



**GOVERNMENT OF KARNATAKA  
DEPARTMENT OF COLLEGIATE EDUCATION  
GOVERNMENT FIRST GRADE COLLEGE, RAIBAG, BELAGAVI – 591317**

**Department of Computer Science**

# **Lab Manual**

**DATABASE MANAGEMENT SYSTEM**

**BSC-V Semester**

**Subject Incharge: Dr. Bhagirathi Halalli**

## INTRODUCTION TO SQL

Pronounced as SEQUEL: Structured English QUERY Language

- Pure non-procedural query language
- Designed and developed by IBM, Implemented by Oracle
- 1978 System/R IBM- 1st Relational DBMS
- 1979 Oracle and Ingres
- 1982 SQL/DS and DB2 IBM
- Accepted by both ANSI + ISO as **Standard Query Language** for any RDBMS
- SQL86 (SQL1) : first by ANSI and ratified by ISO (SQL-87), minor revision on 89 (SQL-89)
- SQL92 (SQL2) : major revision
- SQL99 (SQL3) : add recursive query, trigger, some OO features, and non-scholar type
- SQL2003 : XML, Window functions, and sequences (Not free)
- Supports all the three sublanguages of DBMS: **DDL, DML, DCL**
- Supports Aggregate functions, String Manipulation functions, Set theory operations, Date Manipulation functions, rich set of operators ( IN, BETWEEN, LIKE, IS NULL, EXISTS)
- Supports REPORT writing features and Forms for designing GUI based applications

### Data Definition in SQL

#### CREATE, ALTER and DROP

table.....relation  
 row.....tuple  
 column.....attribute

### DATA TYPES

- Numeric: NUMBER, NUMBER(s,p), INTEGER, INT, FLOAT, DECIMAL
- Character: CHAR(n), VARCHAR(n), VARCHAR2(n), CHAR VARYING(n)
- Bit String: BLOB, CLOB
- Boolean: true, false, and null

## List of Experiments

### 17BScCSCT52: Programming Lab- SQL and PL/SQL Lab.

Practical Hours: 4 Hrs/week

arks: Main exam: 40

IA: 10

1. Draw E-R diagram and convert entities and relationships to relation table for a given scenario.
  - a. Two assignments shall be carried out i.e. consider two different scenarios (eg. bank, college)
2. Write relational algebra queries for a given set of relations.
3. Perform the following:
  - a. Viewing all databases, Creating a Database, Viewing all Tables in a Database, Creating Tables (With and Without Constraints), Inserting/Updating/Deleting Records in a Table, Saving (Commit) and Undoing (rollback)
4. Perform the following:
  - a. Altering a Table, Dropping/Truncating/Renaming Tables, Backing up / Restoring a Database.
5. For a given set of relation schemes, create tables and perform the following  
Simple Queries, Simple Queries with Aggregate functions, Queries with Aggregate functions (group by and having clause), Queries involving- Date Functions, String Functions , Math Functions  
Join Queries- Inner Join, Outer Join  
Subqueries- With IN clause, With EXISTS clause
6. For a given set of relation tables perform the following
  - a. Creating Views (with and without check option), Dropping views, Selecting from a view
7. Write a PL/SQL program using FOR loop to insert ten rows into a database table.
8. Given the table EMPLOYEE (EmpNo, Name, Salary, Designation, DeptID) write a cursor to select the five highest paid employees from the table.
9. Illustrate how you can embed PL/SQL in a high-level host language such as C/Java  
And demonstrates how a banking debit transaction might be done.
10. Given an integer i, write a PL/SQL procedure to insert the tuple (i, 'xxx') into a given relation.

SQL and PL/SQL tutorial: <https://www.w3schools.com/sql/>, <http://www.plsqltutorial.com/>

**Experiment 1:**

Consider following databases and draw ER diagram and convert entities and relationships to relation table for a given scenario.

**1. COLLEGE DATABASE:**

STUDENT (USN, SName, Address, Phone, Gender)

SEMSEC (SSID, Sem, Sec)

CLASS (USN, SSID)

SUBJECT (Subcode, Title, Sem, Credits)

IAMARKS (USN, Subcode, SSID, Test1, Test2, Test3, FinalIA)

**2. COMPANY DATABASE:**

EMPLOYEE (SSN, Name, Address, Sex, Salary, SuperSSN, DNo)

DEPARTMENT (DNo, DName, MgrSSN, MgrStartDate)

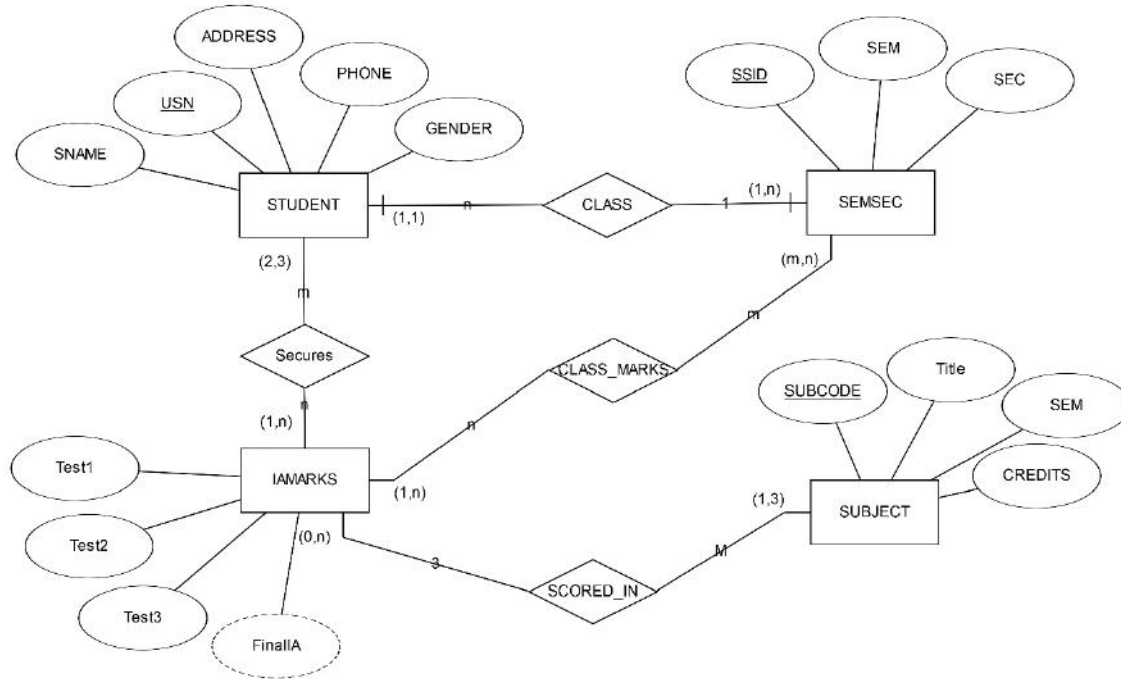
DLOCATION (DNo, DLoc)

