CHAPTER 7

Module IV

Complex Queries

Syllabus

Recursive Queries, nested Queries ;

7.1 SQL Joins Concept

Q. What is JOIN ? Explain different types of JOIN along with example. **MU - May 16, 2 Marks**

- The relational operations can merge columns from two different tables to form new join table.

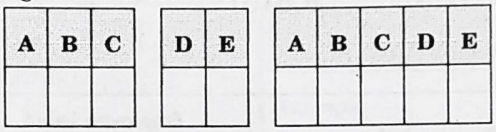


Fig. 7.1.1 : Joining operation

- We can join multiple tables with help of some join condition (Equality or Inequality) or without any join condition (cross join).
- The join concepts will be demonstrated using following HR schema which contains Employee table and Department table. These table also having a common column between them.

```
MySQL > SELECT   *
        FROM     Employees;

ID      Name      Address    Department   Salary      Age      value
------- --------  ---------  -----------  ---------   ------   ---------
1001    Ganesh    Mumbai     10           32000.00    26       M
1002    Shilpa    Pune       11           22000.00    25       F
1003    Tanushri  Goa        12           35000.00    32       F
1004    Kasturi   Pune       11           12000.00    30       F
1005    Mayuri    Mumbai     12           26000.00    31       F
1006    Harshad   Pune       10           38000.00    34       M
1007    Ganesh    Pune       NULL         18000.00    27       M
1008    Mahesh    Mumbai     19           20000.00    31       M
------- ----------  ---------  -----------  ----------  ---      ------
8 rows in set (0.00 sec) 8 rows in set (0.08 sec)
-----------------------------------------------------------------

MySQL>  SELECT   *
        FROM     Departments;
```

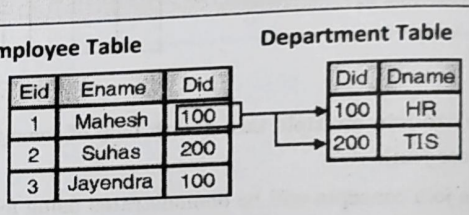
```
ID      Name        value
------  ----------  ----------
10      Technology  Mumbai
11      HR          Mumbai
12      Admin       Pune

3 rows in set (0.02 sec)
```

```
MySQL> SELECT *
       FROM Jobs;

ID      Job_Title   Min_Salary   value
------  ----------  -----------  ---------
101     Admin       10000.00     20000.00
102     Officer     20000.00     30000.00
101     Executive   30000.00     50000.00

3 rows in set (0.00 sec)
```

7.2 Cartesian product / Cross join

Q. Explain different types of JOIN along with example. **MU - May 16, 5 Marks**

- A cross join performs the relational product or Cartesian product of two tables in such product every tuple of the first table is joined with each tuple in the second table without any join condition.
- In DBMS, CROSS join occur due to missing WHERE condition in query or some invalid operations in where clause.
- A cartesian product is formed when :
  - A join condition is omitted.
  - A join condition is invalid.
  - All rows in the first table are joined to all rows in the second table.
- In such cross product operation all tuples are merged, although there is a common column or not and even tuple have matching or non-matching values in order to avoid non-matching tuples, always include a valid join condition in a WHERE clause.
- In CROSS joins, every tuple in table 1 will be joined with each tuple in table 2.



Syntax

```
SELECT  Column_List
FROM    Table1
CROSS   JOIN Table2
```

Example

```
MySQL> SELECT *
       FROM Dept;

DID     Name        value
------  ----------  --------
10      Technology  Mumbai
11      HR          Mumbai
12      Admin       Pune

3 rows in set (0.00 sec)
```

```
MySQL> SELECT *
       FROM Emp;
```

```
ID    Name     Address  DID   Salary     Age  value
----  -------  -------  ----  --------   ---  -----
1001  Ganesh   Mumbai   10    32000.00   26   M
1002  Shilpa   Pune     11    22000.00   25   F
1003  Tanushri Goa      12    35000.00   32   F
1004  Kasturi  Pune     11    12000.00   30   F
1005  Mayuri   Mumbai   12    26000.00   31   F
1006  Harshad  Pune     10    38000.00   34   M
1007  Ganesh   Pune     NULL  18000.00   27   M
1008  Mahesh   Mumbai   19    20000.00   31   M
8 rows in set (0.02 sec)
```

Find all employees with their possible department.

OR

Find all combinations of employees and departments.

```
Mysql> SELECT e.ID, e.name, e.did, d.did, d.name
    -> FROM   Emp e CROSS JOIN Dept d;

ID    Name     DID  DID  Name
----  -------  ---  ---  ----------
1001  Ganesh   10   10   Technology
1001  Ganesh   10   11   HR
1001  Ganesh   10   12   Admin
1002  Shilpa   11   10   Technology
1002  Shilpa   11   11   HR
1002  Shilpa   11   12   Admin
1008  Mahesh   19   10   Technology
1008  Mahesh   19   11   HR
1008  Mahesh   19   12   Admin

24 rows in set (0.00 sec)
```

- In above result set every row of Employee table is merged with each tuple of department table to for a row in result set after cross join.
- The 24 rows would be returned which is multiplication of rows in employee table and rows in departments table (8*3=24). Hence, this join is also called as Cross Product Join.

7.3 Inner join

Q. Explain different types of JOIN along with example.

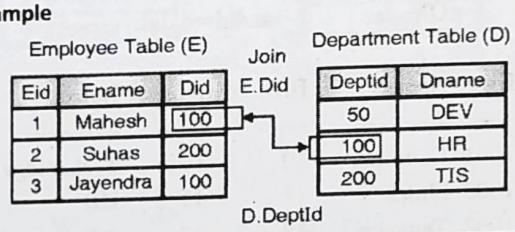
MU - May 16, 5 Marks

- An INNER JOIN performs the relational product between two tables first and than use join condition to select only tuples satisfying join condition.
- Inner join will ignore all tuples that does not satisfy join condition or having missing values in it.
- Inner join will only merge tuples if merged tuples satisfy join condition. Hence it is called as Inner Join.

Syntax

```
SELECT  Column_List
FROM    Table1
INNER JOIN Table2
ON (JOIN_Condition);
OR
SELECT  Column_List
FROM    Table1
INNER JOIN Table2
ON (JOIN_Condition);
```

Example




```
MySQL> SELECT  *
       FROM    Dept;

DID   Name        value
----  ----------  --------
10    Technology  Mumbai
11    HR          Mumbai
12    Admin       Pune

3 rows in set (0.00 sec)
```

```
MySQL> SELECT  *
       FROM    Emp;
```

ID	Name	Address	DID	Salary	Age	value
1001	Ganesh	Mumbai	10	32000.00	26	M
1002	Shilpa	Pune	11	22000.00	25	F
1003	Tanushri	Goa	12	35000.00	32	F
1004	Kasturi	Pune	11	12000.00	30	F
1005	Mayuri	Mumbai	12	26000.00	31	F
1006	Harshad	Pune	10	38000.00	34	M
1007	Ganesh	Pune	NULL	18000.00	27	M
1008	Mahesh	Mumbai	19	20000.00	31	M

8 rows in set (0.02 sec)

INNER Join

```
MySQL> SELECT e.ID, e.name, e.did, d.did, d.name
       FROM   Emp e
       INNER JOIN Dept d
       ON     e.did=d.did;
```

ID	Name	DID	DID	Name
1001	Ganesh	10	10	Technology
1002	Shilpa	11	11	HR
1003	Tanushri	12	12	Admin
1004	Kasturi	11	11	HR
1005	Mayuri	12	12	Admin
1006	Harshad	10	10	Technology

6 rows in set (0.02 sec)

- In MySQL the INNER JOIN selects all tuples from both participating tables if and only if both tables meet the conditions specified in the ON clause.

