Practical No. 4

Aim: To implement Integrity Constraints. Queries (along with sub Queries)

Theory:

The practical exercises you've been working on involve SQL queries and operations on a set of database tables. Below, I'll provide a comprehensive theory section that covers the key concepts and SQL statements used in these exercises:

SQL (Structured Query Language): SQL is a domain-specific language used for managing and manipulating relational databases. It's a standard for communicating with and querying databases.

Relational Database: A relational database is a structured collection of data organized into tables with rows and columns, where each row represents a record, and each column represents an attribute.

SELECT Statement: The SELECT statement is used to retrieve data from one or more database tables. It allows you to specify which columns to retrieve and can include conditions for filtering the data.

FROM Clause: The FROM clause in a SELECT statement specifies the tables from which to retrieve data.

WHERE Clause: The WHERE clause is used to filter rows based on specified conditions. It acts as a condition or filter applied to the rows in the result set.

INNER JOIN: An INNER JOIN is used to combine rows from two or more tables based on a related column between them. It returns only the rows where there is a match in both tables.

LEFT JOIN (or LEFT OUTER JOIN): A LEFT JOIN returns all rows from the left table and the matched rows from the right table. If there is no match, NULL values are returned for the right table's columns.

RIGHT JOIN (or RIGHT OUTER JOIN): A RIGHT JOIN is similar to a LEFT JOIN but returns all rows from the right table and the matched rows from the left table.

FULL JOIN (or FULL OUTER JOIN): A FULL JOIN returns all rows when there is a match in either the left or right table. If there's no match, NULL values are returned.

GROUP BY Clause: The GROUP BY clause is used to group rows that have the same values in specified columns into summary rows.

HAVING Clause: The HAVING clause is used in combination with GROUP BY to filter grouped rows based on specified conditions.

ORDER BY Clause: The ORDER BY clause is used to sort the result set based on one or more columns, either in ascending (ASC) or descending (DESC) order.

Aggregate Functions: Aggregate functions perform a calculation on a set of values and return a single value. Common aggregate functions include COUNT, SUM, AVG, MIN, and MAX.

SUBQUERY: A subquery, also known as a nested query, is a query embedded within another query. It can be used to retrieve data that will be used as a condition in the outer query.

EXISTS: The EXISTS condition is used to test whether a subquery returns any rows. If the subquery returns at least one row, the condition is true.

UNION Operator: The UNION operator is used to combine the result sets of two or more SELECT statements into a single result set, removing duplicate rows.

EXCEPT Operator: The EXCEPT operator is used to return the distinct rows from the left SELECT statement that are not present in the right SELECT statement.

Queries:

(1) List the description of product which are supplied by supplier SMITH using

IN.

```
mysql> /*202203103510124*/
mysql> SELECT P.DESCRIPTION
    -> FROM PRODUCT P
    -> WHERE P.SUPPLIER_NO IN (SELECT SUPPLIER_NO FROM SUPPLIER WHERE NAME = 'SMITH');
+----------------+
| DESCRIPTION    |
+----------------+
| REDUCER        |
| WIDGET REMOVER |
+----------------+
2 rows in set (0.01 sec)
```

(2) List all product no which are not ordered by the customer having same CORDER_NO

As the CUSTOMER_NO 20.

```
mysql> /*202203103510124*/
mysql> SELECT DISTINCT P.PRODUCT_NO
    -> FROM PRODUCT P
    -> WHERE P.PRODUCT_NO NOT IN (SELECT OL.PRODUCT_NO
    ->                            FROM OLINE OL
    ->                            JOIN CORDER O ON OL.CORDER_NO = O.CORDER_NO
    ->                            WHERE O.CUSTOMER_NO = 20);
+------------+
| PRODUCT_NO |
+------------+
|        121 |
|        122 |
|        124 |
|        136 |
|        137 |
+------------+
5 rows in set (0.01 sec)
```