PROJECT (PNo, PName, PLocation, DNo)

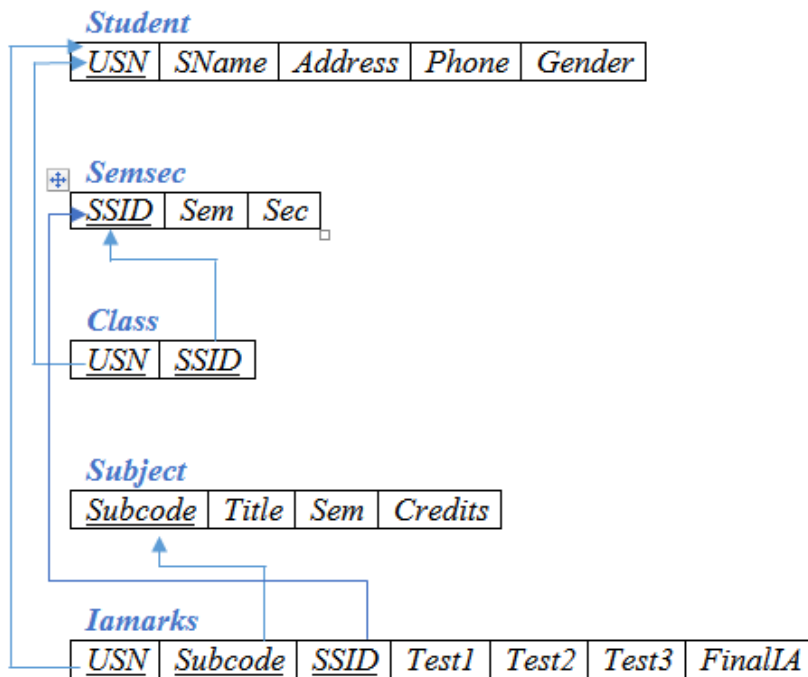
WORKS\_ON (SSN, PNo, Hours)

**SOLUTION:**

**College Database: E-R Diagram**

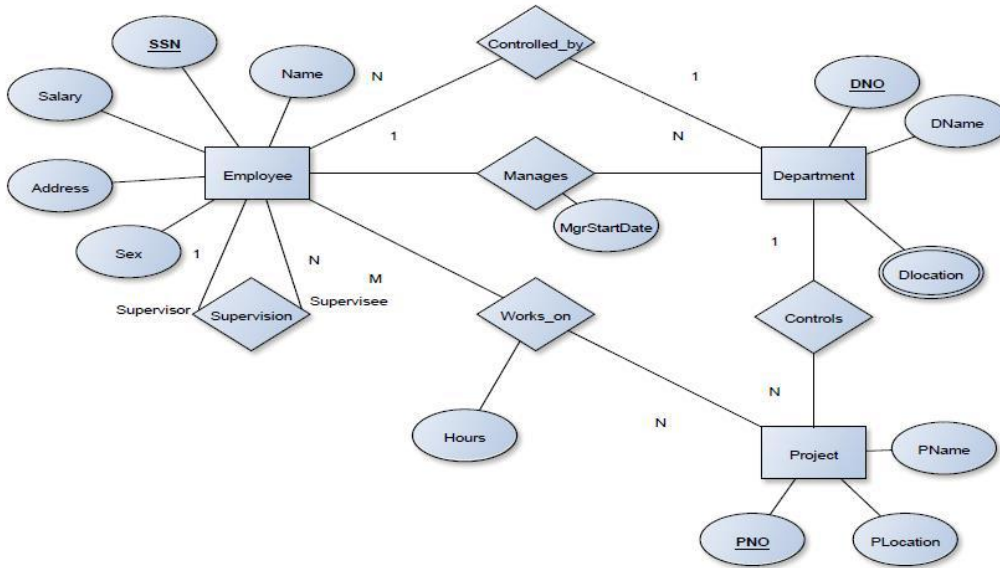


**Mapping entities and relationships to relation table (Schema Diagram)**

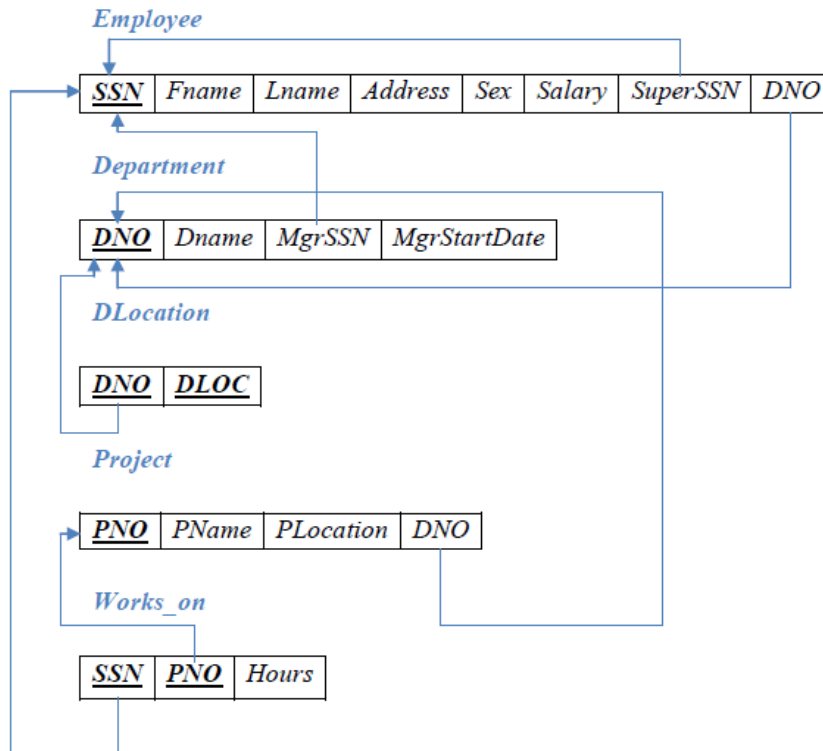


COMPANY DATABASE:

E-R Diagram



Schema Diagram



## Experiment 2

Consider the MOVIE DATABASE

Movies

| title           | director | myear | rating |
|-----------------|----------|-------|--------|
| Fargo           | Coen     | 1996  | 8.2    |
| Raising Arizona | Coen     | 1987  | 7.6    |
| Spiderman       | Raimi    | 2002  | 7.4    |
| Wonder Boys     | Hanson   | 2000  | 7.6    |

Actors

| actor     | ayear |
|-----------|-------|
| Cage      | 1964  |
| Hanks     | 1956  |
| Maguire   | 1975  |
| McDormand | 1957  |

Acts

| actor     | title           |
|-----------|-----------------|
| Cage      | Raising Arizona |
| Maguire   | Spiderman       |
| Maguire   | Wonder Boys     |
| McDormand | Fargo           |
| McDormand | Raising Arizona |
| McDormand | Wonder Boys     |

Directors

| director | dyear |
|----------|-------|
| Coen     | 1954  |
| Hanson   | 1945  |
| Raimi    | 1959  |

Write following relational algebra queries for a given set of relations.

