Assignment No. 7

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 ofJoins.
- To implement different subqueries.

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 simplejoin)
- MySQL LEFT OUTER JOIN (or sometimes called LEFTJOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHTJOIN)

MySQL Inner JOIN (Simple Join)

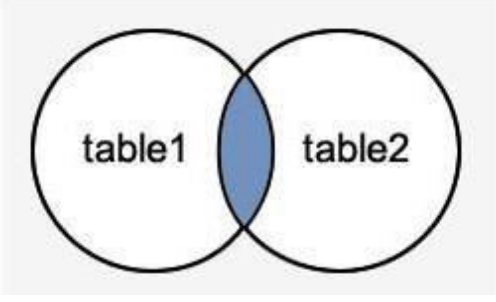
The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

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;
```

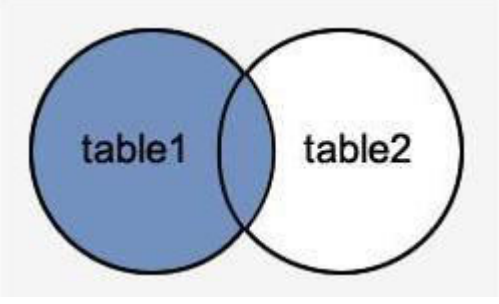
MySQL Left Outer Join

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;
```

Imagerepresentation:



Let's take anexample:

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

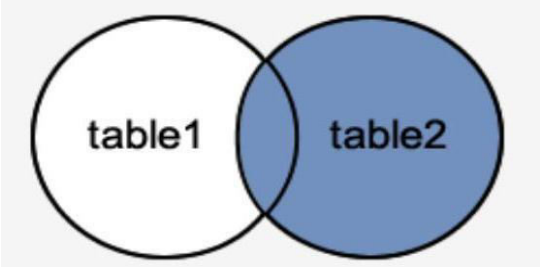
The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

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 ofthese

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

Unionall

- Intersect
- Minus

LAB PRACTICE ASSIGNMENT:

Consider the following table structure for this assignment:

- Location(Location_Id integer, Reginal_Groupvarchar(20))
- Department (Department_Id, Name,Location_Id)
- Job(Job_IdInteger,FunctionVarchar(30))
- Employee(Employee_Id, Lastname ,Firstname, Middlename, Job_Id, Manager_Id, Hiredate, Salary,Department_Id)
- Loan(Employee_Id, Firstname , Loan_Amount)

LOCATION TABLE

LOCATION_ID	REGINAL_GROUP
122	New York
123	Dallas
124	Chicago
167	Boostan

DEPARTMENT TABLE

DEPARTMENT_ID	NAME	LOCATION_ID

10	Accounting	122

20	Research	124
30	Sale	123
40	Operation	164

JOB TABLE

JOB_ID	FUNCTION
667	Cleark
668	Staff
669	Analyst
670	Saleperson
671	Manager
672	President

EMPLOYEE TABLE

EMPLOYEE_ID	LASTNAME	FIRSTNAME	MIDDLENAME	JOB_ID	MANAGER_ID	HIREDATE	SALARY	DEPARTMENT_ID
7369	Smith	Jon	Q	667	7902	17-DEC-84	800	10
7499	Allen	Kevin	J	670	7698	20-FEB-85	1600	20
7505	Doyle	Jean	K	671	7839	04-APR-85	2850	20
7506	Dennis	Lynn	S	671	7839	15-MAY-85	2750	30
7507	Baker	Leslie	D	671	7839	10-JUN-85	2200	40
7521	wark	cynthia	D	670	7698	22-FEB-85	1250	10

Perform the following queries on the above table:

1) Perform all types of JOIN operations on Employee and Loantables.