Case 1 : Natural join

Q. Explain different types of JOIN along with example. **MU - May 16, 5 Marks**

- A natural join will selects all tuples from both participating tables, if and only if both tables has one common column with same attribute value in both tables.
- In this type of join, joining tables must have at least one common column between them which has same column name and data type.

Working

- Join processor looks for a columns present in both tables.
- Perform Cross Join between participating tables (unconditional step).
- Retain row in result if it has matching attribute values in common column.

Syntax

```
SELECT  Column_List
FROM    Table1
NATURAL JOIN Table2;
```

Example

```
MySQL> SELECT  *
       FROM    Dept;
```

DID	Name	value
10	Technology	Mumbai
11	HR	Mumbai
12	Admin	Pune

3 rows in set (0.00 sec)

```
MySQL> SELECT  *
       FROM    Emp;
```

ID	Name	Address	DID	Salary	Age	value
1001	Ganesh	Mumbai	10	32000.00	26	M
1002	Shilpa	Pune	11	22000.00	25	F
1003	Tanushri	Goa	12	35000.00	32	F
1004	Kasturi	Pune	11	12000.00	30	F
1005	Mayuri	Mumbai	12	26000.00	31	F
1006	Harshad	Pune	10	38000.00	34	M
1007	Ganesh	Pune	NULL	18000.00	27	M
1008	Mahesh	Mumbai	19	20000.00	31	M

8 rows in set (0.02 sec)

Natural Join

```
MySQL > SELECT *
        FROM  Emps NATURAL JOIN Depts;
```

DID	ID	Name	Address	Salary	Age	Dname
10	1001	Ganesh	Mumbai	32000.00	26	Technology
11	1002	Shilpa	Pune	22000.00	25	HR
12	1003	Tanushri	Goa	35000.00	32	Admin
11	1004	Kasturi	Pune	12000.00	30	HR
12	1005	Mayuri	Mumbai	6000.00	31	Admin
10	1006	Harshad	Pune	38000.00	34	Technology

6 rows in set (0.00 sec)

- In MySQL the INNER JOIN selects all tuples from both participating tables if and only if both tables meet the conditions specified in the ON clause.

Case 2 : Self join

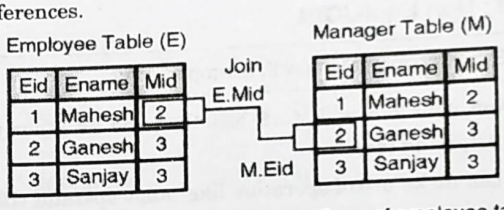
Q. Explain different types of JOIN along with example.

MU - May 16, 5 Marks

- A Self-join will work exactly same as Inner join but there is only one participating table and both columns involved in joining are in the same table.
- The tables being joined in a query do not need to be distinct; you can join any table to itself.
- Self-joins are useful for discovering relationships between different columns of the same table.
- In this type of join query we must use rename operation or table alias.

Example

This query joins the Employee table to itself over the ID (Employee ID) and Mid (Manager Id) column, using the table aliases E and M to distinguish the table references.



(Copy of employee table)

```
MySQL> SELECT   *
       FROM     Emps;
```

ID	Name	Address	DID	Salary	Age	MID
1001	Ganesh	Mumbai	10	32000.00	26	1004
1002	Shilpa	Pune	11	22000.00	25	1004
1003	Tanushri	Goa	12	35000.00	32	1004
1004	Kasturi	Pune	11	12000.00	30	1006
1005	Mayuri	Mumbai	12	26000.00	31	1006
1006	Harshad	Pune	10	38000.00	34	1006
1007	Ganesh	Pune	NULL	18000.00	27	1006
1008	Mahesh	Mumbai	19	20000.00	31	1006

8 rows in set (0.00 sec)

Self Join

```
MySQL> SELECT      E.ID, E.Name, E.MID, M.ID, M.Name
       FROM        Emps E
       INNER JOIN  Emps M
       ON          E.MID = M.ID;
```

ID	Name	MID	ID	Name
1001	Ganesh	1004	1004	Kasturi
1002	Shilpa	1004	1004	Kasturi
1003	Tanushri	1004	1004	Kasturi
1004	Kasturi	1006	1006	Harshad
1005	Mayuri	1006	1006	Harshad
1006	Harshad	1006	1006	Harshad
1007	Ganesh	1006	1006	Harshad
1008	Mahesh	1006	1006	Harshad

8 rows in set (0.00 sec)

Case 3 : Non Equi-JOIN

Q. Explain non-equijoin with example.

- A Non equi-join will work exactly same as Inner join but there is equality join condition is involved in joining participating table.
- We can make use of operators like range operator (BETWEEN) or limit operator (IN).
- The join query in which equality condition is not used then such joins are called as **Non Equi-join.**

Example :

To join department and Employee table using joining condition that 'Did' column of employee table should match with 'Did' column of department table.

```
MySQL > SELECT   *
        FROM     Empdt;

  ID        NAME       JOB_ID
  ------    ---------- --------
  1001      Ganesh     101
  1002      Shilpa     102
  1003      Tanushri   101
  1004      Kasturi    103
  1005      Mayuri     102
  1006      Harshad    101
  1007      Ganesh     103
  1008      Mahesh     102

8 rows in set (0.00 sec)
-------------------------------------------------------------
MySQL> SELECT   *
       FROM     Jobs;
```

```
ID      Job_Title    minSalary     maxSalary
------  ----------   ----------    ----------
101     Admin        10000.00      20000.00
102     Officer      20000.00      30000.00
101     Executive    30000.00      50000.00

3 rows in set (0.00 sec)
```

Non Equi JOIN

```
MySQL> SELECT     E.ID, E.Name, E.Job_Id, J.Job_Title,
                  E.Salary, J.minsalary, J.maxsalary
       FROM       Empdt E
       INNER JOIN Jobdt J
       ON         E.Salary BETWEEN J.minSalary AND J.maxSalary;
```

ID	Name	Job_Id	Job_Title	Salary	minsalary	maxsalary
1001	Ganesh	101	Executive	32000.00	30000.00	50000.00
1002	Shilpa	102	Officer	22000.00	20000.00	30000.00
1003	Tanushri	101	Executive	35000.00	30000.00	50000.00
1004	Kasturi	103	Admin	12000.00	10000.00	20000.00
1005	Mayuri	102	Officer	26000.00	20000.00	30000.00
1006	Harshad	101	Executive	38000.00	30000.00	50000.00
1007	Ganesh	103	Admin	18000.00	10000.00	20000.00
1008	Mahesh	102	Admin	20000.00	10000.00	20000.00
1008	Mahesh	102	Officer	20000.00	20000.00	30000.00

9 rows in set (0.00 sec)

7.4 Outer join

Q. Explain different types of JOIN along with example. **MU - May 16, 5 Marks**

1. Introduction

- In an inner join, the resultant table contains only the combinations of rows that satisfy the join conditions. Rows that do not satisfy the join conditions are discarded. Outer join, will merge two table although there is no match between attribute values of common column.
- An outer join makes one of the tables dominant. Such table is called the outer table and other table is called as subordinate table. In an outer join, the resultant table contains the combinations of rows from dominant table that satisfy the join conditions and also rows that do not having matching rows in the subordinate table.
- The rows from the dominant table that do not have matching rows in the subordinate table contain NULL values in the columns selected from the subordinate table.
- The Join that can be used to see rows of table that do not meet the join condition along with rows that satisfies join condition is called as **outer join.**

