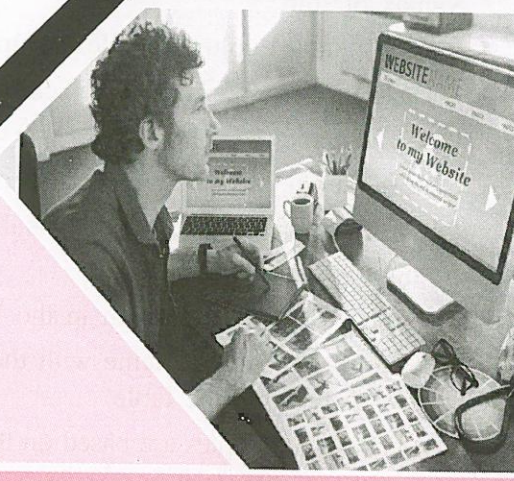
# CHAPTER 26




# JOINING OF TABLES

## CHAPTER OBJECTIVES

**In this chapter you will learn:**

- Need for joining of tables
- Different types of join
- Usage of Cartesian product
- Usage of Inner Join
- Usage of Outer Join
- Usage of Self Join

## 26.1 JOINING OF TABLES FOR MULTIPLE TABLE QUERIES

In Relational Database Management Systems, data stored in different tables is related. We can use the power of SQL to relate the information and query data. A SELECT statement has a mandatory SELECT clause and FROM clause. The SELECT clause can have a list of columns, expressions, functions, and so on. The FROM clause tells you which table(s) to look in for the required information. So far, we have seen only one table in the FROM clause, now we will learn how to retrieve data from more than one table. In order to query data from more than one table, we need to identify a common column that relates the two tables. In the WHERE clause, we define the relationship between the tables listed in the FROM clause using comparison operators.

When data from more than one table in the database is required, a join condition is used. Rows in one table can be joined to rows in another table according to common values existing in corresponding columns, that is, usually primary and foreign key columns. A simple join

condition in the WHERE clause. Oracle performs a join whenever multiple tables appear in the query's FROM clause. The query's SELECT clause can have the columns or expressions from any or all of these tables.

Syntax of Join

SELECT table1.column, table2.column FROM table1, table2

WHERE table1.column1=table2.column2;

In the syntax:

table1.column, table2.column denotes the table and column from which data is retrieved

table1.column1= table2.column2 is the condition that joins (or relates) the tables together

Points to Note:

- Write the join condition in the WHERE clause.
- Prefix the column name with the table name when the same column name appears in more than one table.

Here, all the examples are based on the EMP and DEPT tables, whose data shown below:

SQL>SELECT * FROM dept;

The output is:

| DEPTNO | DNAME | LOC |
|---|---|---|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |
| 50 | PAYROLL | DALLAS |

SQL>SELECT * FROM emp;

The output is:

| EMPNO | ENAME | SALARY | COMM | DEPTNO |
|---|---|---|---|---|
| 7566 | JONES | 2975 | | 20 |
| 7654 | MARTIN | 1250 | 1400 | 30 |
| 7698 | K_BLAKE | 2850 | | 30 |
| 7788 | SCOTT | 3000 | | 20 |
| 7839 | A_EDWARD | 5000 | 50000 | 10 |
| 7844 | TURNER | 1500 | 0 | 30 |
| 7845 | FORD | 3000 | | 20 |

How would we list the department name location for each employee, along with his or her salary? The department name and location are in the DEPT table; the employee name and salary are in the EMP table. So, to list the information together in one query, we need to do a join. The DEPTNO column is common to both tables; use this column to relate the rows.

To get the required information following query is used:

```
SQL>SELECT dname, loc, ename, sal FROM dept, emp
    WHERE dept.deptno = emp.deptno;
```

The Output is:

| DNAME | LOC | ENAME | SALARY |
|---|---|---|---|
| RESEARCH | DALLAS | JONES | 2975 |
| SALES | CHICAGO | MARTIN | 1250 |
| SALES | CHICAGO | K_BLAKE | 2850 |
| RESEARCH | DALLAS | SCOTT | 3000 |
| ACCOUNTING | NEW YORK | A_EDWARD | 5000 |
| SALES | CHICAGO | TURNER | 1500 |
| RESEARCH | DALLAS | FORD | 3000 |

7 rows selected.

