Lecture Notes – Case_When-Where

Section 1: Lecture Summary

This lecture covers the use of **CASE WHEN** expressions within the WHERE clause for conditional row filtering. Rather than writing simple WHERE conditions directly, CASE WHEN allows you to construct more complex conditional logic that evaluates to a result value. The WHERE clause then filters rows based on whether this result equals a specified value (typically 1 for TRUE). This approach becomes particularly valuable when multiple conditions need to be evaluated together.

Section 2: Key Concepts and Explanations

CASE WHEN in WHERE Clause

The WHERE clause is designed for conditional filtering of rows. By introducing CASE WHEN, you extend its capability to handle complex conditional logic. Instead of writing a simple condition like "WHERE column > value," you can write a CASE WHEN block that evaluates multiple conditions and returns a numeric result, which the WHERE clause then checks.

Basic Structure

The syntax follows this pattern:

```
SELECT column1, column2
FROM table
WHERE CASE
  WHEN condition1 THEN result_value
  WHEN condition2 THEN result_value
  ELSE default_value
END = target_value
```

The CASE WHEN block evaluates conditions sequentially. When a condition is TRUE, it returns the specified result value. If no conditions are met, the ELSE clause provides a default value. The entire CASE expression is then compared to a target value in the WHERE clause (commonly 1 for TRUE conditions).

Why Use CASE WHEN in WHERE

While simple conditions can often be written directly in WHERE, CASE WHEN becomes advantageous when:

- Multiple related conditions must be evaluated together

- Complex logic needs to be simplified and organized

- You want to make conditional filtering more readable and maintainable

Section 3: Example Code and Use Cases

Example 1: Filter Employees Hired in 2020 or Later

```
SELECT EmpID, FirstName, LastName, HireDate
FROM Employees
WHERE CASE
  WHEN HireDate >= '2020-01-01' THEN 1
  ELSE 0
END = 1;
```

This query retrieves employees hired on or after January 1, 2020. The CASE expression returns 1 when the hire date condition is met, and 0 otherwise. The WHERE clause then filters to show only records where the CASE result equals 1.

Example 2: Filter Premium Products with Price Above Threshold

```
SELECT ProductID, ProductName, Price
FROM Products
WHERE CASE
  WHEN Price >= 50000 THEN 1
  ELSE 0
END = 1;
```

This query returns products with a price of 50,000 or greater. The CASE WHEN structure allows for straightforward evaluation, though in this case a simple WHERE Price >= 50000 would achieve the same result.

Example 3: Multiple Conditions - High-Value Orders from Specific Regions

```
SELECT OrderID, CustomerID, TotalAmount
FROM Orders o
JOIN Customers c ON o.CustomerID = c.CustomerID
WHERE CASE
  WHEN TotalAmount > 5000 AND c.City IN ('New York', 'Los Angeles') THEN 1
  WHEN TotalAmount > 3000 AND c.City = 'Chicago' THEN 1
  ELSE 0
END = 1;
```

This example demonstrates the power of CASE WHEN with multiple distinct conditions. Different cities have different spending thresholds, and the CASE expression handles both conditions elegantly.

Example 4: Categorizing Employees by Salary Range in WHERE Clause

```
SELECT EmpID, FirstName, LastName, Salary
FROM Employees
WHERE CASE
  WHEN Salary >= 80000 THEN 1
  WHEN Salary >= 50000 AND Salary < 80000 THEN 1
  ELSE 0
END = 1;
```

This retrieves employees earning 50,000 or more by using nested conditions within the CASE WHEN block.

Section 4: Key Takeaways

CASE WHEN extends WHERE clause functionality by allowing conditional expressions instead of simple comparisons. You construct a CASE block that evaluates one or more conditions and returns a numeric value, which WHERE then uses for filtering.

The result value (typically 1) acts as a TRUE indicator. You can use any numeric value, but 1 is the standard convention for TRUE conditions. The WHERE clause compares the entire CASE expression to this target value.

CASE WHEN is most valuable with multiple conditions. While simple single conditions can be written directly in WHERE, CASE WHEN becomes advantageous when you have complex logic involving several related conditions that you want to organize clearly.

Readability improves with CASE WHEN. Complex filtering logic becomes more organized and maintainable when structured as a CASE WHEN block rather than multiple AND/OR operators in a single WHERE condition.