use/Importance:** This is a data analyst's superpower for creating custom columns and segmentations on the fly. It allows you to transform raw data into meaningful categories (e.g., categorizing sales transactions as 'High', 'Medium', or 'Low') without changing the underlying data.
* **Syntax/Structure:**

CASE

WHEN condition1 THEN result1

WHEN condition2 THEN result2

...

ELSE else\_result

END AS new\_column\_name

* **Example:** Categorize products based on their price from a products table.

SELECT

product\_name,

price,

CASE

WHEN price > 1000 THEN 'Premium'

WHEN price BETWEEN 500 AND 1000 THEN 'Standard'

ELSE 'Economy'

END AS price\_category

FROM

products;

**Subtopic 2: The Simple CASE Statement**

* **Definition:** A simpler form of CASE that compares a single column or expression against a list of specific, discrete values. It's less flexible than the searched CASE but more concise for direct mappings.
* **Use/Importance:** It is ideal for replacing codes or abbreviations with more readable text.2 For example, converting a status\_code like '1' to 'Active', '2' to 'Inactive', and '3' to 'Pending'.
* **Syntax/Structure:**

SQL

CASE column\_name

WHEN value1 THEN result1

WHEN value2 THEN result2

...

ELSE else\_result

END AS new\_column\_name

* **Example:** Translate shipping method codes into full text descriptions.

SQL

SELECT

order\_id,

CASE shipping\_code

WHEN 'STD' THEN 'Standard Shipping'

WHEN 'EXP' THEN 'Express Shipping'

WHEN 'NXT' THEN 'Next-Day Air'

ELSE 'Unknown'

END AS shipping\_method

FROM

orders;

**Subtopic 3: Using CASE with Aggregate Functions**

* **Definition:** This advanced technique involves placing a CASE statement inside an aggregate function like SUM() or COUNT() to perform conditional aggregation. This allows you to count or sum different subsets of data as separate columns in a single query.
* **Use/Importance:** It is extremely useful for creating summary reports or pivot tables directly in SQL. You can count different types of items in one pass over the data, which is far more efficient than running multiple queries.
* **Syntax/Structure:**

SQL

AGGREGATE\_FUNCTION(

CASE

WHEN condition THEN column\_name\_or\_value

ELSE NULL

END

) AS conditional\_aggregate\_column

* **Example:** From a single employees table, count the number of male and female employees in one query.

SQL

SELECT

COUNT(CASE WHEN gender = 'Male' THEN employee\_id END) AS male\_employees,

COUNT(CASE WHEN gender = 'Female' THEN employee\_id END) AS female\_employees

FROM

employees;

**Quick Recap:**

* The **CASE** statement adds if/then/else logic to your queries, allowing you to create new, derived columns.
* The **Searched CASE** (WHEN condition THEN ...) is best for complex conditions involving ranges, comparisons, or different columns.
* The **Simple CASE** (CASE column WHEN value THEN ...) is a concise way to map specific values to new results.
* Placing CASE inside **aggregate functions** like COUNT() or SUM() enables you to create powerful summary reports by conditionally aggregating data into separate columns.

**Practice Tasks:**

1. **Task 1:** From an employees table with a hire\_date column, write a query that adds a new column named tenure\_category. It should categorize employees as 'New Hire' if hired after Jan 1, 2024, 'Veteran' if hired before Jan 1, 2020, and 'Established' for everyone in between.
2. **Task 2:** You have a support\_tickets table with a status column containing values like 'Open', 'Closed', 'In Progress'. Write a query to count the total number of tickets in each status, presenting each count in a separate column (open\_tickets, closed\_tickets, in\_progress\_tickets).
3. **Task 3:** From a sales table, create a commission\_rate column. If the sale\_amount is over $5,000, the rate is 10% (0.10). If it's between $1,000 and $5,000, the rate is 7% (0.07). Otherwise, the rate is 5% (0.05).
4. **Task 4 (Challenge):** Using a customers table with a country column, write a query that counts the total number of customers from 'India', the total from 'USA', and the total from all 'Other' countries in a single result with three columns.