Practical No. 6

Aim: To implement Joins.

Theory:

1. **Database Tables:** We worked with several tables, including customer, corder, depot, oline, product, salesrep, stock, and suppliers. Each table represents a different aspect of a business scenario, such as customer data, orders, products, and sales representatives.
2. **SQL Queries:** We used SQL (Structured Query Language) to query the database and retrieve meaningful information. SQL allows us to interact with the database by performing operations like selecting, filtering, joining, and aggregating data.
3. **SELECT Statement:** We frequently used the SELECT statement to specify which columns we wanted to retrieve from a table. This statement is fundamental in SQL and forms the basis of most queries.
4. **JOIN Operations:** We used various types of joins, including INNER JOIN and LEFT JOIN, to combine data from multiple tables based on matching keys. Joins are essential for retrieving related data from different tables.
5. **Filtering Data:** The WHERE clause allowed us to filter rows based on specific conditions. We used it to narrow down our results and retrieve only the data that met certain criteria.
6. **Aggregation:** We employed aggregate functions like SUM, COUNT, and COALESCE to perform calculations on groups of data. Aggregation functions are useful for obtaining summary information, such as total quantities and averages.
7. **Subqueries:** Subqueries were used to create nested queries within our main queries. They helped us retrieve data from one table based on information obtained from another table.
8. **Aliases:** We used aliases to assign temporary names to columns or tables. This made our query results more readable and provided clarity when dealing with complex queries.

we delved into the realm of relational databases and SQL (Structured Query Language). Databases are the backbone of modern information management, and SQL is the language used to interact with them. We explored the fundamental aspects of database querying, including the use of SQL statements like SELECT, JOIN, and WHERE to retrieve, combine, and filter data from multiple tables. We learned how to perform calculations, aggregate data, and use subqueries to tackle complex questions. Additionally, we employed aliases to enhance query readability. These practical exercises provided valuable hands-on experience and insight into the world of relational databases and SQL, which are vital skills for data professionals and anyone seeking to harness the power of data for decision-making and analysis.

Queries:

(1) Give a list of depot locations paired with the name of the sales rep who covers that depot.

```
mysql> /*202203103510124*/
mysql> SELECT D.LOCATION, SR.NAME AS SALES_REP_NAME
    -> FROM DEPOT D
    -> JOIN SALESREP SR ON D.REP_NO = SR.REP_NO;
+-----------------+----------------+
| LOCATION        | SALES_REP_NAME |
+-----------------+----------------+
| NORTH UK        | MIKE           |
| SOUTH USA       | FRED           |
| LONDON WEST USA | ALI            |
| EAST NZ         | SAM            |
| WALES UK        | BILL ADAMS     |
| NORTH KENYA     | SAM            |
| SOUTH UK        | FRED           |
+-----------------+----------------+
```

(2) List the customer name and the depot location for the depot delivering to that customer for all customers who receive deliveries from depots looked after by sales rep number (rep_no) 3.

```
mysql> /*202203103510124*/
mysql> SELECT C.NAME AS CUSTOMER_NAME, D.LOCATION AS DEPOT_LOCATION
    -> FROM CUSTOMER C
    -> JOIN DEPOT D ON C.DEPOT_NO = D.DEPOT_NO
    -> WHERE D.REP_NO = 3;
+---------------+-----------------+
| CUSTOMER_NAME | DEPOT_LOCATION  |
+---------------+-----------------+
| JAMES         | LONDON WEST USA |
+---------------+-----------------+
1 row in set (0.00 sec)
```

(3) List the sales rep number (rep_no) and depot location and address for depots looked after by the sales rep whose name is mike.

```
mysql> /*202203103510124*/
mysql> SELECT SR.REP_NO, D.LOCATION, D.ADDRESS
    -> FROM DEPOT D
    -> JOIN SALESREP SR ON D.REP_NO = SR.REP_NO
    -> WHERE SR.NAME = 'MIKE';
+--------+----------+---------+
| REP_NO | LOCATION | ADDRESS |
+--------+----------+---------+
|      1 | NORTH UK | 1       |
+--------+----------+---------+
1 row in set (0.00 sec)
```

(4) For all order lines (oline) for all orders (corder) for customers whose name is patel, list the customer address, the date_placed, the product_no and the quantity.

```
mysql> /*202203103510124*/
mysql> SELECT C.NAME AS CUSTOMER_NAME, C.ADDRESS AS CUSTOMER_ADDRESS, O.DATE_PLACED, OL.PRODUCT_NO, OL.QUANTITY
    -> FROM CUSTOMER C
    -> JOIN CORDER O ON C.CUSTOMER_NO = O.CUSTOMER_NO
    -> JOIN OLINE OL ON O.CORDER_NO = OL.CORDER_NO
    -> WHERE C.NAME = 'PATEL';
+---------------+------------------+-------------+------------+----------+
| CUSTOMER_NAME | CUSTOMER_ADDRESS | DATE_PLACED | PRODUCT_NO | QUANTITY |
+---------------+------------------+-------------+------------+----------+
| PATEL         | GRANGE           | 1993-01-01  |        120 |        5 |
| PATEL         | GRANGE           | 1993-01-01  |        120 |        5 |
+---------------+------------------+-------------+------------+----------+
```