2. Syntax

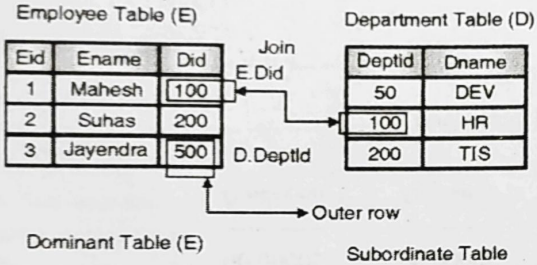
```
SELECT    Column_List
FROM      Table1
```

```
[LEFT/RIGHT/FULL] OUTER JOIN Table2
ON          Join_Condition;
```

3. Left outer join

- Left outer join has the dominant table of the outer join on left side of the keyword 'outer join'.
- A left outer join returns all of the rows for which the join condition is true and, in addition, returns all other rows from the dominant table and displays the corresponding values from the subordinate table as NULL.

Example



Eid	Ename	Did
1	Mahesh	100
2	Suhas	200
3	Jayendra	500

Deptid	Dname
50	DEV
100	HR
200	TIS

```
MySQL> SELECT E.ID, E.Name, E.DID, D.DID, D.Name
       FROM Emps E
       LEFT OUTER JOIN Dept D
       ON E.DID=D.DID;
```

ID	Name	DID	DID	Name
1001	Ganesh	10	10	Technology
1006	Harshad	10	10	Technology
1002	Shilpa	11	11	HR
1004	Kasturi	11	11	HR
1003	Tanushri	12	12	Admin
1005	Mayuri	12	12	Admin
1007	Ganesh	NULL	NULL	NULL
1008	Mahesh	19	NULL	NULL

8 rows in set (0.00 sec)

- The above table returns rows in employees table those having or not having any department. The query below will returns only rows in which employees do not have any department.
- In this query, the database server applies the filter in the WHERE clause after it performs the outer join on Did column of the Employee and Department tables.

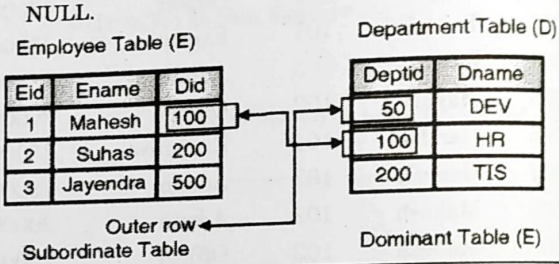
```
MySQL> SELECT E.ID, E.Name, E.DID, D.DID, D.Name
       FROM Emps E
       LEFT OUTER JOIN Dept D
       ON E.DID=D.DID
       WHERE D.DID IS NULL;
```

ID	Name	DID	DID	Name
1007	Ganesh	NULL	NULL	NULL
1008	Mahesh	19	NULL	NULL

2 rows in set (0.00 sec)

4. Right outer join

- Right outer join has the dominant table of the outer join on right side of the keyword 'outer join'.
- A right outer join returns all of the rows for which the join condition is true and also includes all other rows from the dominant table and displays the corresponding values from the subordinate table as NULL.



Eid	Ename	Did
1	Mahesh	100
2	Suhas	200
3	Jayendra	500

Deptid	Dname
50	DEV
100	HR
200	TIS

```
MySQL> SELECT E.ID, E.Name, E.DID, D.DID, D.Name
       FROM Emps E
       RIGHT OUTER JOIN Dept D
       ON E.DID=D.DID;
```

ID	Name	DID	DID	Name
1001	Ganesh	10	10	Technology
1002	Shilpa	11	11	HR
1003	Tanushri	12	12	Admin
1004	Kasturi	11	11	HR
1005	Mayuri	12	12	Admin
1006	Harshad	10	10	Technology
NULL	NULL	NULL	21	Extra

7 rows in set (0.00 sec)

- The above table returns rows in employees table those having or not having any department.
- The query below will returns only rows in department table without any Employees.
- In this query, the database server applies the filter in the WHERE clause after it performs the outer join on Eid column of the Employee and Department tables.

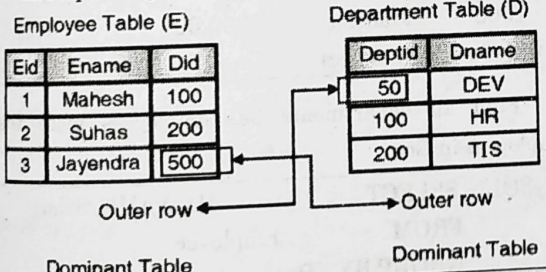
```
MySQL> SELECT   E.ID, E.Name, E.DID, D.DID,
       D.Name
       FROM     Emps E
       LEFT OUTER JOIN Dept D
       ON       E.DID=D.DID
       WHERE    E.DID IS NULL;

+------+---------+------+------+-------+
| ID   | Name    | DID  | DID  | Name  |
+------+---------+------+------+-------+
| NULL | NULL    | NULL | 21   | Extra |
+------+---------+------+------+-------+
1 rows in set (0.00 sec)
```

5. Full outer join

- In full outer join both the table acts as dominant tables alternatively.
- A full outer join returns all of the rows for which the join condition is true and also includes,

a. All other rows from the right table and displays the corresponding values from the left table as NULL.

b. All other rows from the left table and displays the corresponding values from the right table as NULL.




```
MySQL> SELECT   E.ID, E.Name, E.DID, D.DID,
       D.Name
       FROM     Emps E
       FULL OUTER JOIN Dept D
       ON       E.DID=D.DID;

+------+---------+------+------+-----------+
| ID   | Name    | DID  | DID  | Name      |
+------+---------+------+------+-----------+
| 1001 | Ganesh  | 10   | 10   | Technology|
| 1002 | Shilpa  | 11   | 11   | HR        |
| 1003 | Tanushri| 12   | 12   | Admin     |
| 1004 | Kasturi | 11   | 11   | HR        |
| 1005 | Mayuri  | 12   | 12   | Admin     |
| 1006 | Harshad | 10   | 10   | Technology|
| 1007 | Ganesh  | NULL | NULL | NULL      |
| 1008 | Mahesh  | 19   | NULL | NULL      |
| NULL | NULL    | NULL | 21   | Extra     |
+------+---------+------+------+-----------+
9 rows in set (0.00 sec)
```

- The above table returns rows in employees table those having or not having any department and vice versa.

```
MySQL> SELECT   E.ID, E.Name, E.DID, D.DID,
       D.Name
       FROM     Emps E
       LEFT OUTER JOIN Dept D
       ON       E.DID=D.DID
       WHERE    E.DID IS NULL OR D.DID IS
                NULL;

+------+---------+------+------+-----------+
| ID   | Name    | DID  | DID  | Name      |
+------+---------+------+------+-----------+
| 1007 | Ganesh  | NULL | NULL | NULL      |
| 1008 | Mahesh  | 19   | NULL | NULL      |
| NULL | NULL    | NULL | 21   | Extra     |
+------+---------+------+------+-----------+
3 rows in set (0.00 sec)
```

7.5 Nested Sub Queries

Q. Explain Nested queries. **MU - Dec.17, May 19, 5 Marks**

Q. Explain concept of sub query.

Q. Describe various operators for multiple sub query.

Q. What is nested sub query ? Explain ANY ALL operators with example.

- A Subquery or Nested sub query is a query within another SQL query.
- Subquery is query appear within **WHERE** or **HAVING** clause of other query.