Here data is selected from two tables: EMP and DEPT. The department number (DEPTNO) is the column on which join is established. Notice that in the WHERE clause, the column names are qualified by the table name; this is required to avoid ambiguity, because the column names are the same in both tables. If the column names are different in each table, you need not qualify the column names. Just as we can provide column alias names, we can alias table names, also. Aliases improve the readability of the code, and they can be short names that are easy to type and use as references. The table alias name is given next to the table name.

The following example uses alias names d and e for DEPT and EMP tables and uses them to qualify the column names:

```
SQL> SELECT d.name, d.loc, e.ename, e.sal
     FROM dept d, emp e
     WHERE d.deptno = e.deptno
     ORDER BY d.dname;
```

**NOTE:** Once table alias names are defined; you cannot use the table name to qualify a column. You should use the alias name to qualify the column.

To execute a join of three or more tables, Oracle takes these steps:

- Oracle joins two of the based tables on the join conditions, comparing their columns
- Oracle joins the result to another table, based on join conditions.
- Oracle continues this process until all tables are joined into the result.

The Join query also contained in the WHERE clause to restrict rows, based on column in one table. Here's example:

```
SQL> SELECT d.dname, d.loc, e.ename, e.sal
     FROM dept d, emp e
     WHERE e.deptno = d.deptno and comm IS NOT NULL
```

The Output is:

| DNAME | LOC | ENAME | SAL |
|---|---|---|---|
| SALES | CHICAGO | ALLEN | 1600 |
| SALES | CHICAGO | WARD | 1250 |
| SALES | CHICAGO | MARTIN | 1250 |
| SALES | CHICAGO | TURNER | 1500 |

## 26.2 TYPES OF JOINS

Following are the type of join.

- Cartesian Product
- Inner join
  - Equi join
  - Non-equi join
- Outer join
  - Left-outer join
  - Right-outer join
- Self join

### 26.2.1 Cartesian product

A Cartesian join occurs when data is selected from two or more tables and there is no common relation specified in the WHERE clause. If we do not specify a join condition for the tables listed in the FROM clause, Oracle joins each row from the first table to every row in the second table. In Cartesian join each row from the first table is combined with all rows from the second table.

- The working of Cartesian product is illustrated below with example database.

**Select * from Table 1, Table 2;**

Table 1

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a2 | b2 | c2 |

Table 2

| X | Y |
|---|---|
| x1 | y1 |
| x2 | y2 |

Cartesian Product
(m* n) rows

| A | B | C | X | Y |
|---|---|---|---|---|
| a1 | b1 | c1 | x1 | y1 |
| a2 | b2 | c2 | x2 | y2 |
| a3 | b3 | c3 | x3 | y3 |
| a4 | b4 | c4 | x4 | y4 |

Product of Table 1 and Table 2

If the first table has three rows and the second table has four rows, the result will have 12 rows. Suppose we add another table with two rows without specifying a join condition; the result will have 24 rows. We should avoid Cartesian joins; for the most part, they happen when there are many tables in the FROM clause and developers forget to include the join condition.

Example:

```
SQL>Select ename,dname FROM emp, dept;
```

The above query displays employee name and department name from EMP and DEPT tables. Because no WHERE clause has been specified, all rows(14 rows) from EMP table are joined with all (4 rows) in the DEPT table, there by generating 56 rows in the output as shown below:

**EMP (14 rows)**

| EMPNO | ENAME | DEPTNO |
|---|---|---|
| 7839 | KING | 10 |
| 7698 | BLAKE | 30 |
| ..... | ..... | ..... |
| 7934 | MILLER | 10 |

**DEPT (4 rows)**

| DEPTNO | DNAME | LOC |
|---|---|---|
| 10 | ACCOUNTING | NEW YORK |
| 20 | RESEARCH | DALLAS |
| 30 | SALES | CHICAGO |
| 40 | OPERATIONS | BOSTON |

| ENAME | DNAME |
|---|---|
| KING | ACCOUNTING |
| BLAKE | RESEARCH |
| ..... | SALES |
| MILLER | OPERATIONS |

Cartesian product 14*4=56 rows selected

## 26.2.2 Inner Join

An inner join between two (or more) tables is the Cartesian product that satisfies the join condition in the WHERE clause. It is of two types, depending upon the WHERE consition.

