Introduction to Conditional Expressions week 13

Power Learn Project

Database Design

You are live with Gerald Macherechedze

Learning Objectives

Learn the concept of conditional expressions in a database.

- Understand the basic syntax of the CASE statement.
- Explore the use of CASE for conditional logic in SQL queries.
- Apply simple and searched CASE expressions.
- Comprehend the role of conditional expressions in data transformation.
- Use conditional expressions for creating calculated columns.
- Understand the impact of conditional expressions on result sets.
- Apply conditional expressions in filtering and sorting data.

Explore scenarios where conditional expressions enhance query flexibility.

I. Introduction to Conditional Expressions

Power Learn Project

Conditional expressions in SQL allow you to perform actions or make decisions based on specified conditions. These expressions are valuable in various scenarios, such as filtering data, controlling flow, and manipulating result sets.

Predefined database :

Step 1: CREATE DATABASE homecafe;

https://drive.google.com/drive/folders/12z6P3DMr38DZxG3hHOkfXReEceqHvnWc?usp=sharing

II. The CASE Statement

1. Basic Syntax: CASE WHEN condition1 THEN result1 WHEN condition2 THEN result2 ... ELSE default_result END;

2. Simple Example:

```
SELECT order_id,
CASE WHEN order_amount  > 50000 THEN 'High order amount'
WHEN order_amount > 30000 THEN 'Moderate order amount'
ELSE 'Low order amount' END AS order_amount_category
FROM orders;
```

III. COALESCE Function

1. Purpose:

Helps to deal with missing values (NULL) in your data. It takes a list of arguments and returns the first non-null value it encounters

2. Syntax: COALESCE(expression1, expression2, ...);

3. Example: SELECT product_name,

COALESCE(discounted_price, regular_price)

AS final_price FROM products;

IV. NULLIF Function

1. Purpose:

Returns null if the two expressions are equal; otherwise, returns the first expression.

2. Syntax:

NULLIF(expression1, expression2);

3. Example:

SELECT order_id, product_name,

NULLIF(discount_rate, 0) AS non_zero_discount FROM Orders;

V. IFNULL / NVL Function

1. Purpose:

Returns the second expression if the first expression is null. Helps you deal with missing values (NULL) by providing a default value in their place.

2. Syntax: IFNULL(expression1, expression2); -- or NVL(expression1, expression2);

3. Example: SELECT product_name, IFNULL(discounted_price, regular_price) AS final_price FROM products;

IF Function

1. Purpose:

Returns one of two values depending on whether the specified condition is true or false.

2. Syntax: IIF(condition, true_value, false_value);

3. Example:

SELECT order_id, IF(order_status = 'Shipped', 'Completed', 'Pending') AS order_status_display FROM orders;

VII. Conclusion

Conditional expressions enhance the flexibility and power of SQL queries, allowing you to handle various scenarios and customize the presentation of data based on specific conditions. By mastering these conditional constructs, you can write more dynamic and expressive SQL queries.