(5) Give the total number of items (quantity) in stock in all depots.

```
mysql> /*202203103510124*/
mysql> SELECT SUM(QUANTITY) AS TOTAL_STOCK
    -> FROM STOCK;
+-------------+
| TOTAL_STOCK |
+-------------+
|         540 |
+-------------+
```

(6) Give the total number of items (order line quantity) which have been ordered on the order with corder_no 200.

```
mysql> /*202203103510124*/
mysql> SELECT SUM(OL.QUANTITY) AS ORDER_TOTAL
    -> FROM OLINE OL
    -> WHERE OL.CORDER_NO = 200;
+-------------+
| ORDER_TOTAL |
+-------------+
|           5 |
+-------------+
```

(7) List the names of all customers who receive deliveries from depots which are looked after by the sales rep whose name is fred.

```
mysql> /*202203103510124*/
mysql> SELECT DISTINCT C.NAME AS CUSTOMER_NAME
    -> FROM CUSTOMER C
    -> JOIN DEPOT D ON C.DEPOT_NO = D.DEPOT_NO
    -> JOIN SALESREP SR ON D.REP_NO = SR.REP_NO
    -> WHERE SR.NAME = 'FRED';
+---------------+
| CUSTOMER_NAME |
+---------------+
| BOB SMITH     |
| JOHN MICHAEL  |
+---------------+
```

(8) List the customer name, order date_placed, order line quantity and product description for each order line (with its linked, order, customer and product rows) for customers who receive deliveries from depot number 2.

```
mysql> /*202203103510124*/
mysql> SELECT C.NAME AS CUSTOMER_NAME, O.DATE_PLACED, OL.QUANTITY, P.DESCRIPTION AS PRODUCT_DESCRIPTION
    -> FROM CUSTOMER C
    -> JOIN CORDER O ON C.CUSTOMER_NO = O.CUSTOMER_NO
    -> JOIN OLINE OL ON O.CORDER_NO = OL.CORDER_NO
    -> JOIN PRODUCT P ON OL.PRODUCT_NO = P.PRODUCT_NO
    -> WHERE C.DEPOT_NO = 2;
+---------------+-------------+----------+--------------------+
| CUSTOMER_NAME | DATE_PLACED | QUANTITY | PRODUCT_DESCRIPTION |
+---------------+-------------+----------+--------------------+
| BOB SMITH     | 1993-01-17  |       10 | PLATE              |
| BOB SMITH     | 1994-01-01  |       30 | SIZE WIDGET        |
+---------------+-------------+----------+--------------------+
```

(9) List supplier names paired with the names of the sales reps who market products supplied by that supplier.

```
mysql> /*202203103510124*/
mysql> SELECT S.NAME AS SUPPLIER_NAME, SR.NAME AS SALES_REP_NAME
    -> FROM SUPPLIER S
    -> JOIN PRODUCT P ON S.SUPPLIER_NO = P.SUPPLIER_NO
    -> JOIN SALESREP SR ON P.MARKETING_REP_NO = SR.REP_NO;
+---------------+----------------+
| SUPPLIER_NAME | SALES_REP_NAME |
+---------------+----------------+
| SMITH         | BILL ADAMS     |
| JOHN          | ALI            |
| BABYLON       | FRED           |
| SMITH         | SAM            |
| MICHAEL       | MIKE           |
| RINGWORLD     | FRED           |
+---------------+----------------+
```

(10) List supplier names paired with the names of the sales reps who look after the depots where products from that supplier are delivered.

```
mysql> /*202203103510124*/
mysql> SELECT S.NAME AS SUPPLIER_NAME, SR.NAME AS SALES_REP_NAME
    -> FROM SUPPLIER S
    -> JOIN PRODUCT P ON S.SUPPLIER_NO = P.SUPPLIER_NO
    -> JOIN STOCK ST ON P.PRODUCT_NO = ST.PRODUCT_NO
    -> JOIN DEPOT D ON ST.DEPOT_NO = D.DEPOT_NO
    -> JOIN SALESREP SR ON D.REP_NO = SR.REP_NO;
+---------------+----------------+
| SUPPLIER_NAME | SALES_REP_NAME |
+---------------+----------------+
| SMITH         | MIKE           |
| RINGWORLD     | FRED           |
| MICHAEL       | ALI            |
| SMITH         | SAM            |
| JOHN          | BILL ADAMS     |
| SMITH         | SAM            |
| BABYLON       | FRED           |
+---------------+----------------+
```

(11) List the names of all customers who have ordered products which are marketed by the sales rep whose name is ali.

```
mysql> /*202203103510124*/
mysql> SELECT DISTINCT C.NAME AS CUSTOMER_NAME
    -> FROM CUSTOMER C
    -> JOIN CORDER O ON C.CUSTOMER_NO = O.CUSTOMER_NO
    -> JOIN PRODUCT P ON C.CUSTOMER_NO = P.SUPPLIER_NO
    -> JOIN SALESREP SR ON P.MARKETING_REP_NO = SR.REP_NO
    -> WHERE SR.NAME = 'ALI';
Empty set (0.00 sec)
```