2) Perform all types of set operations on Employee and Loantables.
3) Find out no.of employees working in “Sales”department
4) Find out the employees who are not working in department 10 or30.
5) List out employee id, last name in descending order based on the salarycolumn.
6) How many employees who are working in different departments wise in the organization
7) List out the department id having at least fouremployees
8) Display the employee who got the maximumsalary.
9) Update the employees’ salaries, who are working as Clerk on the basis of10%.
10) Delete the employees who are working in accountingdepartment.
11) Find out whose department has notemployees.
12) List out the department wise maximum salary, minimum salary, average salary ofthe employees
13) How many employees who are joined in1985.
14) Display the employees who are working in “NewYork”
15) List our employees with their departmentnames

Conclusion:

We have implemented join, set operations, set cardinalities and nested sub queries.

OUTPUT -

```
mysql> use dbms;
Database changed
```

-CREATE TABLE LOCATION -

```
mysql> CREATE TABLE LOCATION(LOCATION_ID INT, REGINAL_GROUP VARCHAR(10));
Query OK, 0 rows affected (0.12 sec)
```

-INSERT INTO LOCATION TABLE –

```
mysql> INSERT INTO `location` (`LOCATION_ID`, `REGINAL_GROUP`) VALUES ('122', 'NEW YORK'),
('123', 'DALLAS'), ('124', 'CHICAGO'), ('167', 'BOOSTAN');
Query OK, 4 rows affected (0.06 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from location;
+-------------+---------------+
| LOCATION_ID | REGINAL_GROUP |
+-------------+---------------+
|         122 | NEW YORK      |
|         123 | DALLAS        |
|         124 | CHICAGO       |
|         167 | BOOSTAN       |
+-------------+---------------+
4 rows in set (0.00 sec)
```

-CREATE TABLE DEPARTMENT -

```
mysql> CREATE TABLE DEPARTMENT(DEPARTMENT_ID INT, NAME VARCHAR(15), LOCATION_ID INT);
Query OK, 0 rows affected (0.08 sec)
```

-INSERT INTO DEPARTMENT TABLE-

```
mysql> INSERT INTO department VALUES ('10', 'ACCOUNTING', '122'), ('20', 'RESEARCH', '124'), ('30',
'SALE', '123'), ('40', 'OPERATION', '164');
Query OK, 4 rows affected (0.02 sec)
Records: 4  Duplicates: 0  Warnings: 0
```

```
mysql> select * from department;
+---------------+------------+-------------+
| DEPARTMENT_ID | NAME       | LOCATION_ID |
+---------------+------------+-------------+
|          10 | ACCOUNTING |        122 |
|          20 | RESEARCH   |        124 |
|          30 | SALE       |        123 |
|          40 | OPERATION  |        164 |
+---------------+------------+-------------+
4 rows in set (0.00 sec)
```

-CREATE TABLE JOB -

```
mysql> CREATE TABLE JOB(JOB_ID INT (30), FUNC VARCHAR (15));
Query OK, 0 rows affected, 1 warning (0.36 sec)
```

-INSERT INTO JOB-

```
mysql> INSERT INTO job VALUES ('667', 'CLERK'), ('668', 'STAFF'), ('669', 'ANALYST'), ('670',
'SALEPERSON'), ('671', 'MANAGER'), ('672', 'PRESIDENT');
Query OK, 6 rows affected (0.03 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> select * from job;
+--------+------------+
| JOB_ID | FUNC       |
+--------+------------+
|   667 | CLERK      |
|   668 | STAFF      |
|   669 | ANALYST    |
|   670 | SALEPERSON |
|   671 | MANAGER    |
|   672 | PRESIDENT  |
+--------+------------+
```

6 rows in set (0.00 sec)

-CREATE TABLE EMPLOYEE-

mysql> CREATE TABLE EMPLOYEE(EMPLOYEE_ID INT, LASTNAME VARCHAR(15), FIRSTNAME VARCHAR(15), MIDDLENAME VARCHAR(15), JOB_ID INT, MANAGER_ID INT, HIREDATE VARCHAR(15), SALARY INT, DEPARTMENT_ID INT);

Query OK, 0 rows affected (0.15 sec)