### 26.2.2.1 Equality Joins

If the WHERE clause in query of relating two tables used an equality operator (=), it is an equality join, also known as an inner join or an equijoin. As illustrated below.

**Get all combinations of emp and cust information such that the emp and cust are co-located.**

```
SELECT Table 1. Emp_ID, Table 1.City, Table 2. Cust
    FROM Table 1, Table 2
        WHERE Table 1, City = Table 2. City;
```

Table 1

| Emp_ID | CITY |
|---|---|
| A1 | New York |
| A2 | NULL |
| A3 | Chicago |
| A4 | Chicago |
| A5 | Paris |

Table 2

| Cust_ID | CITY |
|---|---|
| B1 | New York |
| B2 | New York |
| B3 | NULL |
| B4 | Chicago |
| B5 | Moscow |

INNER JOIN

Output Table

| Table 1. Emp_ID | Table 1. CITY | Table 2. Emp_ID | Table 2. CITY |
|---|---|---|---|
| A1 | New York | B1 | New York |
| A2 | NULL | B2 | New York |
| A3 | Chicago | B3 | NULL |
| A4 | Chicago | B4 | Chicago |

The query discussed earlier to list the department name location for each employee, along with his or her salary is an example of equality join.

To get the required information following query is used:

```
SQL>SELECT dname, loc, ename, sal FROM dept, emp
        WHERE dept.deptno = emp.deptno;
```

### 26.2.2.2 Non-Equality Join

If any other operator instead of equality operator (=) is used to join the tables in the query, it is non-equality join. We have already seen examples of equality joins; let's consider an example of non-equality join.

Example:

**EMP**

| EMPNO | ENAME | SAL |
|---|---|---|
| 7839 | KING | 5000 |
| 7698 | BLAKE | 2850 |
| 7782 | CLARK | 2450 |
| 7566 | JONES | 2975 |
| 7654 | MARTIN | 1250 |
| 7499 | ALLEN | 1600 |
| 7844 | TURNER | 1500 |
| 7900 | JAMES | 950 |

**SAL GRADE**

| GRADE | LOSAL | HISAL |
|---|---|---|
| 1 | 700 | 1200 |
| 2 | 1201 | 1400 |
| 3 | 1401 | 2000 |
| 4 | 2001 | 3000 |
| 5 | 3001 | 9999 |

Salary in the EMP table is between
low salary and high salary in the SALGRADE table

The relationship between the EMP table and the SALGRADE table is a non-equijoin, meaning that no column in the EMP table corresponds directly to a column in the SALGRADE table. The relationship between the two tables is that the SAL column in the EMP table is between the LOSAL and HISAL column of the SALGRADE table. The relationship is obtained using an operator other than equal (=).

```
SQL> SELECT e.ename, e.sal, s.grade from emp e, salgrade s
        WHERE e.sal BETWEEN s.losal and S.hisal;
```

The output is:

| ENAME | SAL | GRADE |
|---|---|---|
| JAMES | 950 | 1 |
| SMITH | 800 | 1 |
| ADAMS | 1100 | 1 |
| ..... | ..... | ..... |

14 rows selected.

The above example creates a non-equijoin to evaluate an employee's grade. The salary must be between any pair of the low and high salary ranges.

### 26.2.3 Outer Joins

It retrieve all rows that match the WHERE clause and also those that have a NULL value in the column used for join. The concept and need of outer join has been explained below.

Sometimes, we might want to see the data from one table, even if there is no corresponding row in the joining table. Oracle provides the outer join mechanism for this.

Such rows can be forcefully selected by using the outer join symbol (+). The corresponding columns for that row will have Nulls. For example, to write a query that performs an outer join of tables A and B and returns all rows from A, apply the outer-join operator (+) to all columns of B in the join condition. For all rows in A that have no matching rows in B, the query returns NULL values for the columns in B.

Example: In the emp table, no record of the employee belongs to the department 40. Therefore, in case of equi join, the row of department 40 from the dept table will not be displayed. In order to include that row in the output Outer join is used.

