The CASE statement in SQL is a powerful tool used to implement conditional logic within SQL queries. It allows you to perform if-then-else logic in a query, making it possible to return different values based on certain conditions. This guide will cover the basics of the CASE statement, its syntax, and provide examples to illustrate its use in various scenarios.

Basic Syntax

The CASE statement comes in two forms: the simple CASE and the searched CASE.

1. Simple CASE Syntax

The simple CASE statement compares an expression to a set of simple expressions to determine the result.

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

2. Searched CASE Syntax

The searched CASE statement evaluates a set of Boolean expressions to determine the result.

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

Examples of Using CASE Statements

Example 1: Basic Usage

We want to classify employees based on their department. Specifically, we want to label employees in the "**IT**" department as "**IT Team**" and all others as "Other".

employee_id	first_name	Age	department
1	John Doe	28	HR
2	Jane Smith	34	Finance
3	Jim Brown	25	IT
4	Jake White	30	Marketing
5	Jill Black	29	IT

```
SELECT
  name,
  CASE department
    WHEN 'IT' THEN 'IT Team'
    ELSE 'Other'
  END AS department_group
FROM employees;
```

This query will return the names of the employees along with a classification of their department.

Explanation

- SELECT name: Retrieves the name column from the employees table.
- CASE department: Starts a CASE statement that evaluates the department column.
- WHEN 'IT' THEN 'IT Team': If the department is 'IT', the result is 'IT Team'.
- ELSE 'Other': For all other departments, the result is 'Other'.
- END AS department_group: Ends the CASE statement and assigns an alias department_group to the result.

name	department_group
John Doe	Other
Jane Smith	Other
Jim Brown	IT Team
Jake White	Other
Jill Black	IT Team

Example 2: Using Multiple Conditions

Suppose we want to classify employees into more specific age groups: 'Junior' for those under 25, 'Young' for those between 25 and 30, and 'Senior' for those over 30.

```
SELECT
  employee_id,
  name,
  age,
  department,
  CASE
    WHEN age < 25 THEN 'Junior'
    WHEN age BETWEEN 25 AND 30 THEN 'Young'
    ELSE 'Senior'
  END AS age_group
FROM employees;
```

employee_id	name	age	department	age_group
1	John Doe	28	HR	Young
2	Jane Smith	34	Finance	Senior
3	Jim Brown	25	IT	Young
4	Jake White	30	Marketing	Young
5	Jill Black	29	IT	Young

Example 3: Nested CASE Statements

The goal here is to label employees under 30 as 'Jr Sales' if they are in the Sales department, otherwise as 'Junior'.

```
SELECT
  name,
  CASE
    WHEN age < 30 THEN
      CASE
        WHEN department = 'Sales' THEN 'Jr Sales'
        ELSE 'Junior'
      END
    ELSE 'Senior'
  END AS employee_name
FROM employees;
```

name	employee_name
John Doe	Junior
Jane Smith	Senior
Jim Brown	Junior
Jake White	Senior
Jill Black	Junior

Explanation

- SELECT name: Retrieves the name column from the employees table.
- CASE WHEN age < 30 THEN ... END: Evaluates if the employee's age is less than 30.
- Nested CASE: Inside the first CASE:
- WHEN department = 'Sales' THEN 'Jr Sales': If the department is 'Sales', then 'Jr Sales'.
- ELSE 'Junior': If the department is not 'Sales', then 'Junior'.
- ELSE 'Senior': If the employee is not under 30, label as 'Senior'.

- END AS employee_name: Ends the CASE statement and assigns the result the alias employee_name.

Example 4: Nested CASE with Additional Conditions

Here, we introduce more conditions to label employees differently based on age and other criteria.

```
SELECT
  name,
  CASE
    WHEN age < 30 THEN
      CASE
        WHEN department = 'Sales' THEN 'Jr Sales'
        ELSE 'Junior'
      END
    WHEN age >= 30 AND age <= 38 THEN
      CASE
        WHEN department = 'Sales' THEN 'Mid Sales'
        ELSE 'Middle'
      END
    ELSE 'Senior'
  END AS employee_name
FROM employees;
```

name	employee_name
John Doe	Junior
Jane Smith	Middle
Jim Brown	Junior
Jake White	Middle
Jill Black	Junior

Explanation

- SELECT name: Retrieves the name column from the employees table.
- CASE: Starts the outer CASE statement.
- WHEN age < 30 THEN ... END: For employees under 30:
- Nested CASE: Labels as 'Jr Sales' if in 'Sales', otherwise 'Junior'.
- WHEN age >= 30 AND age <= 38 THEN ... END: For employees between 30 and 38:
- Nested CASE: Labels as 'Mid Sales' if in 'Sales', otherwise 'Middle'.
- ELSE 'Senior': For all other employees, label as 'Senior'.
- END AS employee_name: Ends the CASE statement and assigns the result the alias employee_name.

Example 5: Handling NULL Values in the Department Column

If the department is NULL, we can handle it by assigning a default value.

```
SELECT
  name,
  CASE
    WHEN department IS NULL THEN 'No Department Assigned'
    ELSE department
  END AS department_status
FROM employees;
```

first_name	last_name	department	department_status
John	Doe	HR	HR
Jane	Smith	Finance	Finance
Jim	Brown	IT	IT
Jake	White	Marketing	Marketing
Jill	Black	NULL	No Department Assigned