Practical No. 7

Aim: To Perform Simple queries, string manipulation operations implement groups by having.

Theory:

1. **List the number of different products supplied by each supplier_no:** ○ This query involves the use of the `COUNT` function and the `GROUP BY` clause. ○ `COUNT(DISTINCT PRODUCT_NO)` is used to count the unique product numbers supplied by each supplier.
   ○ `GROUP BY SUPPLIER_NO` groups the results by supplier number, allowing you to count products for each supplier separately.
2. **List the name of each supplier with the location of each depot and the number of products supplied by that supplier and stocked at that depot:**
   ○ This query combines data from multiple tables using `JOIN` operations. ○ It utilizes `COUNT(DISTINCT PRODUCT_NO)` again to count the unique products supplied by each supplier.
   ○ The `GROUP BY` clause is used to group results by supplier name, depot location, and depot address.
3. **List the depot_no's of all depots where the average credit_limit for all the customers receiving deliveries from the depot is > 20,000:**
   ○ This query calculates the average (`AVG`) credit limit for each depot.
   ○ It uses the `HAVING` clause to filter depots based on the average credit limit condition.
4. **List the total quantity and product number ordered by each customer:** ○ This query uses the `SUM` function to calculate the total quantity of products ordered by each customer.
   ○ It groups the results by customer number.
5. **Give the product number that has the maximum quantity stocked at any depot:** ○ This query uses the `SUM` function to calculate the total quantity of each product stocked at different depots.
   ○ The `ORDER BY` clause sorts the results in descending order of quantity.
6. **Give the customer address with the minimum credit limit:**
   ○ This query uses a subquery with `MIN(CREDIT_LIMIT)` to find the minimum credit limit.
   ○ It then selects the customer address corresponding to that minimum credit limit.
7. **List supplier numbers who have supplied products whose total price is < 1000:** ○ This query involves subqueries to filter suppliers based on the price condition. ○ It uses `DISTINCT` to ensure that each supplier number appears only once in the result.
8. **Display the total number of customers who have ordered products on the same date:**
   ○ This query involves a subquery to count the number of customers who ordered on the same date.
9. **List the sum of quantities stocked at each rack:**
   ○ This query uses the `SUM` function to calculate the total quantity of products stocked at each rack.
   ○ It groups the results by the rack number.
10. **Display the total number of customers who have received products from the**

same location:

- This query combines data from the `depot` and `customer` tables.
- It counts the number of customers for each location that received products.
- The `HAVING` clause filters locations with more than one customer.

Queries:

(1) List the number of different products supplied by each supplier_no.

```
mysql> /*202203103510124*/
mysql> SELECT SUPPLIER_NO, COUNT(DISTINCT PRODUCT_NO) AS NUM_PRODUCTS_SUPPLIED
    -> FROM PRODUCT
    -> GROUP BY SUPPLIER_NO;
+-------------+------------------------+
| SUPPLIER_NO | NUM_PRODUCTS_SUPPLIED |
+-------------+------------------------+
|        1001 |                      1 |
|        1002 |                      1 |
|        1003 |                      1 |
|        1004 |                      1 |
|        1005 |                      2 |
+-------------+------------------------+
5 rows in set (0.00 sec)
```

(2) List the name of each supplier with the location of each depot and the number of products supplied by that supplier and stocked at that depot.

```
mysql> /*202203103510124*/
mysql> SELECT S.NAME AS SUPPLIER_NAME, D.LOCATION AS DEPOT_LOCATION, COUNT(DISTINCT P.PRODUCT_NO) AS NUM_PRODUCTS_SUPPLIED
    -> FROM SUPPLIER S
    -> JOIN PRODUCT P ON S.SUPPLIER_NO = P.SUPPLIER_NO
    -> JOIN STOCK ST ON P.PRODUCT_NO = ST.PRODUCT_NO
    -> JOIN DEPOTE D ON ST.DEPOT_NO = D.DEPOT_NO
    -> GROUP BY S.NAME, D.LOCATION;
+--------------+---------------+----------------------+
| SUPPLIER_NAME | DEPOT_LOCATION | NUM_PRODUCTS_SUPPLIED |
+--------------+---------------+----------------------+
| BABYLON       | SOUTH         |                    1 |
| JOHN          | WALES         |                    1 |
| MICHAEL       | LONDON        |                    1 |
| RINGWORLD     | SOUTH         |                    1 |
| SMITH         | EAST          |                    1 |
| SMITH         | NORTH         |                    2 |
+--------------+---------------+----------------------+
6 rows in set (0.00 sec)
```

(3) List the depot_no's of all depots where the average credit_limit for all the customers receiving deliveries from the depot is > 20,000.

```
mysql> /*202203103510124*/
mysql> SELECT D.DEPOT_NO
    -> FROM DEPOTE D
    -> JOIN CUSTOMER C ON D.DEPOT_NO = C.DEPOT_NO
    -> GROUP BY D.DEPOT_NO
    -> HAVING AVG(C.CREDIT_LIMIT) > 20000;
Empty set (0.00 sec)
```