- Display the list of employees working in each department. Display the department information even if no employee belongs to that department.

```
SQL>SELECT empno, ename, emp.deptno, dname, loc FROM emp, dept
      WHERE emp.deptno (+) = dept.deptno;
```

The output is:

| EMPNO. | ENAME | EMP.DEPTNO | DNAME |
|---|---|---|---|
| 7369 | SMITH | 20 | RESEARCH |
| 7499 | ALLEN | 30 | SALES |
| 7521 | WARD | 30 | SALES |
| 7566 | JONES | 20 | RESEARCH |
| 7654 | MARTIN | 30 | SALES |
| 7698 | BLAKE | 30 | SALES |
| 7782 | CLARK | 10 | ACCOUNTING |
| 7788 | SCOTT | 20 | RESEARCH |
| 7839 | KING | 10 | ACCOUNTING |
| 7844 | TURNER | 30 | SALES |
| 7876 | ADAMS | 20 | RESEARCH |
| 7900 | JAMES | 30 | SALES |
| 7902 | FORD | 20 | RESEARCH |
| 7934 | MILLER | 10 | ACCOUNTING |
| 7945 | ALLEN | 20 | ACCOUNTING |
| 7526 | MARTIN | 20 | RESEARCH |
| 7985 | SCOTT | 30 | SALES |
| | | 40 | OPERATIONS |

If the symbol (+) is placed on the other side of the equation then all the employee details with no corresponding department name and location, will be displayed with NULL values in DNAME and LOC column.

Outer Join is of two types, i.e., left outer join and right outer join.

### 26.2.3.1 Left/Right-Outer join

Left outer joins include all records from the first (left) of two tables, A = B (+), while right outer joins include all records from the second (right) of two tables, A (+) = B.

Example: Can you answer, the earlier example of outer join is left outer join or right outer join?

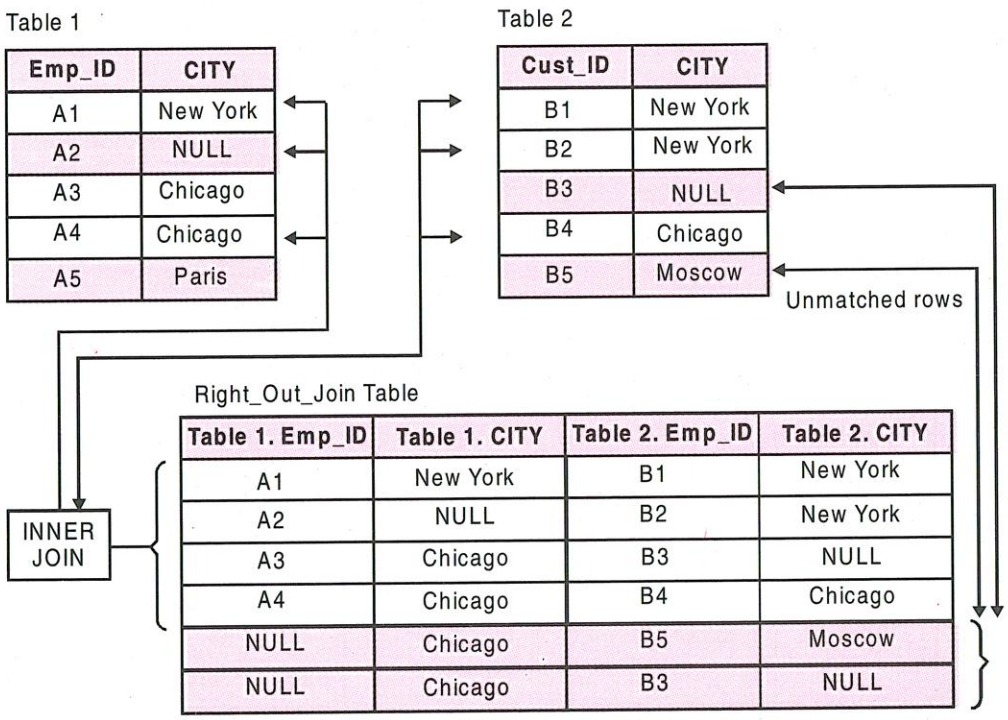
```
SQL>SELECT empno, ename, emp.deptno, dname, loc FROM emp, dept
        WHERE emp.deptno (+) = dept.deptno;
```

Yes, it is right outer join, as full right table appears in the output by having NULL corresponding to missing values of EMP table. Here, right table appears fully and + appears on left side, so it is right outer join. In simple words, in right outer join plus appears on left side, while in left outer join plus appears on right side.

A case of left outer join has been illustrated below.