-**INSERT INTO EMPLOYEE-**

mysql> INSERT INTO employee VALUES ('7369', 'SMITH', 'JON', 'Q', '667', '7902', '17-DEC-84', '800', '10'), ('7499', 'ALLEN', 'KEVIN', 'J', '670', '7698', '20-FEB-85', '1600', '20'), ('7505', 'DOYLE', 'JEAN', 'K', '671', '7839', '04-APR-85', '2850', '20'), ('7506', 'DENNIS', 'LYNN', 'S', '671', '7839', '15-MAY-85', '2750', '30'), ('7507', 'BAKER', 'LESLIE', 'D', '671', '7839', '10-JUN-85', '2200', '40'), ('7521', 'WARK', 'CYNTHIA', 'D', '670', '7698', '22-FEB-85', '1250', '10');

Query OK, 6 rows affected (0.02 sec)

Records: 6 Duplicates: 0 Warnings: 0

mysql> select * from employee;

+-------------+----------+-----------+------------+--------+------------+-----------+--------+---------------+

| EMPLOYEE_ID | LASTNAME | FIRSTNAME | MIDDLENAME | JOB_ID | MANAGER_ID | HIREDATE | SALARY | DEPARTMENT_ID |

+-------------+----------+-----------+------------+--------+------------+-----------+--------+---------------+

| 7369 | SMITH | JON | Q | 667 | 7902 | 17-DEC-84 | 800 | 10 |

| 7499 | ALLEN | KEVIN | J | 670 | 7698 | 20-FEB-85 | 1600 | 20 |

| 7505 | DOYLE | JEAN | K | 671 | 7839 | 04-APR-85 | 2850 | 20 |

| 7506 | DENNIS | LYNN | S | 671 | 7839 | 15-MAY-85 | 2750 | 30 |

| 7507 | BAKER | LESLIE | D | 671 | 7839 | 10-JUN-85 | 2200 | 40 |

| 7521 | WARK | CYNTHIA | D | 670 | 7698 | 22-FEB-85 | 1250 | 10 |

+-------------+----------+-----------+------------+--------+------------+-----------+--------+---------------+

6 rows in set (0.00 sec)

-**CREATE TABLE LOAN-**

mysql> CREATE TABLE LOAN(EMPLOYEE_ID INT, FIRSTNAME VARCHAR(10), AMOUNT INT);

Query OK, 0 rows affected (0.23 sec)

-INSERT INTO LOAN TABLE-

mysql> INSERT INTO loan VALUES ('7506', 'LYNN', '35000'), ('7521', 'CYNTHIA', '100000'), ('6734', 'KEVIN', '65000'), ('7666', 'CHRIS', '56000'), ('7369', 'JON', '30000');

```
Query OK, 5 rows affected (0.03 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> select * from loan;

+-------------+-----------+--------+

| EMPLOYEE_ID | FIRSTNAME | AMOUNT |

+-------------+-----------+--------+

|        7506 | LYNN      |  35000 |

|        7521 | CYNTHIA   | 100000 |

|        6734 | KEVIN     |  65000 |

|        7666 | CHRIS     |  56000 |

|        7369 | JON       |  30000 |

+-------------+-----------+--------+

5 rows in set (0.00 sec)
```

CONDITIONS:

1- PERFORM JOIN OPERATIONS ON EMPLOYEE ANDLOAN TABLE:

A- INNER JOIN:

```
mysql> SELECT EMPLOYEE.FIRSTNAME,employee.EMPLOYEE_ID, EMPLOYEE.SALARY,
LOAN.AMOUNT FROM employee INNER JOIN LOAN ON
employee.EMPLOYEE_ID=loan.EMPLOYEE_ID;

+-----------+-------------+--------+--------+

| FIRSTNAME | EMPLOYEE_ID | SALARY | AMOUNT |

+-----------+-------------+--------+--------+

| JON       |        7369 |    800 |  30000 |

| LYNN      |        7506 |   2750 |  35000 |

| CYNTHIA   |        7521 |   1250 | 100000 |

+-----------+-------------+--------+--------+
```

3 rows in set (0.01 sec)