(3) List the locations and addresses of all depots which stock any product which is supplied to the depot whose location is wales.

```
mysql> /*202203103510124*/
mysql> SELECT D.LOCATION, D.ADDRESS
    -> FROM DEPOT D
    -> WHERE EXISTS (SELECT 1
    ->               FROM STOCK S
    ->               JOIN PRODUCT P ON S.PRODUCT_NO = P.PRODUCT_NO
    ->               WHERE S.DEPOT_NO = D.DEPOT_NO
    ->                 AND P.SUPPLIER_NO IN (SELECT SUPPLIER_NO
    ->                                       FROM SUPPLIER
    ->                                       WHERE LOCATION = 'WALES'));
Empty set (0.00 sec)
```

(4) List the customer_no, date_placed and date_delivered for all orders which contain order lines for the product with product_no 137 using existential quantification (ie the where exists condition).

```
mysql> /*202203103510124*/
mysql> SELECT O.CUSTOMER_NO, O.DATE_PLACED, O.DATE_DELIVERED
    -> FROM CORDER O
    -> WHERE EXISTS (SELECT 1
    ->               FROM OLINE OL
    ->               WHERE OL.CORDER_NO = O.CORDER_NO
    ->                 AND OL.PRODUCT_NO = 137);
Empty set (0.00 sec)
```

(5) List the depots which do not stock any product supplied by the supplier whose name is ringworld.

```
mysql> /*202203103510124*/
mysql> SELECT D.LOCATION, D.ADDRESS
    -> FROM DEPOT D
    -> WHERE D.DEPOT_NO NOT IN (SELECT DISTINCT S2.DEPOT_NO
    ->                          FROM STOCK S2
    ->                          JOIN PRODUCT P2 ON S2.PRODUCT_NO = P2.PRODUCT_NO
    ->                          WHERE P2.SUPPLIER_NO IN (SELECT SUPPLIER_NO
    ->                                                   FROM SUPPLIER
    ->                                                   WHERE NAME = 'RINGWORLD'));
+-----------------+---------+
| LOCATION        | ADDRESS |
+-----------------+---------+
| NORTH UK        | 1       |
| LONDON WEST USA | 3       |
| EAST NZ         | 4       |
| WALES UK        | 5       |
| NORTH KENYA     | 6       |
| SOUTH UK        | 2       |
+-----------------+---------+
6 rows in set (0.00 sec)
```

(6) List the locations and addresses of all depots which stock all products supplied by the supplier babylon 5.

```
mysql> /*202203103510124*/
mysql> SELECT D.LOCATION, D.ADDRESS
    -> FROM DEPOT D
    -> WHERE NOT EXISTS (SELECT P3.PRODUCT_NO
    ->                   FROM PRODUCT P3
    ->                   WHERE P3.SUPPLIER_NO = (SELECT SUPPLIER_NO
    ->                                           FROM SUPPLIER
    ->                                           WHERE NAME = 'BABYLON 5')
    ->                     AND P3.PRODUCT_NO NOT IN (SELECT S3.PRODUCT_NO
    ->                                               FROM STOCK S3
    ->                                               WHERE S3.DEPOT_NO = D.DEPOT_NO));
+-----------------+---------+
| LOCATION        | ADDRESS |
+-----------------+---------+
| NORTH UK        | 1       |
| SOUTH USA       | 2       |
| LONDON WEST USA | 3       |
| EAST NZ         | 4       |
| WALES UK        | 5       |
| NORTH KENYA     | 6       |
| SOUTH UK        | 2       |
+-----------------+---------+
7 rows in set (0.00 sec)
```

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

```
mysql> /*202203103510124*/
mysql> SELECT S.SUPPLIER_NO, COUNT(DISTINCT P4.PRODUCT_NO) AS NUM_PRODUCTS_SUPPLIED
    -> FROM SUPPLIER S
    -> LEFT JOIN PRODUCT P4 ON S.SUPPLIER_NO = P4.SUPPLIER_NO
    -> GROUP BY S.SUPPLIER_NO;
+-------------+-----------------------+
| SUPPLIER_NO | NUM_PRODUCTS_SUPPLIED |
+-------------+-----------------------+
|        1001 |                     1 |
|        1002 |                     1 |
|        1003 |                     1 |
|        1004 |                     1 |
|        1005 |                     2 |
+-------------+-----------------------+
5 rows in set (0.00 sec)
```

