## CASE STATEMENT IN SQL

The `CASE` statement in SQL is a powerful tool used for conditional logic within queries. It allows you to perform conditional logic and return different values based on different conditions. This is particularly useful for transforming data, creating derived columns, or categorizing results.

**There are two types of `CASE` statements:**

**1. Simple CASE Statement:**

- Compares an expression to a set of simple expressions and returns the result of the first matching expression.

```
CASE expression
    WHEN value1 THEN result1
    WHEN value2 THEN result2
    ...
    ELSE default_result
END
```

**2. Searched CASE Statement:**

- Evaluates a set of Boolean expressions and returns the result of the first expression that evaluates to true.

```
CASE
    WHEN condition1 THEN result1
    WHEN condition2 THEN result2
    ...
    ELSE default_result
END
```

**Industry-Related Examples**

**1. Customer Loyalty Tiers**

**Scenario:** In a retail business, you want to categorize customers based on their total purchase amount into different loyalty tiers.

```
SELECT CustomerID, TotalPurchaseAmount,
    CASE
        WHEN TotalPurchaseAmount >= 1000 THEN 'Gold'
        WHEN TotalPurchaseAmount >= 500 THEN 'Silver'
        ELSE 'Bronze'
    END AS LoyaltyTier
FROM Customers;
```

- **Explanation**: This query categorizes customers into 'Gold', 'Silver', or 'Bronze' loyalty tiers based on their total purchase amount.

**2. Employee Salary Bands**

**Scenario**: In a company, you want to classify employees into different salary bands for reporting purposes.

```
SELECT EmployeeID, Salary,
    CASE
        WHEN Salary > 100000 THEN 'High'
        WHEN Salary BETWEEN 50000 AND 100000 THEN 'Medium'
        ELSE 'Low'
    END AS SalaryBand
FROM Employees;
```

- **Explanation**: This query assigns employees to 'High', 'Medium', or 'Low' salary bands based on their salary.

**3. Product Performance**

**Scenario**: In an e-commerce system, you want to classify products based on their sales performance as 'Top Seller', 'Average Seller', or 'Low Seller'.

```
SELECT ProductID, ProductName, SalesAmount,
    CASE
        WHEN SalesAmount > 50000 THEN 'Top Seller'
        WHEN SalesAmount BETWEEN 20000 AND 50000 THEN 'Average Seller'
        ELSE 'Low Seller'
    END AS PerformanceCategory
FROM Products;
```

- **Explanation**: This query classifies products into different performance categories based on their sales amount.

**4. Order Status**

**Scenario**: In an order management system, you want to provide a more descriptive status for each order based on its current status code.

```
SELECT OrderID, StatusCode,
    CASE StatusCode
        WHEN 'P' THEN 'Pending'
        WHEN 'S' THEN 'Shipped'
        WHEN 'D' THEN 'Delivered'
        WHEN 'C' THEN 'Cancelled'
```

```
        ELSE 'Unknown Status'
    END AS StatusDescription
FROM Orders;
```

- **Explanation**: This query translates status codes into descriptive status messages for orders.

**5. Student Grade Classification**

**Scenario**: In an educational institution, you want to classify student grades into letter grades (A, B, C, etc.) based on their numeric scores.

```
SELECT StudentID, Score,
    CASE
        WHEN Score >= 90 THEN 'A'
        WHEN Score >= 80 THEN 'B'
        WHEN Score >= 70 THEN 'C'
        WHEN Score >= 60 THEN 'D'
        ELSE 'F'
    END AS LetterGrade
FROM Students;
```

- **Explanation**: This query assigns letter grades to students based on their numeric scores.

**Summary**

1. Simple CASE Statement: Compares an expression to a set of values and returns the result of the first match.
2. Searched CASE Statement: Evaluates Boolean conditions and returns the result of the first true condition.

The `CASE` statement is versatile and can be used to handle complex data transformations, create new calculated columns, and enhance data readability in queries. It's especially useful for data analysts when creating reports or preparing data for further analysis.