Practical No. 7

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

Theory:
Simple queries involving string manipulation operations, combined with the GROUP BY and HAVING clauses, offer powerful tools for extracting specific data subsets and performing aggregate calculations

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

```
mysql> SELECT SUPPLIER_NO, COUNT(PRODUCT_NO) AS DIFFERENT_PRODUCTS
    -> FROM PRODUCT
    -> GROUP BY SUPPLIER_NO;
+-------------+--------------------+
| SUPPLIER_NO | DIFFERENT_PRODUCTS |
+-------------+--------------------+
|        1005 |                  2 |
|        1004 |                  1 |
|        1003 |                  1 |
|        1001 |                  1 |
|        1002 |                  1 |
+-------------+--------------------+
5 rows in set (0.07 sec)

mysql> #202203103510097
```

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> SELECT S.NAME, D.LOCATION, COUNT(P.PRODUCT_NO)
    -> FROM SUPPLIER S
    -> JOIN PRODUCT P ON S.SUPPLIER_NO = P.SUPPLIER_NO
    -> JOIN DEPOT D ON P.SUPPLY_DEPOT_NO = D.DEPOT_NO
    -> GROUP BY S.NAME, D.LOCATION
    -> ORDER BY S.NAME, D.LOCATION;
+-----------+----------+--------------------+
| NAME      | LOCATION | COUNT(P.PRODUCT_NO) |
+-----------+----------+--------------------+
| BABYLON   | EAST     |                  1 |
| JOHN      | NORTH    |                  1 |
| MICHAEL   | WALES    |                  1 |
| RINGWORLD | SOUTH    |                  1 |
| SMITH     | NORTH    |                  1 |
| SMITH     | SOUTH    |                  1 |
+-----------+----------+--------------------+
6 rows in set (0.10 sec)

mysql> #202203103510097
```

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> SELECT D.DEPOT_NO
    -> FROM DEPOT D
    -> JOIN CUSTOMER C ON D.DEPOT_NO = C.DEPOT_NO
    -> GROUP BY D.DEPOT_NO
    -> HAVING AVG(C.CREDIT_LIMIT) > 10000;
+----------+
| DEPOT_NO |
+----------+
|        5 |
+----------+
1 row in set (0.00 sec)

mysql> #202203103510097
```

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

```
mysql> SELECT SUM(O.QUANTITY), O.PRODUCT_NO
    -> FROM ONLINE O
    -> GROUP BY O.PRODUCT_NO;
+-----------------+------------+
| SUM(O.QUANTITY) | PRODUCT_NO |
+-----------------+------------+
|              10 |        120 |
|              10 |        121 |
|              20 |        122 |
|              60 |        136 |
|              15 |        124 |
+-----------------+------------+
5 rows in set (0.07 sec)

mysql> #202203103510097
```

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

```
mysql> SELECT PRODUCT_NO
    -> FROM STOCK
    -> GROUP BY PRODUCT_NO
    -> HAVING SUM(QUANTITY) = (
    -> SELECT MAX(TOTAL_QUANTITY)
    -> FROM (
    -> SELECT SUM(QUANTITY) AS TOTAL_QUANTITY
    -> FROM STOCK
    -> GROUP BY PRODUCT_NO
    -> ) AS MAX_QUANTITY
    -> );
+------------+
| PRODUCT_NO |
+------------+
|        124 |
+------------+
1 row in set (0.01 sec)

mysql> #202203103510097
```

6) Give customer address which has minimum credit limit.

```
mysql> SELECT C.ADDRESS
    -> FROM CUSTOMER C
    -> GROUP BY C.ADDRESS
    -> HAVING MIN(CREDIT_LIMIT) = (
    -> SELECT MIN(CREDIT_LIMIT)
    -> FROM CUSTOMER
    -> );
+---------+
| ADDRESS |
+---------+
| BRIXTON |
+---------+
1 row in set (0.02 sec)

mysql> #202203103510097
```

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

```
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 (0.01 sec)

mysql> #202203103510097
```

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

```
mysql> SELECT COUNT(DISTINCT CUSTOMER_NO) AS TOTAL_CUSTOMER
    -> FROM CORDER
    -> WHERE DATE_PLACED IN (
    -> SELECT DATE_PLACED
    -> FROM CORDER
    -> GROUP BY DATE_PLACED
    -> HAVING COUNT(*) > 1
    -> );
+----------------+
| TOTAL_CUSTOMER |
+----------------+
|              1 |
+----------------+
1 row in set (0.00 sec)

mysql> #202203103510097
```

9) List sum of quantity stocked at each rack.

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

mysql> #202203103510097
```

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

```
mysql> SELECT D.LOCATION AS DEPOT_LOCATION, COUNT(DISTINCT C.CUSTOMER_NO) AS TOTAL_CUSTOMERS
    -> FROM CUSTOMER C
    -> JOIN DEPOT D ON C.DEPOT_NO = D.DEPOT_NO
    -> GROUP BY D.LOCATION
    -> HAVING COUNT(DISTINCT C.CUSTOMER_NO) > 1;
+----------------+-----------------+
| DEPOT_LOCATION | TOTAL_CUSTOMERS |
+----------------+-----------------+
| NORTH          |               2 |
| SOUTH          |               2 |
+----------------+-----------------+
2 rows in set (0.00 sec)

mysql> #202203103510097
```

Conclusion: Leveraging simple queries, string manipulation operations, and the GROUP BY and HAVING clauses in SQL empowers data analysts and database administrators to perform more complex and targeted data extractions and analyses. By combining these techniques, organizations can gain valuable insights into their data, leading to better decision-making and improved data-driven strategies. Understanding the synergy between these SQL components is a crucial skill for anyone working with relational databases.