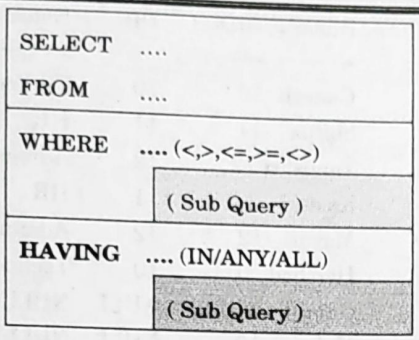


Fig. 7.5.1 : Subquery

- Outer query is called as **main query** and inner query which is written in (where or having clause) main query is called **subquery**.
  - **Subquery in WHERE clause :** The result of the subquery (inner query) is used to select some rows from main query.
  - **Subquery in HAVING clause :** The result of the subquery (inner query) is used to select some groups from main query.
- Sub queries can be nested within other sub queries.

Syntax

```
SELECT      Select_list
FROM        Table
 WHERE      expr_operator ( SELECT Select_list
                            FROM   Table);
OR
SELECT      Select_list
 FROM       Table
GROUP BY Column
 HAVING  expr_operator (SELECT  Select_list
                        FROM    Table);
```

Expr_operator can be of two types like,

a. Single row operator (<, >, <=, >=, <>)

b. Multiple row operator (IN, ANY, ALL)

7.6 Independent Subquery

- A subquery is one which returns only one row to main query.
- A subquery which uses single row operators in above syntax is called as single row subquery.

Example

Consider an Employee table (Empdt) has a columns like name and salary.

```
mysql> SELECT   *
       FROM     Empdt;
  ID      NAME       SALARY
------   ---------  ----------
1001     Ganesh     32000.00
1002     Shilpa     22000.00
1003     Tanushri   35000.00
1004     Kasturi    12000.00
1005     Mayuri     26000.00
1006     Harshad    38000.00
1007     Ganesh     18000.00
1008     Mahesh     20000.00
8 rows in set (0.00 sec)
```

Find all employees whose salary less than Shilpa's salary.

```
MySQL> SELECT   ID, NAME, Salary
       FROM     Empdt
       WHERE    Salary < ( SELECT  Salary
                           FROM    Empdt
                           WHERE   NAME LIKE
                                   'Shilpa');
  ID      NAME       Salary
------   ---------  ----------
1004     Kasturi    12000.00
1007     Ganesh     18000.00
1008     Mahesh     20000.00
3 rows in set (0.03 sec)
```

Find all departments has employees more than employees in Goa.

```
MySQL> SELECT          ID, NAME, Salary
       FROM            Employee
       GROUP BY        Department
     HAVING Count(ID) > ( SELECT  count(*)
                          FROM    Employee
                          WHERE Address = 'GOA');
  ID      NAME       Salary
------   ---------  ----------+
1001     Ganesh     32000.00
```

```
1002    Shilpa      22000.00
1003    Tanushri    35000.00

3 rows in set (0.05 sec)
```

As only one row (Value) is returned by inner query (Subquery) so these types of query is called as Single row subquery.

7.7 Multiple Row Subquery

- Multiple row subquery returns one or more than rows to the outer SQL statement.
- We can use the IN, ANY, or ALL operator in outer query to handle a subquery returns multiple rows.
- If subquery returns more than one row then that type of query known as multiple row sub query.
- The quantified tests (i.e. **ANY** and **ALL**) uses any one of the simple comparison operators to compare a test value to all of the values returned by a subquery, checking to see whether the comparison holds for some or all of the values.

Sr. No.	Operator	Meaning	Name
1.	IN	Equal to any member in the list	Set Membership Test
2.	ANY	Compare value to each value returned by the subquery	Quantified Tests
3.	ALL	Compare value to every value returned by the subquery	
4.	EXISTS	Checks weather subquery returns a value or not.	Existence Tests

Consider employee details table (Empdtls) shows details of employee with department details table (Deptdtls) represents department employees working for and Job details table (Jobdtls) shows salary details of employee.

```
MySQL> SELECT   *
       FROM     Empdtls;

ID      Name        Address     Department   Jod_id      salary
------  ----------  ---------   ----------   ----------  ----------
1001    Ganesh      Mumbai      10           Admin       32000.00
1002    Shilpa      Pune        11           Officer     22000.00
1003    Tanushri    Goa         12           Admin       35000.00
1004    Kasturi     Pune        11           Executive   12000.00
1005    Mayuri      Mumbai      12           Officer     26000.00
1006    Harshad     Pune        10           Admin       38000.00
1007    Ganesh      Pune        NULL         Executive   18000.00
1008    Mahesh      Mumbai      19           Officer     20000.00

8 rows in set (0.00 sec)

MySQL> SELECT   *
       FROM     Deptdtls;

IDD     Name        Address
------  ----------  ---------
1001    Ganesh      Mumbai
1002    Shilpa      Pune
```

```
1003    Tanushri    Goa
1004    Kasturi     Pune
1005    Mayuri      Mumbai
1006    Harshad     Pune
1007    Ganesh      Pune
1008    Mahesh      Mumbai

8 rows in set (0.03 sec)
```

```
MySQL> SELECT    *
       FROM      Deptdtls;

ID      DName       Address
------  ----------  ---------
1001    Ganesh      Mumbai
1002    Shilpa      Pune
1003    Tanushri    Goa
1004    Kasturi     Pune
1005    Mayuri      Mumbai
1006    Harshad     Pune
1007    Ganesh      Pune
1008    Mahesh      Mumbai

8 rows in set (0.03 sec)
```

```
MySQL> SELECT    *
       FROM      Jobdtls;

ID      Job_Title     minSalary     maxSalary
------  ----------    ----------    ----------
101     Admin         10000.00      20000.00
102     Officer       20000.00      30000.00
101     Executive     30000.00      50000.00
------  ----------    ----------    ----------
3 rows in set (0.04 sec)
```

1. IN

- The subquery forms the set membership test (IN) matches a test value to the set of values returned by a subquery.
- This multiple row operator used to check that a given tuple in main query satisfy at least one values (out of set of values) returned by a subquery.
- IN operator used to check equlity of data.
- Inversely we can use NOT IN condition to check test value is not a member of result set of inner subquery.

Example

Find out all employees working for HR and Admin department.

```
MySQL> SELECT    *
       FROM      Empdtls
```

```
    WHERE Department IN (SELECT    ID
                         FROM Deptdtls
                    WHERE Dname IN ('HR','Admin'));
ID    Name    Address   Dept    Jod_id      salary
----  ------  --------- --------- ---------- -------
1002  Shilpa  Pune      11      Officer    22000.00
1003  Tanushri Goa      12      Admin      5000.00
1004  Kasturi  Pune     11      Executive  12000.00
1005  Mayuri   Mumbai   12      Officer    26000.00
4 rows in set (0.00 sec)
```

Find out all employees not working for departments in Mumbai and Pune.

```
MySQL> SELECT    *
       FROM      Empdtls
      WHERE  Department NOT IN (SELECT ID
                                FROM      Deptdtls
                WHERE     Address IN ('Mumbai','Pune'));
ID    Name   Address  Department  Jod_id      salary
----  ------ -------- ----------- ---------- -------
1008  Mahesh Mumbai   19          Officer    20000.00
1 row in set (0.00 sec)
```

Find out all employees with salary lower than 20000 and working for departments in Mumbai.

```
MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Salary < 40000
     AND Department NOT IN  (SELECT      ID
                             FROM   Deptdtls
                         WHERE      Address = 'Mumbai');
  ID    Name    Address    Dept  Jod_id      salary
  ------ --------- ---------  ------ ---------- -------
  1003  Tanushri  Goa        12    Admin      35000.00
  1005  Mayuri    Mumbai     12    Officer    26000.00
  1008  Mahesh    Mumbai     19    Officer    20000.00
3 rows in set (0.00 sec)
```

2. ANY

- This multiple row operator used to check that a given tuple in main query satisfies given condition for at least one tuple value (out of set of values) returned by a subquery.
- Checking to see whether the comparison holds true for some values of subquery.
- It is used along with Less than (<) or greater than (>) operators for testing set comparison.