B- LEFT JOIN:

mysql> SELECT EMPLOYEE.FIRSTNAME,employee.EMPLOYEE_ID, EMPLOYEE.SALARY, LOAN.AMOUNT FROM employee LEFT JOIN LOAN ON employee.EMPLOYEE_ID=loan.EMPLOYEE_ID;

```
+-----------+-------------+--------+--------+
| FIRSTNAME | EMPLOYEE_ID | SALARY | AMOUNT |
+-----------+-------------+--------+--------+
| JON       |        7369 |    800 |  30000 |
| KEVIN     |        7499 |   1600 |   NULL |
| JEAN      |        7505 |   2850 |   NULL |
| LYNN      |        7506 |   2750 |  35000 |
| LESLIE    |        7507 |   2200 |   NULL |
| CYNTHIA   |        7521 |   1250 | 100000 |
+-----------+-------------+--------+--------+
```

6 rows in set (0.00 sec)

C- RIGHT JOIN:

mysql> SELECT loan.FIRSTNAME,employee.EMPLOYEE_ID, employee.SALARY, loan.AMOUNT FROM employee RIGHT JOIN LOAN ON employee.EMPLOYEE_ID=loan.EMPLOYEE_ID;

```
+-----------+-------------+--------+--------+
| FIRSTNAME | EMPLOYEE_ID | SALARY | AMOUNT |
+-----------+-------------+--------+--------+
| LYNN      |        7506 |   2750 |  35000 |
| CYNTHIA   |        7521 |   1250 | 100000 |
| KEVIN     |        NULL |   NULL |  65000 |
| CHRIS     |        NULL |   NULL |  56000 |
| JON       |        7369 |    800 |  30000 |
+-----------+-------------+--------+--------+
```

5 rows in set (0.00 sec)

2- PERFORM SET OPERATIONS ON LOAN AND EMPLOYEE DETAILS TABLE:

A- UNION

mysql> SELECT FIRSTNAME,EMPLOYEE_ID FROM employee UNION SELECT FIRSTNAME,EMPLOYEE_ID FROM loan ORDER BY FIRSTNAME;

```
+-----------+-------------+
| FIRSTNAME | EMPLOYEE_ID |
+-----------+-------------+
| CHRIS     |        7666 |
| CYNTHIA   |        7521 |
| JEAN      |        7505 |
| JON       |        7369 |
| KEVIN     |        7499 |
| KEVIN     |        6734 |
| LESLIE    |        7507 |
| LYNN      |        7506 |
+-----------+-------------+
8 rows in set (0.00 sec)
```

3- FIND OUT NO. OF EMPLOYEE WORKING IN SALES DEPARTMENT:

```
mysql> SELECT count(employee.EMPLOYEE_ID) AS TOTAL_SALES_PERSON FROM
employee INNER JOIN department ON
employee.DEPARTMENT_ID=department.DEPARTMENT_ID WHERE
department.NAME="SALE";
+--------------------+
| TOTAL_SALES_PERSON |
+--------------------+
|                  1 |
+--------------------+
1 row in set (0.01 sec)
```

4- FIND OUT THE EMPLOYEES NOT WORKING IN DEPARTMENT 10 AND 30:

```
mysql> SELECT employee.EMPLOYEE_ID, employee.FIRSTNAME, employee.LASTNAME,
department.DEPARTMENT_ID, department.NAME FROM employee INNER JOIN
department ON employee.DEPARTMENT_ID=department.DEPARTMENT_ID WHERE
department.DEPARTMENT_ID!=10 AND department.DEPARTMENT_ID!=30;
+-------------+-----------+----------+---------------+-----------+
| EMPLOYEE_ID | FIRSTNAME | LASTNAME | DEPARTMENT_ID | NAME      |
+-------------+-----------+----------+---------------+-----------+
|        7499 | KEVIN     | ALLEN    |            20 | RESEARCH  |
|        7505 | JEAN      | DOYLE    |            20 | RESEARCH  |
|        7507 | LESLIE    | BAKER    |            40 | OPERATION |
+-------------+-----------+----------+---------------+-----------+
3 rows in set (0.00 sec)
```