1. Find movies made after 1997
2. Find movies made by Hanson after 1997
3. Find all movies and their ratings
4. Find all actors and directors
5. Find Coen's movies with McDormand

**SOLUTION:****Common notations of Relational Algebra**

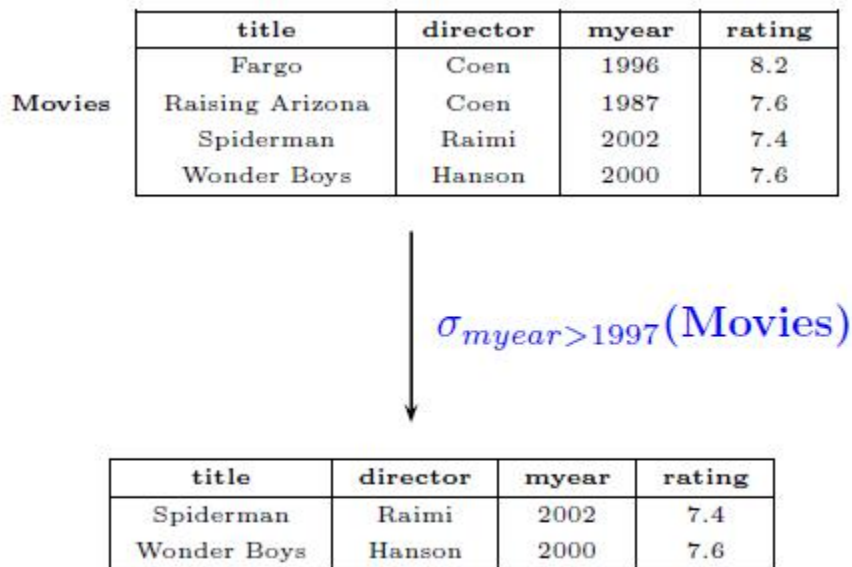
| Operation                     | Purpose  |
|-------------------------------|--|
| Select( $\sigma$ )            | The SELECT operation is used for selecting a subset of the tuples according to a given selection condition       |
| Projection( $\pi$ )           | The projection eliminates all attributes of the input relation but those mentioned in the projection list.       |
| Union Operation( $\cup$ )     | UNION is symbolized by symbol. It includes all tuples that are in tables A or in B.                              |
| Set Difference( $-$ )         | - Symbol denotes it. The result of $A - B$ , is a relation which includes all tuples that are in A but not in B. |
| Intersection( $\cap$ )        | Intersection defines a relation consisting of a set of all tuple that are in both A and B.                       |
| Cartesian Product( $\times$ ) | Cartesian operation is helpful to merge columns from two relations.  |
| Inner Join                    | Inner join, includes only those tuples that satisfy the matching criteria.                                       |
| Theta Join( $\theta$ )        | The general case of JOIN operation is called a Theta join. It is denoted by symbol $\theta$ .                    |
| EQUI Join                     | When a theta join uses only equivalence condition, it becomes a equi join.                                       |
| Natural Join( $\bowtie$ )     | Natural join can only be performed if there is a common attribute (column) between the relations.                |
| Outer Join                    | In an outer join, along with tuples that satisfy the matching criteria.  |
| Left Outer Join( $\ltimes$ )  | In the left outer join, operation allows keeping all tuple in the left relation.                                 |



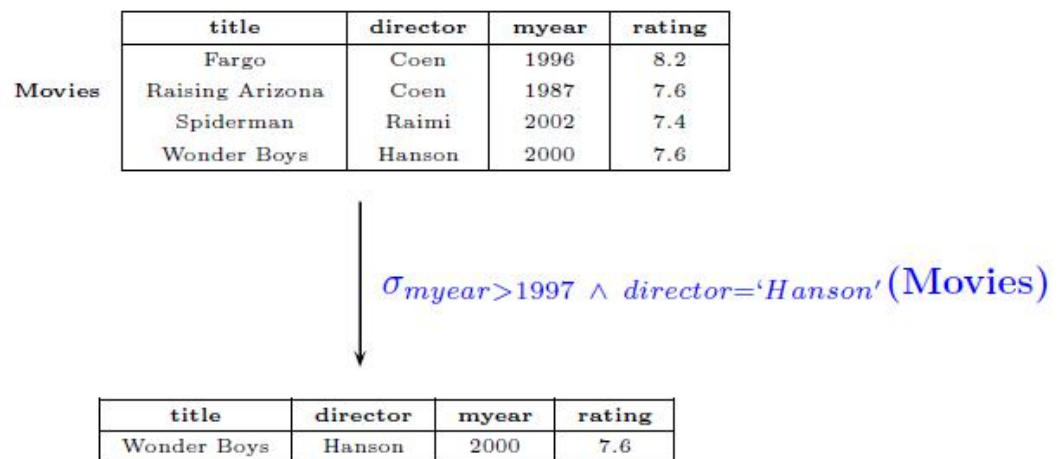
Right Outer join( $\bowtie_r$ ) In the right outer join, operation allows keeping all tuple in the right relation.

Full Outer Join( $\bowtie_{full}$ ) In a full outer join, all tuples from both relations are included in the result irrespective of the matching condition.

