Assignment No. 4

Aim: Implement nested sub queries. Perform a test for set membership (in, not in), set comparison (<some, >=some, <all etc.) and set cardinality (unique, not unique).

Objective:

- To learn different types of Joins.
- To implement different sub queries.

Theory :

MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

There are three types of MySQL joins:

- MySQL INNER JOIN (or sometimes called simple join)
- MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

MySQL Inner JOIN (Simple Join)

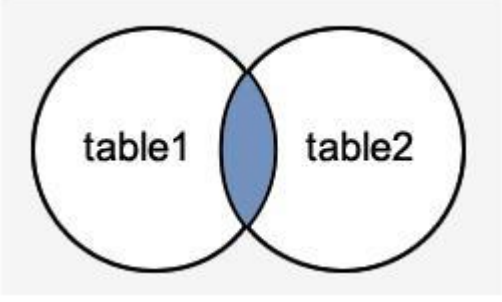
The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

Syntax:

```
SELECT columns
FROM table1

INNER JOIN table2
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.
Execute the following query:

```
SELECT officers.officer_name, officers.address, students.course_name
FROM officers
INNER JOIN students
ON officers.officer_id = students.student_id;
```

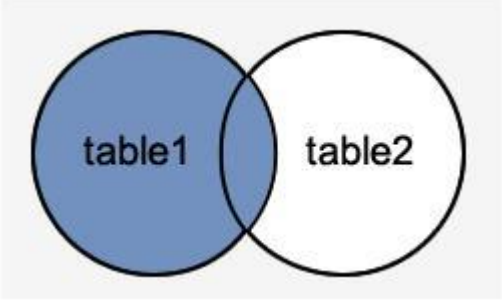
MySQL Left Outer Join

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

Syntax:

```
SELECT columns
FROM table1
LEFT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.
Execute the following query:

```
SELECT officers.officer_name, officers.address, students.course_name
FROM officers
LEFT JOIN students
ON officers.officer_id = students.student_id;
```

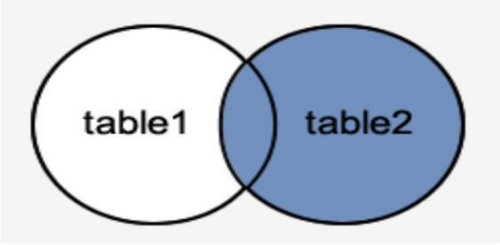
MySQL Right Outer Join

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

Syntax:

```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.
Execute the following query:

```
SELECT      officers.officer_name,      officers.address,
            students.course_name,
students.student_name
FROM officers
RIGHT JOIN students
ON officers.officer_id = students.student_id;
```

SPECIAL OPERATOR:

MySQL IN Condition

The MySQL IN condition is used to reduce the use of multiple OR conditions in a SELECT, INSERT, UPDATE and DELETE statement.

Syntax:

expression IN (value1, value2, ... value_n);

Parameters:

expression: It specifies a value to test.
value1, value2, ... or value_n: These are the values to test against expression. If any of these values matches expression, then the IN condition will evaluate to true. This is a quick method to test if any one of the values matches expression.

Execute the following query:

```
SELECT *

FROM officers
WHERE officer_name IN ('Ajeet', 'Vimal', 'Deepika');
```

MySQL NOT Condition

The MySQL NOT condition is opposite of MySQL IN condition. It is used to negate a condition in a SELECT, INSERT, UPDATE or DELETE statement.

Syntax:

NOT condition

Parameter:

condition: It specifies the conditions that you want to negate.

MySQL NOT Operator with IN condition

Consider a table "officers", having the following data.
Execute the following query:

```
SELECT *

FROM officers
WHERE officer_name NOT IN ('Ajeet','Vimal','Deepika');
```

MySQL IS NULL Condition

MySQL IS NULL condition is used to check if there is a NULL value in the expression. It is used with SELECT, INSERT, UPDATE and DELETE statement.

Syntax:

expression IS NULL

Parameter:

expression: It specifies a value to test if it is NULL
Execute the following query:

```
SELECT *
FROM officers
WHERE officer_name IS NULL;
```

MySQL IS NOT NULL Condition

MySQL IS NOT NULL condition is used to check the NOT NULL value in the expression. It is used with SELECT, INSERT, UPDATE and DELETE statements.

Syntax:

expression IS NOT NULL

Parameter:

expression: It specifies a value to test if it is not NULL value.

Execute the following query:

```
SELECT *
FROM officers
WHERE officer_name IS NOT NULL;
```

SET OPERATORS:

The Set operator combines the result of 2 queries into a single result. The following are the operators:

- Union
- Union all
- Intersect
- Minus

LAB PRACTICE ASSIGNMENT:

Consider the following table structure for this assignment:

- Location(Location_Id integer, Reginal_Group varchar(20))
- Department (Department_Id, Name, Location_Id)
- Job(Job_Id Integer,Function Varchar(30))
- Employee(Employee_Id, Lastname ,Firstname, Middlename, Job_Id, Manager_Id, Hiredate, Salary, Department_Id)
- Loan(Employee_Id, Firstname , Loan_Amount)

Perform the following queries on the above table:

1) Perform all types of JOIN operations on Employee and Loan tables.
2) Perform all types of set operations on Employee and Loan tables.
3) Find out no.of employees working in "Sales" department
4) Find out the employees who are not working in department 10 or 30.
5) List out employee id, last name in descending order based on the salary column.
6) How many employees who are working in different departments wise in the organization
7) List out the department id having at least four employees
8) Display the employee who got the maximum salary.
9) Update the employees' salaries, who are working as Clerk on the basis of 10%.
10) Delete the employees who are working in accounting department.
11) Find out whose department has not employees.

12) List out the department wise maximum salary, minimum salary, average salary of the employees

13) How many employees who are joined in 1985.

14) Display the employees who are working in “New York”

15) List our employees with their department names