📌 Dataset Overview

Table Name: products

Columns:

- product_name
- brand_name
- marked_price
- discounted_price
- rating
- rating_count
- brand_tag
- product_tag

Purpose:
Used to demonstrate SQL data retrieval and manipulation queries.

1️⃣ Displaying the Dataset

🔷 Show All Data

SELECT * FROM products;

- Displays all rows and all columns.
- * means everything.

2️⃣–3️⃣ Selecting Specific Columns

🔷 Select Specific Columns

SELECT product_name, brand_name FROM products;

- Displays only selected columns.

🔷 Column Order Matters

SELECT brand_name, product_name FROM products;

- Output column order changes based on SELECT order.

4️⃣–6️⃣ Creating New Columns (Using Calculations)

We can perform calculations inside SELECT using arithmetic operators.

🔷 Discounted Amount

SELECT marked_price - discounted_price AS discounted_amount

FROM products;

Formula:
Discount = Marked Price - Discounted Price

🔷 Rating Filter

SELECT rating * rating_count AS rating_filter

FROM products;

Used to calculate weighted rating score.

🔷 Discount Percentage

SELECT ((marked_price - discounted_price) / marked_price) * 100

AS discounted_percent

FROM products;

Formula:

\text{Discount %} = \frac{(Marked - Discounted)}{Marked} \times 100

👉 AS is used to give a new column name (alias).

7️⃣ Finding Unique Values

🔷 DISTINCT

SELECT DISTINCT(brand_name) AS unique_brands

FROM products;

- Removes duplicate values.
- Shows only unique brand names.

8️⃣ Adding WHERE Clause (Filtering Data)

SELECT *

FROM products

WHERE brand_tag = 'Adidas';

- Filters rows based on condition.
- Only Adidas products are shown.

9️⃣ DISTINCT with WHERE

SELECT DISTINCT(product_tag), brand_name

FROM products

WHERE brand_tag = 'Adidas';

- Shows unique product tags for Adidas.

🔟 Counting Distinct Values

SELECT COUNT(DISTINCT(product_tag))

FROM products

WHERE brand_tag = 'Adidas';

- Counts number of unique product tags.
- COUNT() is an aggregate function.

1️⃣1️⃣ Multiple Conditions (AND)

🔷 Using AND

WHERE brand_tag = 'Adidas'

AND discounted_price > 5000;

- Both conditions must be TRUE.

🔷 Using BETWEEN (Range Filter)

WHERE discounted_price BETWEEN 5000 AND 8000;

- Includes both 5000 and 8000.
- Equivalent to:

discounted_price >= 5000 AND discounted_price <= 8000

1️⃣2️⃣ Complex AND Conditions

WHERE (discounted_price BETWEEN 5000 AND 8000)

AND brand_tag = 'Adidas'

AND rating > 4

AND rating_count > 10;

- Combines multiple filters.
- Parentheses improve readability.
- All conditions must be true.

1️⃣3️⃣ Using OR Condition

WHERE (brand_tag = 'Adidas' OR brand_tag = 'Puma')

AND discounted_price BETWEEN 3000 AND 5000;

- OR → At least one condition must be TRUE.
- Combined with AND for multiple logic conditions.

1️⃣4️⃣ Using NOT Condition

WHERE NOT (brand_tag = 'Adidas' OR brand_tag = 'Puma')

AND discounted_price BETWEEN 3000 AND 5000;

- Excludes Adidas and Puma.
- Returns other brands within price range.

1️⃣5️⃣ Using IN Condition

WHERE brand_tag IN ('Adidas', 'Puma')

AND discounted_price BETWEEN 3000 AND 5000;

- Shorter version of multiple OR conditions.
- Cleaner and more readable.

Equivalent to:

brand_tag = 'Adidas' OR brand_tag = 'Puma'

1️⃣6️⃣ Using NOT IN Condition

WHERE brand_tag NOT IN ('Adidas', 'Puma')

AND discounted_price BETWEEN 3000 AND 5000;

- Excludes listed values.
- Alternative to multiple NOT conditions.