1. Find movies made after 1997  
 $\sigma_{myear>1997}(\text{Movies})$



2. Find movies made by Hanson after 1997  
 $\sigma_{myear>1997 \wedge director='Hanson'}(\text{Movies})$



3. Find all movies and their ratings  
 $\pi_{title, rating}(Movies)$

|        | title           | director | myear | rating |
|--------|-----------------|----------|-------|--------|
| Movies | Fargo           | Coen     | 1996  | 8.2    |
|        | Raising Arizona | Coen     | 1987  | 7.6    |
|        | Spiderman       | Raimi    | 2002  | 7.4    |
|        | Wonder Boys     | Hanson   | 2000  | 7.6    |

$\pi_{title, rating}(Movies)$

| title           | rating |
|-----------------|--------|
| Fargo           | 8.2    |
| Raising Arizona | 7.6    |
| Spiderman       | 7.4    |
| Wonder Boys     | 7.6    |

4. Find all actors and directors  
 $\pi_{actor}(Actors) \cup \pi_{director}(Directors)$

| actor     | ayear |
|-----------|-------|
| Cage      | 1964  |
| Hanks     | 1956  |
| Maguire   | 1975  |
| McDormand | 1957  |

| director | dyear |
|----------|-------|
| Coen     | 1954  |
| Hanson   | 1945  |
| Raimi    | 1959  |

$\pi_{actor}(Actors)$

$\pi_{director}(Directors)$

| actor     |
|-----------|
| Cage      |
| Hanks     |
| Maguire   |
| McDormand |

| director |
|----------|
| Coen     |
| Raimi    |
| Hanson   |

$\pi_{actor}(Actors) \cup \pi_{director}(Directors)$

| actor     |
|-----------|
| Cage      |
| Hanks     |
| Maguire   |
| McDormand |
| Coen      |
| Raimi     |
| Hanson    |

**Union Example:**

Find all actors & directors

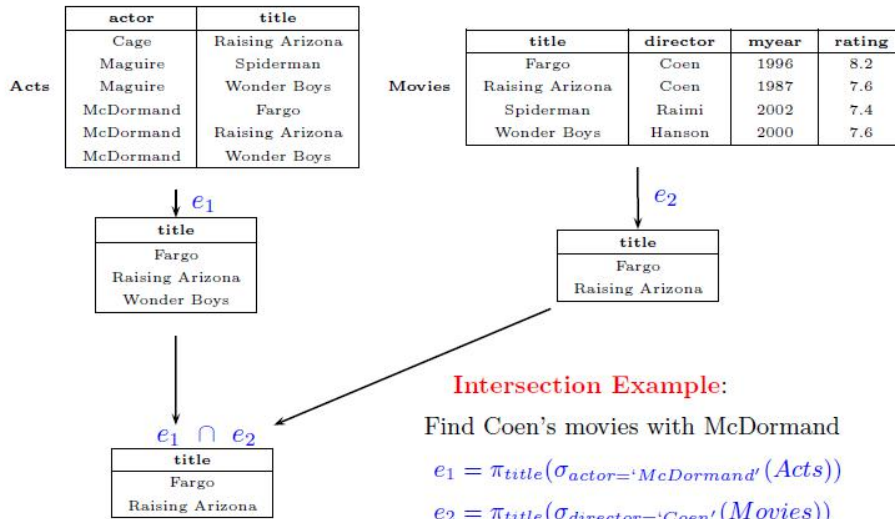
$\pi_{actor}(Actors) \cup \pi_{director}(Directors)$

5. Find Coen's movies with McDormand

$e_1 = \pi_{\text{title}}(\sigma_{\text{actor}='McDormand\_'}(Acts))$

$e_2 = \pi_{\text{title}}(\sigma_{\text{director}='Coen\_'}(Movies))$

**result** =  $e_1 \cap e_2$



### Experiment 3

Consider the Company database with following tables

| EMPLOYEE | FNAME    | MINIT | LNAME    | SSN       | BDATE      | ADDRESS                  | SEX | SALARY | SUPERSSN  | DNO |
|----------|----------|-------|----------|-----------|------------|--------------------------|-----|--------|-----------|-----|
|          | John     | B     | Smith    | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M   | 30000  | 333445555 | 5   |
|          | Franklin | T     | Wong     | 303445555 | 1955-12-08 | 638 Voss, Houston, TX    | M   | 40000  | 888665555 | 5   |
|          | Alicia   | J     | Zelensky | 989687777 | 1968-07-19 | 3321 Castle, Spring, TX  | F   | 25000  | 987654321 | 4   |
|          | Jennifer | S     | Wallace  | 987654321 | 1941-06-20 | 291 Berry, Bellare, TX   | F   | 43000  | 888665555 | 4   |
|          | Ramoth   | K     | Narsyan  | 888884444 | 1982-09-15 | 975 Pine Oak, Humble, TX | M   | 38000  | 333445555 | 5   |
|          | Joyce    | A     | English  | 453453453 | 1972-07-31 | 5831 Rice, Houston, TX   | F   | 25000  | 333445555 | 5   |
|          | Ahmad    | V     | Jabbar   | 987987987 | 1999-03-29 | 960 Dallas, Houston, TX  | M   | 25000  | 987654321 | 4   |
|          | James    | E     | Borg     | 888665555 | 1937-11-10 | 450 Stone, Houston, TX   | M   | 55000  | null      | 1   |

| DEPARTMENT | DNAME          | DNUMBER | MGRSSN    | MGRSTARTDATE |
|------------|----------------|---------|-----------|--------------|
|            | Research       | 5       | 333445555 | 1988-05-22   |
|            | Administration | 4       | 987654321 | 1995-01-01   |
|            | Headquarters   | 1       | 888665555 | 1981-06-19   |

Perform the following:

1. Create company database
2. Viewing all databases
3. Viewing all Tables in a Database,
4. Creating Tables (With and Without Constraints)
5. Inserting/Updating/Deleting Records in a Table
6. Saving (Commit) and Undoing (rollback)

#### SOLUTION:

1. Creating a Database  
CREATE DATABASE Company;
2. Viewing all databases  
SHOW DATABASES;
3. Viewing all Tables in a Database,  
SHOW tables;
4. Creating Tables (With and Without Constraints)  
CREATE TABLE DEPARTMENT  
(DNO VARCHAR2 (20) PRIMARY KEY,  
DNAME VARCHAR2 (20),  
MGRSTARTDATE DATE);

```

CREATE TABLE EMPLOYEE
(SSN VARCHAR2 (20) PRIMARY KEY,
FNAME VARCHAR2 (20),
LNAME VARCHAR2 (20),
ADDRESS VARCHAR2 (20),
SEX CHAR (1),
SALARY INTEGER,
SUPERSSN REFERENCES EMPLOYEE (SSN),
DNO REFERENCES DEPARTMENT (DNO));

```

**NOTE:** Once DEPARTMENT and EMPLOYEE tables are created we must alter department table to add foreign constraint MGRSSN using sql command

```

ALTER TABLE DEPARTMENT
ADD MGRSSN REFERENCES EMPLOYEE (SSN);