Example

Find out all employees having salary less than any of the Officer.

```
MySQL> SELECT    *
       FROM      Empdtls
         WHERE  Salary > ( SELECT    Salary
                  FROM      Empdtls
                  WHERE       Job_id = 'Officer');
ERROR 1242 (21000): Subquery returns more than 1 row
-----------------------------------------------------------

MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Salary > ANY (SELECT  Salary
                               FROM   Empdtls
                           WHERE Job_id = 'Officer');
ID    Name     Address   Dept.     salary     Job_Id
-----  ---------- --------- ---------  --------  -------
1001 Ganesh   Mumbai  10       32000.00  Admin
1002 Shilpa   Pune    11       22000.00  Officer
1003 Tanushri Goa     12       35000.00  Admin
1005  Mayuri  Mumbai  12       26000.00  Officer
1006 Harshad  Pune    10     38000.00  Admin

5 rows in set (0.00 sec)
```

Find out all employees who are not officer but have salary more than any of the Officer.

```
MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Job_Id <> 'Officer'
       AND       Salary > ANY (  SELECT Salary
                                 FROM  Empdtls
                             WHERE Job_id = 'Officer');

ID     Name     Address   Dept.   salary      Job_Id
----   -------  -------  --------  --------  --------
1004  Kasturi   Pune      11    12000.00  Executive
1007  Ganesh    Pune     NULL   18000.00  Executive
2 rows in set (0.00 sec)
```

Find out all Executives having salary less than any one officer in employees.

```
MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Job_Id = 'Executive'
   AND       Salary < ANY ( SELECT  Salary
                            FROM    Empdtls
                          WHERE Job_id = 'Officer');
ID   Name   Address Department   salary   Job_Id
-----  ---------  --------  ---------- ----  -------
1004 Kasturi  Pune   11        12000.00 Executive
1007 Ganesh  Pune   NULL      18000.00 Executive
2 rows in set (0.00 sec)
```

3. ALL

- This multiple row operator used to check that a given tuple in main query satisfies given condition for all values (out of set of values) returned by a subquery.
- This quantified tests use one of the simple comparison operators to compare a test value to all of the values returned by a subquery, checking to see whether the comparison holds for all values.

Example

Find out all employees having salary less than all the Officers.

```
MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Salary > ( SELECT      Salary
                            FROM        Empdtls
                  WHERE     Job_id = 'Officer');
ERROR 1242 (21000): Subquery returns more than 1 row
-----------------------------------------------------------------------
MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Salary > ALL (SELECT       Salary
                               FROM         Empdtls
                               WHERE        Job_id = 'Officer');

  ID        Name       Address      Department     salary          Job_Id
------  ---------  ---------  ------------  ----------  --------
  1001      Ganesh     Mumbai       10          32000.00    Admin
  1003      Tanushri   Goa          12          35000.00    Admin
  1006      Harshad    Pune         10          38000.00    Admin

3 rows in set (0.00 sec)
```

Find out all employees having salary more than all employees in HR department.

```
MySQL> SELECT    *
    FROM      Empdtls
       WHERE     Salary > ALL
                 (SELECT     Salary
                 FROM  Empdtls
                 WHERE Department IN (SELECT    ID
                                      FROM      Deptdtls
                                     WHERE      Dname= 'HR'));
```

ID	Name	Address	Department	salary	Job_Id
1001	Ganesh	Mumbai	10	32000.00	Admin
1003	Tanushri	Goa	12	35000.00	Admin
1005	Mayuri	Mumbai	12	26000.00	Officer
1006	Harshad	Pune	10	38000.00	Admin

4 rows in set (0.00 sec)

4. EXISTS clause

- The existence test (EXISTS) checks whether a subquery returns any values or returns nothing.
- EXISTS clause is used for testing whether a subquery has any tuples in there result set or is it empty.
- The EXISTS clause results True value if the subquery is nonempty else it returns in False value.
- NOT EXISTS is complementary test to above existence test checks whether a subquery does not return any values which satisfy search criteria.
- The NOT EXISTS clause results in True if the subquery is empty.

Example

Print all Employees only if there are few employees have salary more than 35000.

```
MySQL> SELECT   *
       FROM     Empdtls
       WHERE  EXISTS ( SELECT     ID
                       FROM       Empdtls
                       WHERE      Salary > 35000);
```

ID	Name	Address	Department	salary	Job_Id
1001	Ganesh	Mumbai	10	32000.00	Admin
1002	Shilpa	Pune	11	22000.00	Officer
1003	Tanushri	Goa	12	35000.00	Admin
1004	Kasturi	Pune	11	12000.00	Executive
1005	Mayuri	Mumbai	12	26000.00	Officer
1006	Harshad	Pune	10	38000.00	Admin
1007	Ganesh	Pune	NULL	18000.00	Executive
1008	Mahesh	Mumbai	19	20000.00	Officer

8 rows in set (0.00 sec)

Find out all employees having salary more than all employees in HR department.

```
MySQL> SELECT   *
       FROM     Empdtls
       WHERE    Salary > ALL   (SELECT     ID
                               FROM       Empdtls
                               WHERE      Department IN (SELECT ID
                               FROM       Deptdtls
```

```
                    WHERE    Dname = 'HR'));
ID      Name      Address    Department    salary      Job_Id
------  --------  ---------  ------------  ----------  ---------
1001    Ganesh    Mumbai     10            32000.00    Admin
1003    Tanushri  Goa        12            35000.00    Admin
1005    Mayuri    Mumbai     12            26000.00    Officer
1006    Harshad   Pune       10            38000.00    Admin

4 rows in set (0.00 sec)
```

Print all Department if there are few employees with no department.

```
MySQL> SELECT    *
       FROM      Deptdtls
       WHERE EXISTS (  SELECT  ID
                       FROM    Empdtls
                       WHERE   Department IS NULL);
ID      DName          Address
------  -------------  ---------
10      Technology     Mumbai
11      HR             Mumbai
12      Admin          Pune
21      Extra          Pune
4 rows in set (0.00 sec)
```

Print all officer employees if there are no employee work as manager.

```
MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Job_Id = 'Officer'
     AND NOT EXISTS ( SELECT ID
                      FROM   Empdtls
                      WHERE Job_Id = 'Manager');
ID    Name   Address Department salary   Job_Id
----- -------- --------- ------------ --------- ---------
1002  Shilpa   Pune      11           22000.00  Officer
1005  Mayuri   Mumbai    12           26000.00  Officer
1008  Mahesh   Mumbai    19           20000.00  Officer
3 rows in set (0.00 sec)
```

Print all officer employees if there are no employee work as Executives.

```
MySQL> SELECT    *
       FROM      Empdtls
       WHERE     Job_Id = 'Officer'
     AND NOT EXISTS ( SELECT ID
                      FROM   Empdtls
                      WHERE Job_Id = 'Executive');
Empty set (0.00 sec)
```

7.8 Correlated Sub Queries

Q. Explain recursive queries. **MU - Dec.17, 5 Marks**

1. Introduction

- In a case of nested subquery, the first subquery will be solved then using this result, outer query will be evaluated.
- A simple subquery is evaluated only once for each query.
- A correlated subquery is evaluated once for each result row of main query.
- In a case of correlated subquery, at first main query evaluated to get first result row and than solve entire subquery for that first row of main query.
- Correlated sub queries processes data in row-by-row pattern. Each subquery is executed once for every row of the outer query.
- We can use a correlated subquery in the HAVING clause also.

