Practical No. 11

Aim: Implement set operations, case statement and view queries.

Theory:

You can perform various operations to retrieve, manipulate, and present data effectively by using, the Set operations allow you to combine and manipulate data from multiple tables or result sets. Common set operations include UNION, INTERSECT, and EXCEPT (or MINUS in some database systems). The CASE statement is a powerful conditional expression in SQL that allows you to perform conditional operations within a query. A view in a relational database is a virtual table that is generated by a query and represents the result set of that query.

Queries:

1)Implement “IF” Condition in Query

a) Put if condition on “price” attribute (IF Else)

```
mysql> SELECT PRICE,
    -> CASE
    -> WHEN PRICE > 1000 THEN 'HIGH'
    -> WHEN PRICE BETWEEN 500 AND 1000 THEN 'MEDIUM'
    -> ELSE 'LOW'
    -> END AS RATING
    -> FROM PRODUCT;
+----------+--------+
| PRICE    | RATING |
+----------+--------+
|  1200.00 | HIGH   |
|  1500.00 | HIGH   |
|   700.00 | MEDIUM |
|   900.00 | MEDIUM |
|  1000.00 | MEDIUM |
| 15000.00 | HIGH   |
+----------+--------+
6 rows in set (0.00 sec)

mysql> #202203103510097
```

b) Try nested IF on “price” attribute.

```
mysql> SELECT PRICE,
    -> CASE
    -> WHEN PRICE > 1000 THEN 'HIGH'
    -> ELSE
    -> CASE
    -> WHEN PRICE > 500 THEN 'MEDIUM'
    -> ELSE 'LOW'
    -> END
    -> END AS RATING
    -> FROM PRODUCT;
+----------+--------+
| PRICE    | RATING |
+----------+--------+
|  1200.00 | HIGH   |
|  1500.00 | HIGH   |
|   700.00 | MEDIUM |
|   900.00 | MEDIUM |
|  1000.00 | MEDIUM |
| 15000.00 | HIGH   |
+----------+--------+
6 rows in set (0.00 sec)

mysql> #202203103510097
```

c) Display Price and quantity and there rating with “High” , “Medium” and “Low”

```
mysql> SELECT PRICE, QUANTITY,
    -> CASE
    -> WHEN PRICE > 1000 THEN
    -> CASE
    -> WHEN QUANTITY > 5 THEN 'HIGH PRICE AND HIGH QUANTITY'
    -> ELSE 'HIGH PRICE AND LOW QUANTITY'
    -> END
    -> WHEN PRICE BETWEEN 500 AND 1000 THEN
    -> CASE
    -> WHEN QUANTITY > 5 THEN 'MEDIUM PRICE AND HIGH QUANTITY'
    -> ELSE 'MEDIUM PRICE AND LOW QUANTITY'
    -> END
    -> ELSE
    -> CASE
    -> WHEN QUANTITY > 5 THEN 'LOW PRICE AND HIGH QUANTITY'
    -> ELSE 'LOW PRICE AND LOW QUANTITY'
    -> END
    -> END AS RATING
    -> FROM PRODUCT NATURAL JOIN ONLINE;
+---------+----------+-------------------------------+
| PRICE   | QUANTITY | RATING                        |
+---------+----------+-------------------------------+
| 1200.00 |        5 | HIGH PRICE AND LOW QUANTITY   |
| 1500.00 |       10 | HIGH PRICE AND HIGH QUANTITY  |
| 1200.00 |        5 | HIGH PRICE AND LOW QUANTITY   |
|  700.00 |       25 | MEDIUM PRICE AND HIGH QUANTITY |
| 1000.00 |       30 | MEDIUM PRICE AND HIGH QUANTITY |
|  900.00 |       15 | MEDIUM PRICE AND HIGH QUANTITY |
| 1000.00 |       30 | MEDIUM PRICE AND HIGH QUANTITY |
+---------+----------+-------------------------------+
7 rows in set (0.06 sec)

mysql> #202203103510097
```

2) Create a view VProduct of product‟s id, description and price.

```
mysql> CREATE VIEW VPRODUCT AS
    -> SELECT PRODUCT_NO, DESCRIPTION, PRICE
    -> FROM PRODUCT;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM VPRODUCT;
+------------+----------------+----------+
| PRODUCT_NO | DESCRIPTION    | PRICE    |
+------------+----------------+----------+
|        120 | REDUCER        |  1200.00 |
|        121 | PLATE          |  1500.00 |
|        122 | HANDLE         |   700.00 |
|        124 | WIDGET REMOVER |   900.00 |
|        136 | SIZE WIDGET    |  1000.00 |
|        137 | SIZE WIDGET    | 15000.00 |
+------------+----------------+----------+
6 rows in set (0.00 sec)

mysql> #202203103510097
```