```

#### 5. Inserting/Updating/Deleting Records in a Table,

```

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSECE01, 'JOHN', 'SCOTT', 'BANGALORE', 'M', 450000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSCSE01, 'JAMES', 'SMITH', 'BANGALORE', 'M', 500000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSCSE02, 'HEARN', 'BAKER', 'BANGALORE', 'M', 700000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSCSE03, 'EDWARD', 'SCOTT', 'MYSORE', 'M', 500000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSCSE04, 'PAVAN', 'HEGDE', 'MANGALORE', 'M', 650000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSCSE05, 'GIRISH', 'MALYA', 'MYSORE', 'M', 450000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSCSE06, 'NEHA', 'SN', 'BANGALORE', 'F', 800000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSACC01, 'AHANA', 'K', 'MANGALORE', 'F', 350000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSACC02, 'SANTHOSH', 'KUMAR', 'MANGALORE', 'M', 300000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSISE01, 'VEENA', 'M', 'MYSORE', 'M', 600000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES (_RNSIT01, 'NAGESH', 'HR', 'BANGALORE', 'M', 500000);

```

```

INSERT INTO DEPARTMENT VALUES (_1, 'ACCOUNTS', '01-JAN-01', 'RNSACC02');
INSERT INTO DEPARTMENT VALUES (_2, 'IT', '01-AUG-16', 'RNSIT01');

```

```
INSERT INTO DEPARTMENT VALUES (_3', 'ECE', '01-JUN-08', 'RNSECE01');
INSERT INTO DEPARTMENT VALUES (_4', 'ISE', '01-AUG-15', 'RNSISE01');
INSERT INTO DEPARTMENT VALUES (_5', 'CSE', '01-JUN-02', 'RNSCSE05');
```

#### Update

```
UPDATE EMPLOYEE SET DNO='5', SUPERSSN='RNSCSE06' WHERE
SSN='RNSCSE05';
```

Delete entries of employee table where DNO =1;

```
DELETE FROM EMPLOYEE WHERE DNO=1;
```

#### 6. COMMIT and ROLLBACK

Before concluding this section on Data Manipulation Language commands there are two further commands, which are very useful. Changes made to the database by INSERT, UPDATE and DELETE commands are temporary until explicitly committed. This is performed by the command:

##### **COMMIT;**

On execution of this command all changes to the database made by you are made permanent and cannot be undone.

- A COMMIT is automatically executed when you exit normally from SQL\*Plus. However, it does no harm to occasionally issue a COMMIT command.
- A COMMIT does not apply to any SELECT commands as there is nothing to commit.
- A COMMIT does not apply to any DDL commands (eg CREATE TABLE, CREATE INDEX, etc). These are automatically committed and cannot be rolled back.
- If you wished to rollback (ie undo) any changes made to the database since the last commit, you can issue the command:

##### **ROLLBACK;**

A group of related SQL commands that all have to complete successfully or otherwise be rolled back, is called a transaction. Part of your research for Outcome 3 includes investigating transaction processing and the implications of rollback and commit.

**Experiment 4**

Consider Dept table

|               |       |     |
|---------------|-------|-----|
| <u>DEPTNO</u> | DNAME | LOC |
|---------------|-------|-----|

Perform the following:

1. Rename the table dept as department
2. Add a new column PINCODE with not null constraints to the existing table DEPT
3. All constraints and views that reference the column are dropped automatically, along with the column.
4. Rename the column DNAME to DEPT\_NAME in dept table
5. Change the data type of column loc as CHAR with size 10
6. Delete table

**SOLUTION:****Create Table**

```
SQL> CREATE TABLE DEPT(DEPTNO INTEGER, DNAME VARCHAR(10),LOC
VARCHAR(4), PRIMARY KEY(DEPTNO));
```

1. Rename the table dept as department

```
SQL> ALTER TABLE DEPT RENAME TO DEPARTMENT;
Table altered.
```

2. Add a new column PINCODE with not null constraints to the existing table DEPT

```
SQL> ALTER TABLE DEPARTMENT ADD(PINCODE NUMBER(6) NOT NULL);
```

Table altered.

```
SQL> DESC DEPARTMENT;
```

| Name    | Null?    | Type         |
|---------|----------|--------------|
| DEPTNO  | NOT NULL | NUMBER(38)   |
| DNAME   |          | VARCHAR2(10) |
| LOC     |          | VARCHAR2(4)  |
| PINCODE | NOT NULL | NUMBER(6)    |

- All constraints and views that reference the column are dropped automatically, along with the column.

```
SQL> ALTER TABLE DEPARTMENT DROP column LOC CASCADE
CONSTRAINTS;
```

Table altered.

```
SQL> desc dept
```

| Name    | Null?    | Type         |
|---------|----------|--------------|
| DEPTNO  | NOT NULL | NUMBER(38)   |
| DNAME   |          | VARCHAR2(10) |
| PINCODE | NOT NULL | NUMBER(6)    |

- Rename the column DNAME to DEPT\_NAME in dept table

```
SQL> ALTER TABLE DEPT RENAME COLUMN DNAME TO DEPT_NAME ;
```

Table altered.

```
SQL> DESC DEPARTMENT;
```

| Name      | Null?    | Type         |
|-----------|----------|--------------|
| DEPTNO    | NOT NULL | NUMBER(38)   |
| DEPT_NAME |          | VARCHAR2(10) |
| LOC       |          | VARCHAR2(4)  |
| PINCODE   | NOT NULL | NUMBER(6)    |

- Change the datatype of column loc as CHAR with size 10

```
SQL> ALTER TABLE DEPARTMENT MODIFY LOC CHAR(10) ;
```

Table altered.

```
SQL> DESC DEPARTMENT;
```

| Name      | Null?    | Type         |
|-----------|----------|--------------|
| DEPTNO    | NOT NULL | NUMBER(38)   |
| DEPT_NAME |          | VARCHAR2(10) |
| LOC       |          | CHAR(10)     |
| PINCODE   | NOT NULL | NUMBER(6)    |

- Delete table

```
SQL> DROP TABLE DEPARTMENT;
```

Table dropped.



**Experiment 5A**

Consider Employee table

| EMPNO | EMP_NAME | DEPT       | SALARY | DOJ       | BRANCH    |
|-------|----------|------------|--------|-----------|-----------|
| E101  | Amit     | Production | 45000  | 12-Mar-00 | Bangalore |
| E102  | Amit     | HR         | 70000  | 03-Jul-02 | Bangalore |
| E103  | sunita   | Management | 120000 | 11-Jan-01 | mysore    |
| E105  | sunita   | IT         | 67000  | 01-Aug-01 | mysore    |
| E106  | mahesh   | Civil      | 145000 | 20-Sep-03 | Mumbai    |

Perform the following

1. Display all the fields of employee table
2. Retrieve employee number and their salary
3. Retrieve average salary of all employee
4. Retrieve number of employee
5. Retrieve distinct number of employee
6. Retrieve total salary of employee group by employee name and count similar names
7. Retrieve total salary of employee which is greater than >120000
8. Display name of employee in descending order
9. Display details of employee whose name is AMIT and salary greater than 50000;