2. Syntax

```
SELECT    *
FROM      table1 [ref]
WHERE     column1 operator
              (SELECT  colum1, column2
               FROM    table2
               WHERE   expr1 = ref.expr2);
```

The subquery references a column from a table in the main query.

3. Example

The inner query is executed separately for each row of the outer query i.e. in correlated sub-queries, SQL performs a sub-query over and over again – once for each row of the main query.

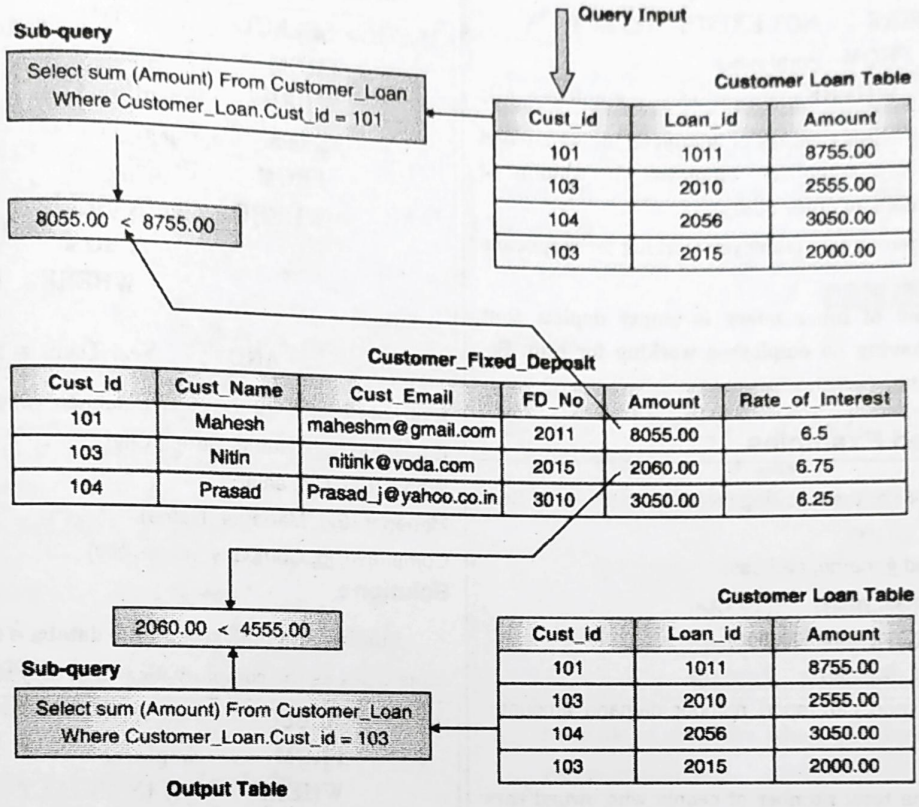


Fig. 7.8.1

Example

Find customers with fixed deposit less than the sum of all their loans.

```
MySQL > SELECT    Cust_id, Cust_Name
    FROM      Customer_Fixed_Deposit CFD
    WHERE Amount < ( SELECT     SUM(Amount)
                     FROM       ustomer_Loan
                WHERE      Cust_id = CFD.Cust_id);
```

Find all employees who earn less than the average salary in own department.

```
MySQL> SELECT     Faculty_Id, department
     FROM      Employees outer
     WHERE     Salary < ( SELECT  AVG(salary)
                          FROM       employees
                WHERE  department = outer.department);
```

Find out the employees who have at least one person reporting to them.

```
MySQL> SELECT     employee_id, last_name
     FROM      employees e
     WHERE
EXISTS (SELECT    'X'
        FROM      employee
        WHERE     manager_id = e.employee_id);
```

TRUE - Print row
FALSE - Do not print that row

- If result set of inner query is nonempty depicts that manager having some employees working for him. So, print result on screen.
- To check there is any employee working for respective manager.
- The EXISTS operator ensures that the search should not continue further to optimize performance of query.

To find out the employees who have no one which is reporting to them.

```
MySQL> SELECT    Employee_id, last_name
       FROM      Employees e
       WHERE     NOT EXISTS ( SELECT  'X'
                 FROM  employees
                 WHERE manager_id = e.employee_id);
```

- Inner query checks values of 'manager_id' is matched with the any value in 'employee_id' column of employee table (of outer query).
- To check there is any employee working for respective manager.
- If result set of inner query is empty depicts that manager having no employees working for him. So, print result on screen.

7.9 Solved Examples

Example 7.9.1 : For the following given database, write SQL queries :

person (driver_id #, name, address)

car (license, model, year)

accident (reportCno, date, location)

owns (driver id #, license)

participated (driverid, car, report_number, damage_amount)

Solution :

1. Finding the total number of people who owned cars that were involved in an accident in 2007.

```
MySQL>  SELECT   COUNT(*)
        FROM     Person
        WHERE    Driver_id IN
  (SELECT  Driverid
        FROM     Participated
        WHERE    Car IN (SELECT ReportCno
        FROM     Accident
        WHERE    Year(Date) = 2007);
```

2. Finding the number of accidents in which the cars belonging to "Ajay" were involved.

```
MySQL> SELECT    count(*)
       FROM      accident
  WHERE  reportCno IN (SELECT    report_number
       FROM      participated
       WHERE     car IN (SELECT driver_id
       FROM      person
       WHERE     name = 'Ajay');
```

3. Finding the number of accidents that were reported in Mumbai region in the year 2004.

```
MySQL> SELECT    COUNT(*)
       FROM      accident
       WHERE     reportCno IN
       (Select   report_number
        FROM     Participated
        WHERE    Car IN (SELECT ReportCno
                 FROM     Accident
                 WHERE    location
                          = 'Mumbai'
              AND       Year(Date) = 2004);
```

Example 7.9.2 : For the given database, write SQL queries.

Employee (Eid. Name, Street, City)

Works (Eid. Cid. salary)

Manager (Eid, Manager_Name)

Company(Cid, Company_name, city)

Solution :

Finding all employees in the database who live in the same cities as the company for which they work.

```
MySQL> SELECT    *
       FROM      Employee
       WHERE     City IN (SELECT City
                 FROM     Company
                 WHERE Eid = Employee·Eid)
```

Example 7.9.3 : Consider the following relations for database that keeps track of student enrolment in courses and books issued for each course.