**List all cities of Table 1 if there is match in cities in Table 2 & also unmatched Cities from Table 1**

```
SELECT Table 1. Emp_ID, Table 1.City, Table 2. Cust_ID, Table 2. City
        FROM Table 1, Table 2
                WHERE Table 1, City = Table 2. City (+);
```



A case of right outer join on same database has been illustrated below.

```
SELECT Table 1. Emp_ID, Table 1.City, Table 2. Cust_ID, Table 2. City
        FROM Table 1, Table 2
                WHERE Table 1, City(+) = Table 2.City;
```

Table 1

| Emp_ID | CITY |
|---|---|
| A1 | New York |
| A2 | NULL |
| A3 | Chicago |
| A4 | Chicago |
| A5 | Paris |

Table 2

| Cust_ID | CITY |
|---|---|
| B1 | New York |
| B2 | New York |
| B3 | NULL |
| B4 | Chicago |
| B5 | Moscow |

Unmatched rows

INNER JOIN

Right_Out_Join Table

| Table 1. Emp_ID | Table 1. CITY | Table 2. Emp_ID | Table 2. CITY |
|---|---|---|---|
| A1 | New York | B1 | New York |
| A2 | NULL | B2 | New York |
| A3 | Chicago | B3 | NULL |
| A4 | Chicago | B4 | Chicago |
| NULL | Chicago | B5 | Moscow |
| NULL | Chicago | B3 | NULL |

Example of Left Outer Join

- List all customer details and loan details if they have availed loans.

```
SQL>Select Customer_details.Cust_id,Cust_Last_name,Loan_no,Amount_in_dollars
from Customer_details,Customer_loan
where Customer_details.Cust_id = Customer_loan.Cust_id (+);
```

**Rules to Place (+) operator:**

- The outer join symbol (+) cannot be on both sides.
- We cannot "outer join" the same table to more than one other table in a single SELECT statement.
- A condition involving an outer join may not use the IN operator or be linked to another condition by the OR operator.

## 26.2.4 Self-Join

To join a table to itself means that each row of the table is combined with itself and with every other row of the table. The self-join can be viewed as a join of two copies of the same table. The table is not actually copied, but SQL performs the command as though it were.

The syntax of the command for joining a table to itself is almost the same as that for joining two different tables. To distinguish the column names from one another, aliases for the actual the table name are used, since both the tables have the same name. Table name aliases are defined in the FROM clause of the query. To define the alias, one space is left after the table name and the alias.

Example:

**EMP TABLE**

| EMPNO | ENAME | MGR |
|---|---|---|
| 7839 | KING | |
| 7566 | JONES | 7839 |
| 7876 | ADAMS | 7788 |
| 7934 | MILLER | 7782 |
| ..... | ..... | |

Consider the emp table shown above. Primary key of the emp table is empno. Details of each employee's manager is just another row in the EMP table whose EMPNO is stored in MGR column of some other row. So every employee except manager has a Manager. Therefore MGR is a foreign key that references empno. To list out the names of the manager with the employee record one will have to join EMP itself.

```
SQL>SELECT WORKER. Ename "Ename", MANAGER.ename "Manager"
     FROM emp WORKER, emp MANAGER
     WHERE WORKER.mgr=MANAGER.empno;
```

Where WORKER and MANAGER are two aliases for the EMP table and acts as a virtual tables.

**WORKER**

| EMPNO | ENAME | MGR |
|---|---|---|
| 7369 | SMITH | 7902 |
| 7499 | ALLEN | 7698 |
| 7521 | WARD | 7698 |
| 7566 | JONES | 7839 |
| 7654 | MARTIN | 7698 |
| 7698 | BLAKE | 7839 |
| 7782 | CLARK | 7839 |
| 7788 | SCOTT | 7566 |
| 7839 | KING | |
| 7844 | TURNER | 7698 |
| 7876 | ADAMS | 7788 |
| 7900 | JAMES | 7698 |
| 7902 | FORD | 7566 |
| 7934 | MILLER | 7782 |

**MANAGER**

| EMPNO | ENAME | MGR |
|---|---|---|
| 7369 | SMITH | 7902 |
| 7499 | ALLEN | 7698 |
| 7521 | WARD | 7698 |
| 7566 | JONES | 7839 |
| 7654 | MARTIN | 7698 |
| 7698 | BLAKE | 7839 |
| 7782 | CLARK | 7839 |
| 7788 | SCOTT | 7566 |
| 7839 | KING | |
| 7844 | TURNER | 7698 |
| 7876 | ADAMS | 7788 |
| 7900 | JAMES | 7698 |
| 7902 | FORD | 7566 |
| 7934 | MILLER | 7782 |