(12) List the names of all customers who are delivered to by the depot which delivers to the customer whose name is drake.

```
mysql> /*202203103510124*/
mysql> SELECT DISTINCT C1.NAME AS CUSTOMER_NAME
    -> FROM CUSTOMER C1
    -> JOIN DEPOT D1 ON C1.DEPOT_NO = D1.DEPOT_NO
    -> JOIN DEPOT D2 ON D1.LOCATION = D2.LOCATION
    -> JOIN CUSTOMER C2 ON D2.DEPOT_NO = C2.DEPOT_NO
    -> WHERE C2.NAME = 'DRAKE';
+---------------+
| CUSTOMER_NAME |
+---------------+
| DRAKE         |
+---------------+
```

(13) List each product description and its price increased by 10%.

```
mysql> /*202203103510124*/
mysql> SELECT P.DESCRIPTION, P.PRICE * 1.1 AS INCREASED_PRICE
    -> FROM PRODUCT P;
+----------------+-----------------+
| DESCRIPTION    | INCREASED_PRICE |
+----------------+-----------------+
| REDUCER        |        1320.000 |
| PLATE          |        1650.000 |
| HANDLE         |         770.000 |
| WIDGET REMOVER |         990.000 |
| SIZE WIDGET    |        1100.000 |
| SIZE WIDGET    |       16500.000 |
+----------------+-----------------+
```

(14) List all order lines for the customer with customer_no 20 giving the product description, the order line quantity and the value of the order line. (i.e. the order line quantity * the price from the linked product row)

```
mysql> /*202203103510124*/
mysql> SELECT P.DESCRIPTION, OL.QUANTITY, P.PRICE * OL.QUANTITY AS ORDER_LINE_VALUE
    -> FROM CUSTOMER C
    -> JOIN CORDER O ON C.CUSTOMER_NO = O.CUSTOMER_NO
    -> JOIN OLINE OL ON O.CORDER_NO = OL.CORDER_NO
    -> JOIN PRODUCT P ON OL.PRODUCT_NO = P.PRODUCT_NO
    -> WHERE C.CUSTOMER_NO = 20;
+-------------+----------+------------------+
| DESCRIPTION | QUANTITY | ORDER_LINE_VALUE |
+-------------+----------+------------------+
| REDUCER     |        5 |          6000.00 |
| REDUCER     |        5 |          6000.00 |
+-------------+----------+------------------+
```

(15) List the locations and addresses of all depots which do not stock product number 122. (ie where there is no stock row for that product for the depot)

```
mysql> /*202203103510124*/
mysql> SELECT D.LOCATION, D.ADDRESS
    -> FROM DEPOT D
    -> LEFT JOIN STOCK S ON D.DEPOT_NO = S.DEPOT_NO AND S.PRODUCT_NO = 122
    -> WHERE S.PRODUCT_NO IS NULL;
+-----------------+---------+
| LOCATION        | ADDRESS |
+-----------------+---------+
| NORTH UK        | 1       |
| SOUTH USA       | 2       |
| LONDON WEST USA | 3       |
| EAST NZ         | 4       |
| WALES UK        | 5       |
| NORTH KENYA     | 6       |
+-----------------+---------+
```

(16) Set up a query which lists the names of all customers who have placed an order with the order number (corder_no) of the order merged with the names of all customers who have never placed an order (shown once, with the order number attribute null) i.e. an outer join.

```
mysql> /*202203103510124*/
mysql> SELECT DISTINCT C1.NAME AS CUSTOMER_NAME, C2.NAME AS MERGED_CUSTOMER_NAME
    -> FROM CUSTOMER C1
    -> LEFT JOIN CORDER O ON C1.CUSTOMER_NO = O.CUSTOMER_NO
    -> LEFT JOIN CUSTOMER C2 ON O.CUSTOMER_NO = C2.CUSTOMER_NO;
+---------------+----------------------+
| CUSTOMER_NAME | MERGED_CUSTOMER_NAME |
+---------------+----------------------+
| GARRY SMITH   | GARRY SMITH          |
| PATEL         | PATEL                |
| DRAKE         | DRAKE                |
| BOB SMITH     | BOB SMITH            |
| JAMES         | NULL                 |
| NORTON        | NULL                 |
| JOHN MICHAEL  | JOHN MICHAEL         |
+---------------+----------------------+
```

Conclusion:

In this series of practical exercises, we gained hands-on experience with SQL, a powerful language for managing and querying relational databases. We learned how to retrieve data from multiple tables, join data together, filter results, and perform calculations. These skills

are fundamental for anyone working with databases, whether in a professional context or for personal projects.

We also explored various real-world scenarios, such as customer orders, product management, and sales representatives, which helped us apply SQL concepts in practical situations. By practicing these exercises, we've developed a strong foundation for working with databases and have learned how to extract valuable insights from data.

Overall, these practical exercises provide a solid introduction to SQL and relational databases, which are essential tools in today's data-driven world.