(8) 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 P5.PRODUCT_NO) AS NUM_PRODUCTS_STOCKED
    -> FROM SUPPLIER S
    -> JOIN PRODUCT P5 ON S.SUPPLIER_NO = P5.SUPPLIER_NO
    -> JOIN STOCK S4 ON P5.PRODUCT_NO = S4.PRODUCT_NO
    -> JOIN DEPOT D ON S4.DEPOT_NO = D.DEPOT_NO
    -> GROUP BY S.NAME, D.LOCATION;
+---------------+----------------+----------------------+
| SUPPLIER_NAME | DEPOT_LOCATION | NUM_PRODUCTS_STOCKED |
+---------------+----------------+----------------------+
| BABYLON       | SOUTH UK       |                    1 |
| JOHN          | WALES UK       |                    1 |
| MICHAEL       | LONDON WEST USA |                   1 |
| RINGWORLD     | SOUTH USA      |                    1 |
| SMITH         | EAST NZ        |                    1 |
| SMITH         | NORTH KENYA    |                    1 |
| SMITH         | NORTH UK       |                    1 |
+---------------+----------------+----------------------+
7 rows in set (0.00 sec)
```

(9) List all product descriptions with the product's supplie name, sorted by product description within supplier name(i.e. all products for a supplier listed together in alphabetic order).

```
mysql> /*202203103510124*/
mysql> SELECT P6.DESCRIPTION AS PRODUCT_DESCRIPTION, S3.NAME AS SUPPLIER_NAME
    -> FROM PRODUCT P6
    -> JOIN SUPPLIER S3 ON P6.SUPPLIER_NO = S3.SUPPLIER_NO
    -> ORDER BY S3.NAME, P6.DESCRIPTION;
+---------------------+---------------+
| PRODUCT_DESCRIPTION | SUPPLIER_NAME |
+---------------------+---------------+
| HANDLE              | BABYLON       |
| PLATE               | JOHN          |
| SIZE WIDGET         | MICHAEL       |
| SIZE WIDGET         | RINGWORLD     |
| REDUCER             | SMITH         |
| WIDGET REMOVER      | SMITH         |
+---------------------+---------------+
6 rows in set (0.00 sec)
```

(10) Display customer name who has ordered on same date.

```
mysql> /* EN.NO - 202203103510223 */
mysql> SELECT
    ->     C1.NAME AS CustomerName1,
    ->     C2.NAME AS CustomerName2,
    ->     O1.DATE_PLACED AS OrderDate
    -> FROM
    ->     corder AS O1
    -> INNER JOIN
    ->     corder AS O2 ON O1.DATE_PLACED = O2.DATE_PLACED
    -> INNER JOIN
    ->     customer AS C1 ON O1.CUSTOMER_NO = C1.CUSTOMER_NO
    -> INNER JOIN
    ->     customer AS C2 ON O2.CUSTOMER_NO = C2.CUSTOMER_NO
    -> WHERE
    ->     O1.CORDER_NO <> O2.CORDER_NO
    -> ORDER BY
    ->     O1.DATE_PLACED;
+---------------+---------------+-------------+
| CustomerName1 | CustomerName2 | OrderDate   |
+---------------+---------------+-------------+
| PATEL         | PATEL         | 01-JAN-1993 |
| PATEL         | PATEL         | 01-JAN-1993 |
+---------------+---------------+-------------+
2 rows in set (0.01 sec)
```

Conclusion:

These SQL exercises cover a range of essential SQL concepts and operations, including data retrieval, filtering, joining tables, aggregating data, and using subqueries. Understanding these concepts and practicing SQL queries is crucial for effectively working with relational databases and extracting meaningful insights from data.