5- LIST OUT EMPLOYEE ID, LAST NAME IN DESCENDING ORDER BASED ON SALARY COLUMN:

```
mysql> SELECT EMPLOYEE_ID,LASTNAME,SALARY from employee order by SALARY
desc;
+-------------+----------+--------+
| EMPLOYEE_ID | LASTNAME | SALARY |
+-------------+----------+--------+
|        7505 | DOYLE    |   2850 |
|        7506 | DENNIS   |   2750 |
|        7507 | BAKER    |   2200 |
|        7499 | ALLEN    |   1600 |
|        7521 | WARK     |   1250 |
|        7369 | SMITH    |    800 |
+-------------+----------+--------+
6 rows in set (0.00 sec)
```

6- HOW MANY EMPLOYEES WHO ARE WORKING IN DIFFERENT DEAPRTMENTS WISE IN THE ORGANIZATION:

```
mysql> SELECT DEPARTMENT_ID,COUNT(*)  from employee group by
DEPARTMENT_ID;
+---------------+----------+
| DEPARTMENT_ID | COUNT(*) |
+---------------+----------+
|            10 |        2 |
|            20 |        2 |
|            30 |        1 |
|            40 |        1 |
+---------------+----------+
4 rows in set (0.00 sec)
```

7- LIST OUT DEPARTMENT ID HAVING ATLEAST 2 EMPLOYEES:

```
mysql> SELECT DEPARTMENT_ID,COUNT(*) from employee group by DEPARTMENT_ID
HAVING COUNT(*)>=2;
+---------------+----------+
| DEPARTMENT_ID | COUNT(*) |
+---------------+----------+
|            10 |        2 |
|            20 |        2 |
+---------------+----------+
2 rows in set (0.00 sec)
```

8- DISPLAY THE EMPLOYEE WHO GOT MAXIMUM SALARY:

```
mysql> SELECT * from employee where SALARY=(SELECT MAX(SALARY) from
employee);
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
| EMPLOYEE_ID | LASTNAME | FIRSTNAME | MIDDLENAME | JOB_ID | MANAGER_ID |
HIREDATE  | SALARY | DEPARTMENT_ID |
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
|        7505 | DOYLE    | JEAN      | K          |    671 |       7839 |
04-APR-85 |   2850 |            20 |
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
1 row in set (0.00 sec)
```

9- UPDATE THE EMPLOYEE'S SALARY WHO ARE WORKING AS CLERK ON BASIS OF 10%:

```
mysql> UPDATE employee SET SALARY=SALARY+SALARY*10/100 WHERE
JOB_ID=(SELECT JOB_ID from job where FUNC='CLERK');
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM EMPLOYEE;
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
| EMPLOYEE_ID | LASTNAME | FIRSTNAME | MIDDLENAME | JOB_ID | MANAGER_ID |
HIREDATE  | SALARY | DEPARTMENT_ID |
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
|        7369 | SMITH    | JON       | Q          |    667 |       7902 |
17-DEC-84 |    880 |            10 |
|        7499 | ALLEN    | KEVIN     | J          |    670 |       7698 |
20-FEB-85 |   1600 |            20 |
|        7505 | DOYLE    | JEAN      | K          |    671 |       7839 |
04-APR-85 |   2850 |            20 |
|        7506 | DENNIS   | LYNN      | S          |    671 |       7839 |
15-MAY-85 |   2750 |            30 |
|        7507 | BAKER    | LESLIE    | D          |    671 |       7839 |
10-JUN-85 |   2200 |            40 |
|        7521 | WARK     | CYNTHIA   | D          |    670 |       7698 |
22-FEB-85 |    800 |            10 |
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
6 rows in set (0.00 sec)
```