**The output will be:**

| Ename | Manager |
|---|---|
| SMITH | FORD |
| ALLEN | BLAKE |
| WARD | BLAKE |
| JONES | KING |
| MARTIN | BLAKE |
| BLAKE | KING |
| CLARK | KING |
| SCOTT | JONES |
| TURNER | BLAKE |
| ADAMS | SCOTT |
| JAMES | BLAKE |
| FORD | JONES |
| MILLER | CLARK |

13 rows selected.

- List all employees who joined the company before their manager.

```
SQL>Select e.ename, e.hiredate, m.ename manager, m.hiredate FROM emp e, emp m
    WHERE e.mgr= m.empno
    and e.hiredate<m.hiredate;
```

The output is:

| ENAME | HIREDATE | MANAGER | HIREDATE |
|---|---|---|---|
| SMITH | 17-DEC-80 | FORD | 03-DEC-81 |
| ALLEN | 20-FEB-81 | BLAKE | 01-MAY-81 |
| WARD | 22-FEB-81 | BLAKE | 01-MAY-81 |
| JONES | 02-APR-81 | KING | 17-NOV-81 |
| BLAKE | 01-MAY-81 | KING | 17-NOV-81 |
| CLARK | 09-JUN-81 | KING | 17-NOV-81 |

6 rows selected.

**Note:** If we wish to include those employees name who has no corresponding manager also in above list. Then it becomes the case of self join and outer join. In this scenario, we wish to list all the employees whether it has manager or not. So, worker table, i.e., left table appears full and (+) will appear on manager side so it becomes the case of right outer join. And corresponding query has been shown below.

```
SQL>SELECT WORKER. Ename "Ename", MANAGER.ename "Manager"
    FROM emp WORKER, emp MANAGER
    WHERE WORKER.mgr=MANAGER.empno(+);
```

The output of query will be as shown below.

**The output will be:**

| Ename | Manager |
|---|---|
| SMITH | FORD |
| ALLEN | BLAKE |
| WARD | BLAKE |
| JONES | KING |
| MARTIN | BLAKE |
| BLAKE | KING |
| CLARK | KING |
| SCOTT | JONES |
| TURNER | BLAKE |
| ADAMS | SCOTT |
| JAMES | BLAKE |
| FORD | JONES |
| MILLER | CLARK |
| KING | |

14 rows selected.

It shows that KING appears in the list and he has no corresponding manager or in simple words he is at top of hierarchy.

- List all employees who joined the company before their manager.

```
SQL>Select e.ename, e.hiredate, m.ename manager, m.hiredate FROM emp e, emp m
    WHERE e.mgr= m.empno
    and e.hiredate<m.hiredate;
```

**FLASH BACK**

In Relational Database Management Systems, data stored in different tables is related. We can use the power of SQL to relate the information and query data. There are following types of Joins Equi join or Equality Join or Inner join, Non-equijoin or Non-equality join, Outer join and Self Join. A Cartesian join occurs when data is selected from two or more tables and there is no common relation specified in the WHERE clause. If we do not specify a join condition for the tables listed in the FROM clause, Oracle joins each row from the first table to every row in the second table.

# HANDS ON SESSION

**Lab Assignmnet-5**

Solve the following on the given database:

S(Sno, Sname, City, Status)

P(Pno, Pname, Color, Weight)

SP(Sno, Pno, Qty)

1. For each part supplied get part number and names of all cities supplying the part.
2. Get Qty supplied for Red parts.
3. Get Sname supplying Red part.
4. Get Sname, Pname who supply qty more than 100.
5. Get all pairs of supplier's numbers such that the two suppliers are located in the same city.
6. Get Sname for suppliers who supply part P2.
7. Get Supplier numbers for suppliers who supply at least one part supplied by supplier S2.
8. Get supplier names for suppliers who do not supply part P2.
9. Get suppliers numbers for suppliers who are located in same city as supplier S1.
10. Get all possible parts which can be supplied by suppliers of Patiala

## Queries Based on Emp table

11. Get ename and mgrname from emp table
12. Get name of emp who join the company before their managers
13. Get dname and ename also get those deptt which has no emp

○○○○○