**1. Display all the fields of employee table**

SQL> select \* from employee;

| EMPNO | EMP_NAME | DEPT       | SALARY | DOJ       | BRANCH    |
|-------|----------|------------|--------|-----------|-----------|
| E101  | Amit     | Production | 45000  | 12-MAR-00 | Bangalore |
| E102  | Amit     | HR         | 70000  | 03-JUL-02 | Bangalore |
| E103  | sunita   | Management | 120000 | 11-JAN-01 | mysore    |
| E105  | sunita   | IT         | 67000  | 01-AUG-01 | mysore    |
| E106  | mahesh   | Civil      | 145000 | 20-SEP-03 | Mumbai    |

**2. Retrieve employee number and their salary**

SQL> select empno, salary from employee;

EMPNO SALARY

```
-----
E101  45000
E102  70000
E103 120000
E105  67000
E106 145000
```

**3. Retrieve average salary of all employee**

```
SQL> select avg(salary) from employee;
```

```
AVG(SALARY)
```

```
-----  
89400
```

**4. Retrieve number of employee**

```
SQL> select count(*) from employee;
```

```
COUNT(*)
```

```
-----  
5
```

**5. Retrieve distinct number of employee**

```
SQL> select count(DISTINCT emp_name) from employee;
```

```
COUNT(DISTINCTEMP_NAME)
```

```
-----  
3
```

**6. Retrieve total salary of employee group by employee name and count similar names**

```
SQL> SELECT EMP_NAME, SUM(SALARY),COUNT(*) FROM EMPLOYEE  
2 GROUP BY(EMP_NAME);
```

| EMP_NAME | SUM(SALARY) | COUNT(*) |
|----------|-------------|----------|
| -----    | -----       | -----    |
| mahesh   | 145000      | 1        |
| sunita   | 187000      | 2        |
| Amit     | 115000      | 2        |

**7. Retrieve total salary of employee which is greater than >120000**

```
SQL> SELECT EMP_NAME, SUM(SALARY) FROM EMPLOYEE  
2 GROUP BY(EMP_NAME)  
3 HAVING SUM(SALARY)>120000;
```

| EMP_NAME | SUM(SALARY) |
|----------|-------------|
| -----    | -----       |
| mahesh   | 145000      |
| sunita   | 187000      |

**8. Display name of employee in descending order**

```
SQL> select emp_name from employee  
2 order by emp_name desc;
```

EMP\_NAME

-----

sunita  
sunita  
mahesh  
Amit  
Amit

**9. Display details of employee whose name is AMIT and salary greater than 50000;**

```
SQL> select * from employee  
2 where emp_name='Amit' and salary>50000;
```

| EMPNO | EMP_NAME | DEPT | SALARY | DOJ       | BRANCH    |
|-------|----------|------|--------|-----------|-----------|
| E102  | Amit     | HR   | 70000  | 03-JUL-02 | Bangalore |

## Experiment 5B

For a given tables

| EMPLOYEE | FNAME | MINIT | LNAME   | SSN       | BDATE      | ADDRESS                  | SEX | SALARY | SUPERSSN  | DNO |
|----------|-------|-------|---------|-----------|------------|--------------------------|-----|--------|-----------|-----|
| John     | B     |       | Smith   | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M   | 30000  | 333445555 | 5   |
| Franklin | T     |       | Wong    | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M   | 40000  | 888665555 | 5   |
| Alicia   | J     |       | Zelens  | 999887777 | 1966-07-19 | 3321 Castle, Spring, TX  | F   | 25000  | 987654321 | 4   |
| Jennifer | S     |       | Wallace | 987054321 | 1941-06-20 | 291 Berry, Belaire, TX   | F   | 43000  | 888665555 | 4   |
| Ramoth   | K     |       | Norsyan | 888884444 | 1982-09-15 | 975 Pine Oak, Humble, TX | M   | 39000  | 333445555 | 5   |
| Joyce    | A     |       | English | 453453453 | 1972-07-31 | 5631 Rice, Houston, TX   | F   | 25000  | 333445555 | 5   |
| Ahmad    | V     |       | Jabbar  | 987987987 | 1969-03-29 | 960 Dallas, Houston, TX  | M   | 25000  | 987654321 | 4   |
| James    | E     |       | Borg    | 888665555 | 1937-11-10 | 400 Stone, Houston, TX   | M   | 55000  | null      | 1   |

| DEPARTMENT | DNAME          | DNUMBER | MGRSSN    | MGRSTARTDATE |
|------------|----------------|---------|-----------|--------------|
|            | Research       | 5       | 333445555 | 1966-05-22   |
|            | Administration | 4       | 987654321 | 1965-01-01   |
|            | Headquarters   | 1       | 888665555 | 1981-05-19   |

Create tables and perform the following

1. How the resulting salaries if every employee working on the 'Research' Departments is given a 10 percent raise.
2. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department
3. Retrieve the name of each employee Controlled by department number 5 (use EXISTS operator).
4. Retrieve the name of each dept and number of employees working in each department which has at least 2 employees
5. Retrieve the name of employees who born in the year 1990's
6. Retrieve the name of employees and their dept name (using JOIN)

**SOLUTION**

```
SQL> CREATE TABLE DEPARTMENT(
  DNO VARCHAR2 (20) PRIMARY KEY,
  DNAME VARCHAR2 (20),
  MGRSTARTDATE DATE);
```

```
SQL> DESC DEPARTMENT;
```

| Name         | Null?    | Type         |
|--------------|----------|--------------|
| DNO          | NOT NULL | VARCHAR2(20) |
| DNAME        |          | VARCHAR2(20) |
| MGRSTARTDATE |          | DATE         |

```
SQL> CREATE TABLE EMPLOYEE(
  SSN VARCHAR2 (20) PRIMARY KEY,
  FNAME VARCHAR2 (20),
  LNAME VARCHAR2 (20),
  ADDRESS VARCHAR2 (20),
  SEX CHAR (1),
  SALARY INTEGER,
  SUPERSSN REFERENCES EMPLOYEE (SSN),
  DNO REFERENCES DEPARTMENT (DNO));
```

```
SQL> DESC EMPLOYEE;
```

| Name     | Null?    | Type         |
|----------|----------|--------------|
| SSN      | NOT NULL | VARCHAR2(20) |
| FNAME    |          | VARCHAR2(20) |
| LNAME    |          | VARCHAR2(20) |
| ADDRESS  |          | VARCHAR2(20) |
| SEX      |          | CHAR(1)      |
| SALARY   |          | NUMBER(38)   |
| SUPERSSN |          | VARCHAR2(20) |
| DNO      |          | NUMBER(38)   |

```
SQL> ALTER TABLE DEPARTMENT
  2 ADD MGRSSN REFERENCES EMPLOYEE (SSN);
```

