Experiment No: 7

Title : Study & Implementation of

- Sub queries
- Views

***Objective*:**

- To perform nested Queries and joining Queries using DML command
- To understand the implementation of views.

***Theory*:**

SUBQUERIES: The query within another is known as a subquery. A statement containing a subquery is called a parent statement. The rows returned by subquery are used by the parent statement or in other words A subquery is a SELECT statement that is embedded in a clause of another SELECT statement

You can place the subquery in a number of SQL clauses:

- WHERE clause
- HAVING clause
- FROM clause
- OPERATORS(IN.ANY,ALL,<,>,>=,<= etc..)

Types

1. Sub queries that return several values

Sub queries can also return more than one value. Such results should be made use along with the operators in and any.

2. Multiple queries

Here more than one subquery is used. These multiple sub queries are combined by means of 'and' & 'or' keywords.

3. Correlated subquery

A subquery is evaluated once for the entire parent statement whereas a correlated Subquery is evaluated once per row processed by the parent statement.

VIEW: In SQL, a view is a virtual table based on the result-set of an SQL statement. A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

A view is a virtual table, which consists of a set of columns from one or more tables. It is

similar to a table but it does not store in the database. View is a query stored as an object.

Syntax: CREATE VIEW <view_name> AS SELECT <set of fields>
FROM relation_name WHERE (Condition)

Example:

SQL> CREATE VIEW employee AS SELECT empno,ename,job FROM EMP
WHERE job = 'clerk';

SQL> View created.

Example:

CREATE VIEW [Current Product List] AS

SELECT ProductID, ProductName

FROM Products

WHERE Discontinued=No;

UPDATING A VIEW : A view can updated by using the following syntax :
Syntax : CREATE OR REPLACE VIEW view_name AS

SELECT column_name(s)

FROM table_name

WHERE condition

DROPPING A VIEW: A view can be deleted with the DROP VIEW command. **Syntax**: DROP VIEW <view_name> ;

LAB PRACTICE ASSIGNMENT:

Consider the following schema:

Sailors (sid, sname, rating, age)

Boats (bid, bname, color)

Reserves (sid, bid, day(date))

Write a subquery statement for the following queries.

1. Find all information of sailors who have reserved boat number 101.

```
mysql> SELECT *
    -> FROM Sailors
    -> WHERE sid IN (SELECT sid FROM Reserves WHERE bid = 101);
+-----+-------+--------+------+
| sid | sname | rating | age  |
+-----+-------+--------+------+
|   1 | Alice |      7 | 22.0 |
|   2 | Bob   |      5 | 19.5 |
+-----+-------+--------+------+
2 rows in set (0.00 sec)
```

2. Find the name of the boat reserved by Bob.

```
mysql> SELECT bname
    -> FROM Boats
    -> WHERE bid IN (SELECT bid FROM Reserves WHERE sid = (SELECT sid FROM Sailors WHERE sname = 'Bob'));
+--------------+
| bname        |
+--------------+
| Sea Breeze   |
| Ocean Spirit |
+--------------+
2 rows in set (0.01 sec)
```

3. Find the names of sailors who have reserved a red boat, and list in the order of age.

```
mysql> SELECT sname
    -> FROM Sailors
    -> WHERE sid IN (SELECT sid FROM Reserves WHERE bid IN (SELECT bid FROM Boats WHERE color = 'red'))
    -> ORDER BY age;
+---------+
| sname   |
+---------+
| Bob     |
| Alice   |
| Charlie |
+---------+
3 rows in set (0.00 sec)
```

4. Find the names of sailors who have reserved at least one boat.

```
mysql> SELECT DISTINCT sname
    -> FROM Sailors
    -> WHERE sid IN (SELECT sid FROM Reserves);
+---------+
| sname   |
+---------+
| Alice   |
| Bob     |
| Charlie |
| David   |
| Eva     |
+---------+
5 rows in set (0.01 sec)
```

5. Find the ids and names of sailors who have reserved two different boats on the same day.

```
mysql> SELECT S.sid, S.sname
    -> FROM Sailors S
    -> WHERE S.sid IN (
    ->     SELECT R1.sid
    ->     FROM Reserves R1, Reserves R2
    ->     WHERE R1.sid = R2.sid
    ->     AND R1.bid != R2.bid
    ->     AND R1.day = R2.day
    -> );
Empty set (0.00 sec)
```

6. Find the ids of sailors who have reserved a red boat or a green boat.

```
mysql> SELECT DISTINCT sid
    -> FROM Reserves
    -> WHERE bid IN (SELECT bid FROM Boats WHERE color = 'red' OR color = 'green');
+------+
| sid  |
+------+
|    1 |
|    2 |
|    4 |
|    3 |
|    5 |
+------+
5 rows in set (0.00 sec)
```

7. Find the name and the age of the youngest sailor.

```
mysql> SELECT sname, age
    -> FROM Sailors
    -> ORDER BY age ASC
    -> LIMIT 1;
+-------+------+
| sname | age  |
+-------+------+
| Bob   | 19.5 |
+-------+------+
1 row in set (0.01 sec)
```

8. Count the number of different sailor names.

```
mysql> SELECT COUNT(DISTINCT sname)
    -> FROM Sailors;
+-----------------------+
| COUNT(DISTINCT sname) |
+-----------------------+
|                     5 |
+-----------------------+
1 row in set (0.00 sec)
```

9. Find the average age of sailors for each rating level.

```
mysql> SELECT rating, AVG(age) AS avg_age
    -> FROM Sailors
    -> GROUP BY rating;
+--------+----------+
| rating | avg_age  |
+--------+----------+
|      7 | 26.00000 |
|      5 | 19.50000 |
|      8 | 35.20000 |
|      6 | 25.10000 |
+--------+----------+
4 rows in set (0.00 sec)
```

10. Find the average age of sailors for each rating level that has at least two sailors.

```
mysql> SELECT rating, AVG(age) AS avg_age
    -> FROM Sailors
    -> GROUP BY rating
    -> HAVING COUNT(*) >= 2;
+--------+----------+
| rating | avg_age  |
+--------+----------+
|      7 | 26.00000 |
+--------+----------+
1 row in set (0.00 sec)
```