STUDENT (Ssn, Name, Subject, DOB)

COURCE (Course_id, Name, Dept)

ENROLL (Ssn, Course_id, Semester, Grade)

BOOK_ISSUED (Course_id, Semester, ISBN)

TEXT (ISBN, Title, Publisher, Author)

Solution :

1. Finding all student details registered for course id 10.

```
MySQL> SELECT    *
       FROM      STUDENT S
       WHERE     Ssn = (SELECT Ssn
                 FROM ENROLL E
                 WHERE E·Course_id = 10);
```

2. Finding various book titles and authors for semester higher than 3.

```
MySQL> SELECT      *
       FROM        BOOK_ISSUED B
       NATURAL JOIN TEXT T
       WHERE       B.Semester > 3
```

3. Finding all students belongs to IT Department (without join)

```
MySQL> SELECT Ssn, CName
       FROM  STUDENT
       WHERE Ssn IN (SELECT Ssn
                     FROM ENROLL
                     WHERE Course_id IN
                     (SELECT Course_id
                     FROM    Course
                     WHERE   Dept = 'IT'))
```

4. Finding total number of student s enrolled in IT Department.

```
MySQL>  SELECT  COUNT (E.Ssn)
                Total_Students
        FROM    STUDENT
        NATURAL JOIN ENROLL
        NATURAL JOIN Course;
```

Example 7.9.4 : Given the following relational schema.

Division (div #, div-name, director)

Department (dept #, dept-name, location, div #)

Employee (emp#, emp-name, salary, address, dept #)

State the following queries in SQL :

1. Get the employee name, dept-name and division name for all employees whose salary is above 20,000/-
2. List the dept-name and employee names in that dept, for all department whose location is "Mumbai".

Solution :

1. Getting the employee name, dept-name and division name for all employees whose salary is above 20,000/-

```
MySQL> SELECT      *
       FROM        EMPLOYEE e
       INNER JOIN  Department d ON e.dept =
                   d.dept
  INNER JOIN Division dv ON d.div = dv.div
       WHERE       e.salary > 20000;
```

2. List the dept-name and employee names in that dept, for all department whose location is "Mumbai".

```
MySQL>  SELECT      *
        FROM        EMPLOYEE e
        INNER JOIN  Department d ON e.dept =
d.dept
        WHERE       d.location = 'Mumbai';
```

Example 7.9.5 : Consider the following relations for a book club :

Members (Member-Id, Name, Designation, Age)

Books (Book-Id, Booktitle, BookAuthor, Bookpublisher, Bookprice)

Reserves (Member-Id, Book-Id, Date)

MU - May 14, 10 Marks

Solution :

1. List the titles of books reserved by professors.

```
MySQL>  SELECT  Booktitle
        FROM    Books
        WHERE   Book-id IN (SELECT Book-id
                            FROM   Reserves
        WHERE   Member-id IN
                (SELECT Member-id
                FROM    Members
                WHERE   Designation='Professor'));
```

2. Find Ids of members who have reserved books that cost more than 500.

```
MySQL> SELECT  Member-id
       FROM    Members
       WHERE   Member-id IN
               (SELECT  Member-id
               FROM     Reserves
             WHERE      Book-id IN
             (SELECT    Book-id
             FROM       Books
             WHERE      Book price > 500));
```

3. Find the authors and titles of books reserved on 20-09-2012.

```
MySQL> SELECT  Booktitle, BookAuthor
       FROM    Books
    WHERE      Book-id IN (SELECT Book-id
               FROM   Reserves
               WHERE  Date = '2012-09-02');
```

Example 7.9.6 : Consider the following employee database.
Employee (empname, street, city, date_of_joining)
Works (empname, company _name, salary)
Company (company_name, city)
Manages (empname, manager_name).

MU - Dec. 13, 10 Marks

Solution :

1. List all employees who live in the same cities as their managers.

```
MySQL> SELECT   *
       FROM     Employee E, Manages M
       WHERE    E.empname = M.empname
       AND      E.city = M.city);
```

2. Find all employees who earn more than average salary of all employees of their company.

```
MySQL> SELECT   *
       FROM     Employee E, Works W
       WHERE    E.empname= W.empname
       AND      W.salary > (SELECT AVG(salary)
                            FROM Employee
                WHERE company_name = W.company_name);
```

Example 7.9.7 : Refer education atabase mentioned in Q. 2 (a) Write SQL queries for the following :

1. List all the teacher who conduct the course titled Database Systems.
2. List all the course offered in Thane on 15/8/15.
3. Find the course enrolled by Monali.
4. List all the employees who work as Teachers.

MU - Dec. 15, 10 Marks

Solution

1. List all the teacher who conduct the course titled "Database Systems".

```
SELECT    *
FROM      Teacher
WHERE course-no IN ( SELECT    course-no
                     FROM      Course
                WHERE  title = 'Database Systems')
```

2. List all the course offered in 'Thane' on 15/8/15.

```
SELECT    *
FROM      Teacher
WHERE     location LIKE 'Thane'
AND       off-date = '15-08-2015'
```

3. Find the course enrolled by "Monali".

```
SELECT    *
FROM      Course
WHERE     course-no IN
(SELECT   course-no
FROM      Enrolment
WHERE     stud-no IN
(SELECT   stud-no
FROM      Student
WHERE     stud-name = 'Monali'))
```

4. List all the employees who work as Teachers.

```
SELECT    *
FROM      Employee
WHERE     emp-no IN ( SELECT  emp-no
                      FROM    Teacher )
```

Example 7.9.8 : Consider the following employee database
Employee (empname, street, city, date of join)
Works (empname, company-name, salary)
Company (company-name, city)
Manages (empname, manager-name)
Write SQL queries for the following statements :

1. Find all employees who joined in the onth of October.
2. Modify the" database so that 'Peter' now lives in 'Newton'.
3. List all employees who live in the same cities as their managers.
4. Find all employees who earn more than average salary of all employees of their company.
5. Give all employees of XYZ corporation a 15 percent raise.

MU - May 15, 10 Marks

Solution :

1. Find all employees who joined in the month of October.

```
SELECT    *
FROM      Employee
WHERE     MONTH (date of join) = 10;
```

2. Modify the" database so that 'Peter' now lives in 'Newton'.

```
UPDATE    Employee
SET city= 'Newton'
WHERE     empname = 'Peter';
```

3. List all employees who live in the same cities as their managers.

```
SELECT    *
FROM      Employee e
WHERE e.  city = (SELECT emp.city
FROM Employee emp, Manages mgr
WHERE emp.empname=mgr.empname
AND       emp.empname=e.empname   )
```

4. Find all employees who earn more than average salary of all employees of their company.

```
SELECT  e.empname
FROM    Employee e NATURAL JOIN Works w
WHERE   empname > (SELECT avg (salary)
                   FROM   Works w
                WHERE w.empname=e.empname)
```

5. Give all employees of XYZ Corporation a 15 percent raise.

```
UPDATE    Employee
SET       salary = 1.15*salary
WHERE     empname > (SELECT    empname
                     FROM      Works w
                  WHERE  company-name= 'XYZ corporation')
```

Example 7.9.9 : Consider the following employee database

Employee (emp_name,street,city,date_of_joining)

Works (emp name, company_name, salary)

Company (company_name, city)

Manages (emp _ name,manager_name)

Write SQL queries for following : **MU - May 16, 10 Marks**

1. Modify the database so that 'Deepa' lives in 'Pune.

Solution :

```
UPDATE    Employee
SET       city ='Pune'
WHERE     emp_name= 'Deepa';
```

2. Give all employees of 'XYZ corporation' a 10% rise in salary.

```
MySQL> SELECT   *
       FROM     Employee E, Manages M
       WHERE    E.empname = M.empname
       AND      E.city = M.city);
```

3. List all employees who lives in the same city as their companycity.

```
MySQL> SELECT   *
       FROM     Employee E, Manages M
       WHERE    E.empname = M.empname
       AND      E.city = M.city);
```

4. Display all employees who joined in the month of 'March'.

```
SELECT    *
FROM      Employee
WHERE     month(date_of_joining) = 3;
```

Find all employees who earn more than average salary of all employees of their company.