(4) List total no of quantity and product number ordered by customer.

```
mysql> /*202203103510124*/
mysql> SELECT C.CUSTOMER_NO, SUM(OL.QUANTITY) AS TOTAL_QUANTITY_ORDERED
    -> FROM CUSTOMER C
    -> JOIN CORDER O ON C.CUSTOMER_NO = O.CUSTOMER_NO
    -> JOIN OLINE OL ON O.CORDER_NO = OL.CORDER_NO
    -> GROUP BY C.CUSTOMER_NO;
+-------------+------------------------+
| CUSTOMER_NO | TOTAL_QUANTITY_ORDERED |
+-------------+------------------------+
|          10 |                     30 |
|          20 |                     10 |
|          30 |                     20 |
|          40 |                     40 |
|          70 |                     15 |
+-------------+------------------------+
5 rows in set (0.00 sec)
```

(5) Give product number which has maximum quantity stock at any depot.

```
mysql> /*202203103510124*/
mysql> SELECT PRODUCT_NO
    -> FROM STOCK
    -> GROUP BY PRODUCT_NO
    -> HAVING SUM(QUANTITY) = (SELECT MAX(TOTAL_QUANTITY) FROM (SELECT PRODUCT_NO, SUM(QUANTITY) AS TOTAL_QUANTITY FROM STOCK GROUP BY PRODUCT_NO) AS
TOTALS);
+------------+
| PRODUCT_NO |
+------------+
|        124 |
+------------+
1 row in set (0.00 sec)
```

(6) Give customer address which has minimum credit limit.

```
mysql> /*202203103510124*/
mysql> SELECT CUSTOMER_NO, ADDRESS
    -> FROM CUSTOMER
    -> ORDER BY CREDIT_LIMIT ASC
    -> LIMIT 1;
+-------------+---------+
| CUSTOMER_NO | ADDRESS |
+-------------+---------+
|          10 | BRIXTON |
+-------------+---------+
1 row in set (0.00 sec)
```

(7) List supplier no who has supplied products whose total price is < 1000.

```
mysql> /*202203103510124*/
mysql> SELECT S.SUPPLIER_NO
    -> FROM SUPPLIER S
    -> JOIN PRODUCT P ON S.SUPPLIER_NO = P.SUPPLIER_NO
    -> GROUP BY S.SUPPLIER_NO
    -> HAVING SUM(P.PRICE) < 1000;
+-------------+
| SUPPLIER_NO |
+-------------+
|        1003 |
+-------------+
1 row in set (0.00 sec)
```

(8) Give total number of customers who has ordered the product on same date.

```
mysql> /*202203103510124*/
mysql> SELECT DATE_PLACED, COUNT(*) AS CUSTOMER_COUNT
    -> FROM CORDER
    -> GROUP BY DATE_PLACED
    -> HAVING COUNT(*) > 1;
+-------------+----------------+
| DATE_PLACED | CUSTOMER_COUNT |
+-------------+----------------+
| 1993-JAN-01 |              2 |
+-------------+----------------+
1 row in set (0.00 sec)
```

(9) List sum of quantity stocked at each rack.

```
mysql> /*202203103510124*/
mysql> SELECT RACK, SUM(QUANTITY) AS TOTAL_QUANTITY_STOCKED
    -> FROM STOCK
    -> GROUP BY RACK;
+------+-------------------------+
| RACK | TOTAL_QUANTITY_STOCKED |
+------+-------------------------+
| 1    |                      50 |
| 10   |                     180 |
| 2    |                      40 |
| 4    |                     120 |
| 5    |                      90 |
| 7    |                      60 |
+------+-------------------------+
6 rows in set (0.00 sec)
```

(10) Display total no of customers who have received product from same location.

```
mysql> /*202203103510124*/
mysql> select d.location, count(distinct c.customer_no) as cust_count from depot d inner join customer c on d.depot_no =
 c.depot_no group by d.location;
+----------------+------------+
| location       | cust_count |
+----------------+------------+
| EAST NZ        |          1 |
| LONDON WEST USA |          1 |
| NORTH KENYA    |          1 |
| NORTH UK       |          1 |
| SOUTH UK       |          1 |
| SOUTH USA      |          1 |
| WALES UK       |          1 |
+----------------+------------+
```

Conclusion:

We learned how to retrieve data, count unique values, and perform aggregations using functions like `COUNT`, `SUM`, and `AVG`. The `GROUP BY` clause allowed us to organize data into meaningful groups, making it invaluable for summarizing information across various categories.

Joining tables using the `JOIN` operation enabled us to link data from different parts of the database, creating a comprehensive view of relationships between entities. Subqueries proved handy for dynamic filtering and calculations, enhancing the flexibility of our queries.