CASE STATEMENT

In SQL wird die `CASE` -Anweisung verwendet, um bedingte Logik direkt in eine SQL-Abfrage einzubauen. Sie funktioniert ähnlich wie `if-else` -Anweisungen in Programmiersprachen. Das `CASE` -Statement überprüft eine Bedingung und gibt basierend auf dieser Bedingung unterschiedliche Werte zurück. Dies ist besonders nützlich, um Daten dynamisch zu kategorisieren oder unterschiedliche Berechnungen abhängig von bestimmten Werten vorzunehmen.

Hier das von Masterschool verlinkte Youtube Video
Intermediate SQL Tutorial | Case Statement | Use C...

Hier die allgemeine Syntax und ein paar Beispiele:

Syntax von CASE

```
SELECT
  CASE
    WHEN Bedingung1 THEN Ergebnis1
    WHEN Bedingung2 THEN Ergebnis2
    ...
    ELSE ErgebnisN
  END AS Alias_Name
FROM Tabelle;
```

- **`WHEN`** : Jede Bedingung, die überprüft wird.
- **`THEN`** : Der Wert oder das Ergebnis, das zurückgegeben wird, wenn die Bedingung wahr ist.
- **`ELSE`** (optional): Der Wert, der zurückgegeben wird, wenn keine der `WHEN` -Bedingungen erfüllt ist.
- **`END`** : Abschluss des `CASE` -Blocks, oft mit einem Alias (z. B. `AS Alias_Name`) benannt.

Beispiel 1: Klassifizierung basierend auf einem numerischen Wert

Angenommen, wir haben eine Tabelle `students` mit den Spalten `name` und `score` . Wir möchten die Studenten basierend auf ihrem Score als "Bestanden" oder "Nicht

bestanden" klassifizieren.

```
SELECT
    name,
    score,
    CASE
        WHEN score >= 50 THEN 'Bestanden'
        ELSE 'Nicht bestanden'
    END AS Ergebnis
FROM students;
```

In diesem Beispiel:

- Studenten mit einer `score` von 50 oder höher werden als "Bestanden" markiert.
- Alle anderen Studenten werden als "Nicht bestanden" markiert.

Beispiel 2: Verschiedene Kategorien erstellen

Nehmen wir an, wir haben eine Tabelle `products` mit den Spalten `product_name` und `price`. Wir möchten die Produkte in Preiskategorien einteilen.

```
SELECT
    product_name,
    price,
    CASE
        WHEN price < 20 THEN 'Günstig'
        WHEN price BETWEEN 20 AND 100 THEN 'Mittel'
        WHEN price > 100 THEN 'Teuer'
        ELSE 'Keine Angabe'
    END AS Preis_Kategorie
FROM products;
```

Hier wird:

- Der Preis unter 20 als "Günstig" klassifiziert.
- Ein Preis zwischen 20 und 100 als "Mittel".
- Ein Preis über 100 als "Teuer".
- Die `ELSE`-Anweisung fängt alle anderen Fälle ab, z. B. wenn der Preis NULL ist.

Beispiel 3: CASE in Berechnungen verwenden

In manchen Fällen möchten wir Berechnungen abhängig von einer Bedingung durchführen. Angenommen, wir haben eine Tabelle `sales` mit den Spalten `sale_amount` und `region`. In bestimmten Regionen soll ein Bonus berechnet werden.

```
SELECT
    sale_amount,
    region,
    CASE
        WHEN region = 'Nord' THEN sale_amount * 0.1
        WHEN region = 'Süd' THEN sale_amount * 0.15
        ELSE sale_amount * 0.05
    END AS Bonus
FROM sales;
```

In diesem Beispiel wird:

- Ein Bonus von 10 % für Verkäufe in der Region "Nord" angewandt.
- Ein Bonus von 15 % für die Region "Süd".
- Ein Bonus von 5 % für alle anderen Regionen.

Hinweise und Best Practices

1. **Datentyp-Konsistenz**: Alle Rückgabewerte in einem `CASE`-Ausdruck sollten denselben Datentyp haben.
2. **NULL-Werte**: Wenn NULL-Werte berücksichtigt werden müssen, können spezifische `WHEN`-Bedingungen dafür definiert oder `COALESCE` genutzt werden.
3. **Lesbarkeit**: Bei komplexen Bedingungen ist es oft hilfreich, `CASE`-Anweisungen in kleinere `SELECT`-Anweisungen oder temporäre Tabellen aufzuteilen, um die Abfrage besser lesbar zu machen.

Die `CASE`-Anweisung ist flexibel und mächtig, besonders wenn komplexe Bedingungslogik direkt in SQL benötigt wird.