Table altered.

```
SQL> DESC DEPARTMENT;
```

| Name  | Null?    | Type         |
|-------|----------|--------------|
| DNO   | NOT NULL | VARCHAR2(20) |
| DNAME |          | VARCHAR2(20) |

MGRSTARTDATE  
MGRSSN

DATE  
VARCHAR2(20)

```

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSECE01','JOHN','SCOTT','BANGALORE','M', 450000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE01','JAMES','SMITH','BANGALORE','M', 500000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE02','HEARN','BAKER','BANGALORE','M', 700000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE03','EDWARD','SCOTT','MYSORE','M', 500000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE04','PAVAN','HEGDE','MANGALORE','M', 650000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE05','GIRISH','MALYA','MYSORE','M', 450000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE06','NEHA','SN','BANGALORE','F', 800000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSACC01','AHANA','K','MANGALORE','F', 350000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSACC02','SANTHOSH','KUMAR','MANGALORE','M', 300000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSISE01','VEENA','M','MYSORE','M', 600000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSIT01','NAGESH','HR','BANGALORE','M', 500000);

```

```

INSERT INTO DEPARTMENT VALUES (1,'ACCOUNTS','01-JAN-01','RNSACC02');
INSERT INTO DEPARTMENT VALUES (2,'IT','01-AUG-16','RNSIT01');
INSERT INTO DEPARTMENT VALUES (3,'ECE','01-JUN-08','RNSECE01');
INSERT INTO DEPARTMENT VALUES (4,'ISE','01-AUG-15','RNSISE01');
INSERT INTO DEPARTMENT VALUES (5,'CSE','01-JUN-02','RNSCSE05');

```

**Note: update entries of employee table to fill missing fields SUPERSSN and DNO**

```

UPDATE EMPLOYEE SET SUPERSSN=NULL, DNO='3' WHERE
SSN='RNSECE01';
UPDATE EMPLOYEE SET SUPERSSN='RNSCSE02', DNO='5' WHERE
SSN='RNSCSE01';
UPDATE EMPLOYEE SET SUPERSSN='RNSCSE03', DNO='5' WHERE
SSN='RNSCSE02';
UPDATE EMPLOYEE SET SUPERSSN='RNSCSE04', DNO='5' WHERE
SSN='RNSCSE03';

```

```

UPDATE EMPLOYEE SET DNO='5', SUPERSSN='RNSCSE05' WHERE
SSN='RNSCSE04'; UPDATE EMPLOYEE SET DNO='5', SUPERSSN='RNSCSE06'
WHERE SSN='RNSCSE05';
UPDATE EMPLOYEE SET DNO='5', SUPERSSN=NULL WHERE
SSN='RNSCSE06';
UPDATE EMPLOYEE SET DNO='1', SUPERSSN='RNSACC02' WHERE
SSN='RNSACC01';
UPDATE EMPLOYEE SET DNO='1', SUPERSSN=NULL WHERE
SSN='RNSACC02';
UPDATE EMPLOYEE SET DNO='4', SUPERSSN=NULL WHERE
SSN='RNSISE01';
UPDATE EMPLOYEE SET DNO='2', SUPERSSN=NULL WHERE
SSN='RNSIT01';

```

1. How the resulting salaries if every employee working on the 'Research' Departments is given a 10 percent raise.

```

SQL> SELECT E.FNAME,E.LNAME, 1.1*E.SALARY AS INCR_SAL
2 FROM EMPLOYEE1 E,DEPARTMENT D,EMPLOYEE1 W
3 WHERE E.SSN=W.SSN
4 AND E.DNO=D.DNUMBER
5 AND D.DNAME='research';

```

| FNAME    | LNAME   | SALARY | DNO | DNUMBER | INC_SAL |
|----------|---------|--------|-----|---------|---------|
| john     | smith   | 30000  | 5   | 5       | 33000   |
| franklin | wong    | 40000  | 5   | 5       | 44000   |
| ramesh   | narayan | 780000 | 5   | 5       | 858000  |
| joyce    | english | 25000  | 5   | 5       | 27500   |

2. Find the sum of the salaries of all employees of the 'Accounts' department, as well as the maximum salary, the minimum salary, and the average salary in this department

```

SQL> SELECT SUM(E.SALARY),MAX(E.SALARY),MIN(E.SALARY),
AVG(E.SALARY)FROM EMPLOYEE1 E,DEPARTMENT D WHERE
E.DNO=D.DNUMBER AND D.DNAME='RESEARCH';

```

| SUM<E.SALARY> | MAX<E.SALARY> | MIN<E.SALARY> | AUG<E.SALARY> |
|---------------|---------------|---------------|---------------|
| 875000        | 780000        | 25000         | 218750        |

3. Retrieve the name of each employee Controlled by department number 5 (use EXISTS operator).

```

SQL> SELECT E.FNAME,E.LNAME
2 FROM EMPLOYEE1 E

```

3 WHERE EXISTS(SELECT DNO FROM EMPLOYEE1 WHERE E.DNO=5);

| FNAME    | LNAME   |
|----------|---------|
| john     | smith   |
| franklin | wong    |
| ramesh   | narayan |
| joyce    | english |

4. Retrieve the name of each dept and number of employees working in each department which has at least 2 employees

```
SELECT DNAME, COUNT(*)
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.DNO=E.DNO
AND D.DNO IN (SELECT E1.DNO
FROM EMPLOYEE E1
GROUP BY E1.DNO
having count(*)>2 )
ORDER BY DNO;
```

5. Retrieve the name of employees who born in the year 1990's

```
SELECT E.FNAME,E.LNAME,E.BDATE FROM EMPLOYEE1 E WHERE BDATE LIKE '196%';
```

| FNAME | LNAME | BDATE       |
|-------|-------|-------------|
| john  | smith | 1965-jan-09 |

6. Retrieve the name of employees and their dept name (using JOIN)

```
SELECT E.FNAME, E.LNAME, DNAME
FROM EMPLOYEE E NATURAL JOIN DEPARTMENT D ON E.DNO=D.DNO;
```