(Ref. Dec. 2013 Q. 6 Chapt 7)

```
MySQL>    SELECT    *
          FROM      Employee E, Works W
          WHERE     E.empname= W.empname
          AND   W.salary > (SELECT AVG(salary)
                            FROM Employee
       WHERE company_name = W.company_name);
```

Example 7.9.10 : Consider the following schema for institute library :

Student (RollNo, Name, Father_Name, Branch)

Book (ISBN, Title, Author, Publisher)

Issue (Roll No, ISBN, Date-of-Issue)

Write the following queries in relational algebra :

1. List Roll Number and Name of all students of the branch 'CSE'.
2. Find the name of students who have issued a book published by 'ABC' publisher.
3. List title of all books and their authors issued by a student 'XYZ'.
4. List title of all books issued on or before Jan 1, 2011. **MU - May 19, 10 Marks**

Solution :

1. List Roll Number and Name of all students of the branch 'CSE'

```
SELECT    RollNo. ,Name
FROM      student
WHERE     Branch = 'CSE'
```

2. Find the name of students who have issued a book published by 'ABC' publisher.

```
SELECT   Name
FROM     student
WHERE    RollNo = (SELECT   RollNo
FROM     issue where ISBN = (SELECT ISBN
                             FROM   Book
WHERE    publisher = 'ABC');
```

3. List title of all books and their authors issued by a student 'XYZ'.

```
SELECT   Title, Author
FROM     Book
WHERE    ISBN = (SELECT   ISBN
                 FROM     issue
          WHERE   rollno = (select rollNo
          FROM    student
          WHERE   name = 'Prashant';
```

4. List title of all books issued on or before Jan 1, 2011.

```
SELECT   title
FROM     book
WHERE    ISBN = (SELECT   ISBN
                 FROM     issue
                 WHERE    Date : of-issue
                          = 01.01.2011);
```

Example 7.9.11 : Consider the following scheme for college library.

Student (Roll_no, Name, Branch)

Book (ISBN, Title, Author, Publisher)

Issue (Roll_no, ISBN, Date_of_Issue)

Write SQL queries for the following statements :

1. List Roll Number and Name of all students of the branch IT.
2. Find the name of students who have issued a book published by 'XYZ' publisher.
3. List title of all books and their author issued by student 'Alice'
4. List title of all books issued on or before 31st DEC, 2019

MU - Dec.19,10 Marks

Solution :

1. List Roll Number and Name of all students of the branch IT.

```
SELECT   RollNo. ,Name
FROM     student
WHERE    Branch = 'IT'
```

2. Find the name of students who have issued a book published by 'XYZ' publisher

```
SELECT   Name
FROM     student
WHERE    RollNo = (SELECT   RollNo
         FROM     issue
         WHERE ISBN =  (SELECT   ISBN
                        FROM     Book
                        WHERE publisher = 'XYZ');
```

3. List title of all books and their author issued by student 'Alice'

```
SELECT   *
FROM     Book
WHERE    ISBN = (SELECT   ISBN
                 FROM     issue
                 WHERE    rollno =
          (SELECT   rollNo
          FROM     student
          WHERE    name = 'Alice';
```

4. List title of all books issued on or before 31st DEC, 2019

```
SELECT   *
FROM     book
WHERE    ISBN = (SELECT   ISBN
                 FROM     issue
          WHERE   Date : of-issue = 31.12.2019);
```

Review Questions

1. What is outer join ?
2. Explain external join. How is it different from natural join ?
3. Explain the concepts of Outer joins.
4. Explain self join with suitable Example.
5. What are JOINS ? What are different types of JOINS give example of each.

6. What is outer join and why it is used ? Give explain its types with example.

7. Give difference between.
   a. Inner Join and outer Join
   b. Left Outer Join and Right outer Join
   c. Equi Join and Non Equi Join
   d. Equi Join and Self Join

8. What is a view ? How is it created and stored ?

9 Specify the following queries in SQL based on given schema :

   Suppliers (sid, sname, city)

   Part (pid, pname, color)

   Sp (sid, pid, quantity)

   1. Get the names of suppliers whose name begins with R.
   2. Get pairs of supplier numbers that both operate from same city.
   3. Get the names of suppliers who supply part P2.
   4. Get the supplier id and names of suppliers in descending order of city.
   5. Get the part numbers and total quantity supplied.

10. For the following relational schema :

    Employee (EmpId, EmpName, Street, City)

    Works (EmpId, CompId, Salary)

    Company (CompId, CompName, City)

    Give an expression in SQL for each of the following queries :

    1. Find the names of employees whose salary is more than the average salary of all employees of their company.
    2. Find the names of companies that has the largest number of employees.
    3. Find names of employees whose salary is more than that of every employee of company named 'ABCD'.
    4. Find names of companies located in the city in which company 'ABCD' is located.
    5. Give all employees a 20 percent raise in salary whose city is different from the city they work in.

11. Consider the following schema :

    EMPLOYEE (EID, EmployeeName, Street, City, Deptt, names of all employees that have 'A' CompanyName)

    COMPANY (CompanyName, City)

    WORKS (EmployeeName, CompanyName, Salary)

    MANAGES (EmployeeName, ManagerName)

    Write SQL queries for the following :

    1. Find out the anywhere in their name and are in department 'IT'.
    2. List the names of departments in ascending order and their employees in descending order.
    3. Find the names, city, deptt of all employee who work for 'TCS'
    4. Find the name of employee who earns salary more than 30000.
    5. List all manager names.

12. Explain concept of sub query.

13. Describe various operators for multiple sub query.

14. What is nested sub query ? Explain ANY ALL operators with example.

15. What is correlated sub query ? Explain with example.

7.10 University Questions and Answers

May 2015

1. Consider the following employee database

   Employee (empname, street, city, date of join)

   Works (empname, company-name, salary)

   Company (company-name, city)

   Manages (empname, manager-name)

   Write SQL queries for the following statements :

   1. Find all employees who joined in the month of October.
   2. Modify the" database so that 'Peter' now lives in 'Newton'.
   3. List all employees who live in the same cities as their managers.
   4. Find all employees who earn more than average salary of all employees of their company.

5. Give all employees of XYZ corporation a 15 percent raise. **(10 Marks)**

Dec. 2015

2. Refer education database mentioned in Q. 2 (a). Write SQL queries for the following :

1. List all the teacher who conduct the course titled Database Systems.
2. List all the course offered in Thane on 15/8/15.
3. Find the course enrolled by Monali.
4. List all the employees who work as Teachers. **(10 Marks)**

May 2016

3. What is JOIN ? Explain different types of JOIN along with example. **(10 Marks)**

4. Consider the following employee database

Employee (emp_name,street,city,date_of_joining)

Works (emp name, company_name, salary)

Company (company_name, city)

Manages (emp _ name,manager_name)

Write SQL queries for following :

1. Modify the database so that 'Deepa' lives in 'Pune.
2. Give all employees of 'XYZ corporation' a 10% rise in salary.
3. List all employees who lives in the same city as their company city.
4. Display all employees who joined in the month of 'March'.
5. Find all employees who earn more than average salary of all employees of their company. **(10 Marks)**

Dec. 2017

5. Explain Recursive queries and Nested queries. **(5 Marks)**

May 2019

6. Define the following term : Nested queries. **(5 Marks)**

❑❑❑