10-DELETE EMPLOYEE WHO ARE WORKING IN ACCOUNTING DEPARTMENT:

```
mysql> DELETE FROM employee where DEPARTMENT_ID=(SELECT DEPARTMENT_ID from
department where NAME='ACCOUNTING');
Query OK, 2 rows affected (0.10 sec)
mysql> select * from employee;
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
| EMPLOYEE_ID | LASTNAME | FIRSTNAME | MIDDLENAME | JOB_ID | MANAGER_ID |
HIREDATE  | SALARY | DEPARTMENT_ID |
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
|        7499 | ALLEN    | KEVIN     | J          |    670 |       7698 |
20-FEB-85 |   1600 |            20 |
|        7505 | DOYLE    | JEAN      | K          |    671 |       7839 |
04-APR-85 |   2850 |            20 |
|        7506 | DENNIS   | LYNN      | S          |    671 |       7839 |
15-MAY-85 |   2750 |            30 |
|        7507 | BAKER    | LESLIE    | D          |    671 |       7839 |
10-JUN-85 |   2200 |            40 |
+-------------+----------+-----------+------------+--------+------------+-
----------+--------+---------------+
4 rows in set (0.18 sec)
```

11- FIND OUT DEPARTMENT WHO HAS NO EMPLOYEE:

```
mysql> SELECT * from department WHERE NOT EXISTS(SELECT EMPLOYEE_ID from
employee where department.DEPARTMENT_ID=employee.DEPARTMENT_ID);
+---------------+------------+-------------+
| DEPARTMENT_ID | NAME       | LOCATION_ID |
+---------------+------------+-------------+
|            10 | ACCOUNTING |         122 |
+---------------+------------+-------------+
1 row in set (0.00 sec)
```

12- LIST OUT DEPARTMENT WISE MAXIMUM, MINIMUM AND AVERAGE SALARY OF THE EMPLOYEES:

```
mysql> select DEPARTMENT_ID,COUNT(*),MAX(SALARY),MIN(SALARY),AVG(SALARY)
from employee group by DEPARTMENT_ID;
+---------------+----------+-------------+-------------+-------------+
| DEPARTMENT_ID | COUNT(*) | MAX(SALARY) | MIN(SALARY) | AVG(SALARY) |
+---------------+----------+-------------+-------------+-------------+
|            20 |        2 |        2850 |        1600 |   2225.0000 |
|            30 |        1 |        2750 |        2750 |   2750.0000 |
|            40 |        1 |        2200 |        2200 |   2200.0000 |
+---------------+----------+-------------+-------------+-------------+
3 rows in set (0.01 sec)
```

13-DISPLAY EMPLOYEES WHO JOINED IN 1985:

```
mysql> select hiredate,count(*) from employee where hiredate>=('01-jan-
1985') and hiredate<=('31-dec-1985');
+-----------+----------+
| hiredate  | count(*) |
+-----------+----------+
| 20-FEB-85 |        4 |
+-----------+----------+
1 row in set (0.00 sec)
```

14- DISPLAY THE EMPLOYEES WHO ARE WORKING IN NEW YORK:

```
mysql> SELECT * FROM employee WHERE DEPARTMENT_ID=(SELECT DEPARTMENT_ID
FROM department where LOCATION_ID= (SELECT LOCATION_ID from location where
REGINAL_GROUP='NEWYORK'));
Empty set (0.11 sec)
```

15-LIST OUT EMPLOYEES WITH THEIR DEPATMENT NAMES:

```
mysql> SELECT FIRSTNAME,LASTNAME,employee.DEPARTMENT_ID,NAME,SALARY from
employee ,department where
employee.DEPARTMENT_ID=department.DEPARTMENT_ID;
+-----------+----------+---------------+-----------+--------+
| FIRSTNAME | LASTNAME | DEPARTMENT_ID | NAME      | SALARY |
+-----------+----------+---------------+-----------+--------+
| KEVIN     | ALLEN    |            20 | RESEARCH  |   1600 |
| JEAN      | DOYLE    |            20 | RESEARCH  |   2850 |
| LYNN      | DENNIS   |            30 | SALE      |   2750 |
| LESLIE    | BAKER    |            40 | OPERATION |   2200 |
+-----------+----------+---------------+-----------+--------+
4 rows in set (0.00 sec)
```