3) Create a view of Vorder to get orders (order_id, product_id, description, customer_name, quantity) placed by customer who belongs to “BRIXTON”.

```
mysql> CREATE VIEW VORDER AS
    -> SELECT CO.CORDER_NO, O.PRODUCT_NO, P.DESCRIPTION, C.NAME, O.QUANTITY
    -> FROM CORDER CO
    -> JOIN CUSTOMER C ON CO.CUSTOMER_NO = C.CUSTOMER_NO
    -> JOIN ONLINE O ON CO.CORDER_NO = O.CORDER_NO
    -> JOIN PRODUCT P ON O.PRODUCT_NO = P.PRODUCT_NO
    -> WHERE C.ADDRESS = 'BRIXTON';
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT * FROM VORDER;
+-----------+------------+-------------+-------------+----------+
| CORDER_NO | PRODUCT_NO | DESCRIPTION | NAME        | QUANTITY |
+-----------+------------+-------------+-------------+----------+
|       203 |        122 | HANDLE      | DRAKE       |       20 |
|       204 |        136 | SIZE WIDGET | GARRY SMITH |       30 |
+-----------+------------+-------------+-------------+----------+
2 rows in set (0.01 sec)

mysql> #202203103510097
```

4) Insert a new row in VProduct 135, „Sofa“ and 35000.

```
mysql> INSERT INTO VProduct VALUES(135, "SOFA", 35000);
Query OK, 1 row affected (0.07 sec)

mysql> SELECT * FROM VProduct;
+------------+----------------+-------+
| PRODUCT_NO | DESCRIPTION    | PRICE |
+------------+----------------+-------+
|        120 | REDUCER        |  1200 |
|        121 | PLATE          |  2000 |
|        122 | HANDLE         |   700 |
|        124 | WIDGET REMOVER |   900 |
|        136 | SIZE WIDGET    |  1000 |
|        137 | SIZE WIDGET    | 15000 |
|        135 | SOFA           | 35000 |
+------------+----------------+-------+
7 rows in set (0.00 sec)
```

5) Update product“s quantity to 25 which is brought by customer „DRAKE“ in Vorder.

```
mysql> UPDATE VORDER
    -> SET QUANTITY = 25
    -> WHERE NAME = 'DRAKE';
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM VORDER;
+-----------+------------+-------------+-------------+----------+
| CORDER_NO | PRODUCT_NO | DESCRIPTION | NAME        | QUANTITY |
+-----------+------------+-------------+-------------+----------+
|       203 |        122 | HANDLE      | DRAKE       |       25 |
|       204 |        136 | SIZE WIDGET | GARRY SMITH |       30 |
+-----------+------------+-------------+-------------+----------+
2 rows in set (0.00 sec)

mysql> #202203103510097
```

6) Delete details of product id 121 from VProduct.

```
mysql> DELETE FROM VProduct WHERE PRODUCT_NO = 121;
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM VProduct;
+------------+----------------+-------+
| PRODUCT_NO | DESCRIPTION    | PRICE |
+------------+----------------+-------+
|        120 | REDUCER        |  1200 |
|        122 | HANDLE         |   700 |
|        124 | WIDGET REMOVER |   900 |
|        136 | SIZE WIDGET    |  1000 |
|        137 | SIZE WIDGET    | 15000 |
|        135 | SOFA           | 35000 |
+------------+----------------+-------+
6 rows in set (0.00 sec)
```

7) Delete view VProduct.

```
mysql> DROP VIEW VPRODUCT;
Query OK, 0 rows affected (0.08 sec)

mysql> SELECT * FROM VPRODUCT;
ERROR 1146 (42S02): Table 'gideon.vproduct' doesn't exist
mysql> #202203103510097
```

8) Display name of all customers and all suppliers with their id by using union operator.

```
mysql> SELECT CUSTOMER_NO AS ID, NAME FROM CUSTOMER
    -> UNION
    -> SELECT SUPPLIER_NO AS ID, NAME FROM SUPPLIER;
+------+--------------+
| ID   | NAME         |
+------+--------------+
|   10 | GARRY SMITH  |
|   20 | PATEL        |
|   30 | DRAKE        |
|   40 | BOB SMITH    |
|   50 | JAMES        |
|   60 | NORTON       |
|   70 | JOHN MICHAEL |
| 1001 | MICHAEL      |
| 1002 | RINGWORLD    |
| 1003 | BABYLON      |
| 1004 | JOHN         |
| 1005 | SMITH        |
+------+--------------+
12 rows in set (0.00 sec)

mysql> #202203103510097
```

9) List product which are not bought by any customer using minus operator.

```
mysql> select PRODUCT_NO, DESCRIPTION
    -> from PRODUCT
    -> where PRODUCT_NO not in (
    -> select distinct PRODUCT_NO from OLINE);
+------------+-------------+
| PRODUCT_NO | DESCRIPTION |
+------------+-------------+
|        137 | SIZE WIDGET |
|        135 | SOFA        |
+------------+-------------+
2 rows in set (0.00 sec)
```

10) Give the name of suppliers who is also customer.

```
mysql> SELECT NAME FROM SUPPLIER
    -> WHERE SUPPLIER_NO IN (
    -> SELECT CUSTOMER_NO FROM CUSTOMER)
    -> UNION
    -> SELECT NAME FROM CUSTOMER
    -> WHERE CUSTOMER_NO IN (SELECT SUPPLIER_NO FROM SUPPLIER);
Empty set (0.01 sec)

mysql> #202203103510097
```

Conclusion: Set operations, CASE statements, and views are essential SQL features that allow you to combine, transform, and present data effectively. By mastering these SQL techniques, you can work with data in a more flexible and efficient manner, providing valuable insights and functionality in database management and application development.