**Experiment 5C****Perform the String Functions, Date functions and Mathematical functions supported by Oracle**

```
SQL> select ascii('t') from dual;
```

```
ASCII('T')
```

```
-----  
116
```

```
SQL> select ascii('a') from dual;
```

```
ASCII('A')
```

```
-----  
97
```

```
SQL> select ascii('A') from dual;
```

```
ASCII('A')
```

```
-----  
65
```

```
SQL> select ascii('Z') from dual;
```

```
ASCII('Z')
```

```
-----  
90
```

```
SQL> select ascii('z') from dual;
```

```
ASCII('Z')
```

```
-----  
122
```

```
SQL> SELECT UPPER('bldea sb arts and kcp science college') from dual;
```

```
UPPER('BLDEASBARTSANDKCPSCIENCECOLLEG
```

```
-----
```

```
BLDEA SB ARTS AND KCP SCIENCE COLLEGE
```

```
SQL> select LOWER('welcome to dbms lab') from dual;
```

```
LOWER('WELCOMETODBM
```

```
-----
```

```
welcome to dbms lab
```

```
SQL> select LOWER('WELCOME TO DBMSLAB') from dual;
```

```
LOWER('WELCOMETODB  
-----  
welcome to dbmslab
```

```
SQL> SELECT REPLACE('HELLO','H','K') FROM DUAL;
```

```
REPLA  
-----  
KELLO
```

```
SQL> SELECT REPLACE('COMPUTER','C','K') FROM DUAL;
```

```
REPLACE(  
-----  
KOMPUTER
```

```
SQL> SELECT REPLACE('HELLO','L','A') FROM DUAL;
```

```
REPLA  
-----  
HEAAO
```

```
SQL> SELECT TRIM('A' FROM 'ANACONDA') FROM DUAL;
```

```
TRIM('A'  
-----  
--  
NACOND
```

```
SQL> SELECT LTRIM('ANACONDA','A') FROM DUAL;
```

```
LTRIM('A'  
-----  
NACONDA
```

```
SQL> SELECT LTRIM('ANIL','A') FROM DUAL;
```

```
LTR  
---  
NIL
```

```
SQL> SELECT RTRIM('ANITA','A') FROM DUAL;
```

```
RTRI  
----
```

ANIT

SQL> SELECT RTRIM('ANACONDA','A') FROM DUAL;

RTRIM('

-----

ANACOND

SQL> SELECT RTRIM('ANACONDA ','A') FROM DUAL;

RTRIM('ANAC

-----

ANACONDA

## Date Functions

SQL> SELECT CURRENT\_DATE FROM DUAL;

CURRENT\_D

-----

14-AUG-19

SQL> SELECT EXTRACT(YEAR FROM SYSDATE) FROM DUAL;

EXTRACT(YEARFROMSYSDATE)

-----

2019

SQL> SELECT EXTRACT(DAY FROM SYSDATE) FROM DUAL;

EXTRACT(DAYFROMSYSDATE)

-----

14

SQL> SELECT EXTRACT(MONTH FROM SYSDATE) FROM DUAL;

EXTRACT(MONTHFROMSYSDATE)

-----

8

SQL> SELECT SYSDATE FROM DUAL;

SYSDATE

-----

14-AUG-19

## Mathematical Functions

```
SQL> select ABS(-100) from dual;
ABS(-100)
```

```
-----
      100
```

```
SQL> select ABS(-6) from dual;
```

```
ABS(-6)
```

```
-----
      6
```

```
SQL> select FLOOR(2345.78) FROM DUAL;
FLOOR(2345.78)
```

```
-----
     2345
```

```
SQL> SELECT GREATEST(23,67,90,123,78,50) FROM DUAL;
GREATEST(23,67,90,123,78,50)
```

```
-----
      123
```

```
SQL> SELECT LEAST(34, 21,67,11,89,9) FROM DUAL;
```

```
LEAST(34,21,67,11,89,9)
```

```
-----
      9
```

```
SQL> SELECT LENGTH('RAJESHWARI') FROM DUAL;
LENGTH('RAJESHWARI')
```

```
-----
     10
```

```
SQL> SELECT LENGTH(17245637) FROM DUAL;
LENGTH(17245637)
```

```
-----
      8
```

```
SQL> SELECT SQRT(16) FROM DUAL;
SQRT(16)
```

```
-----
      4
```

```
SQL> SELECT SQRT(99) FROM DUAL;
SQRT(99)
```

```
-----
9.94987437
```

```
SQL> SELECT POWER(2,4) FROM DUAL;  
POWER(2,4)
```

```
-----  
16
```

```
SQL> SELECT POWER(2,10) FROM DUAL;  
POWER(2,10)
```

```
-----  
1024
```

```
SQL> SELECT power(2,10) FROM DUAL;  
POWER(2,10)
```

```
-----  
1024
```

```
SQL> SELECT ROUND(5.86) FROM DUAL;
```

```
ROUND(5.86)  
-----  
6
```

```
SQL> SELECT ROUND(1001.6) FROM DUAL;  
ROUND(1001.6)
```

```
-----  
1002
```

```
SQL> SELECT ROUND(1001.3) FROM DUAL;  
ROUND(1001.3)
```

```
-----  
1001
```

```
SQL> SELECT SIN(90) FROM DUAL;  
SIN(90)
```

```
-----  
.893996664
```

```
SQL> SELECT COS(45) FROM DUAL;  
COS(45)
```

```
-----  
.525321989
```

```
SQL> SELECT TAN(30) FROM DUAL;  
TAN(30)
```

```
-----  
-6.4053312
```

```
SQL> SELECT TAN(90) FROM DUAL;  
TAN(90)
```

```
-----  
-1.9952004  
  
SQL> SELECT TAN(180) FROM DUAL;  
TAN(180)  
-----  
1.33869021  
  
SQL> SELECT SIGN(-128) FROM DUAL;  
SIGN(-128)  
-----  
-1  
  
SQL> SELECT SIGN(10) FROM DUAL;  
SIGN(10)  
-----  
1  
  
SQL> SELECT SIGN(0) FROM DUAL;  
SIGN(0)  
-----  
0  
  
SQL> SELECT LN(100) FROM DUAL;  
LN(100)  
-----  
4.60517019  
  
SQL> SELECT LN(10) FROM DUAL;  
LN(10)  
-----  
2.30258509  
  
SQL> SELECT LOG(10,100) FROM DUAL;  
LOG(10,100)  
-----  
2  
  
SQL> SELECT LOG(100,10) FROM DUAL;  
LOG(100,10)  
-----  
.5  
  
SQL> SELECT MOD(4,3) FROM DUAL;  
  
MOD(4,3)  
-----  
1
```

```
SQL> SELECT MOD(4,2) FROM DUAL;
```

```
MOD(4,2)
-----
0
```

```
SQL> SELECT EXP(2) FROM DUAL;
```

```
EXP(2)
-----
7.3890561
```

```
SQL> SELECT EXP(-2) FROM DUAL;
```

```
EXP(-2)
-----
.135335283
```

```
SQL> SELECT EXP(0) FROM DUAL;
```

```
EXP(0)
-----
1
```

## Experiment 6

For a given EMPLOYEE tables

| EMPLOYEE | FNAME | MINIT | LNAME    | SSN       | BDATE      | ADDRESS                  | SEX | SALARY | SUPERSSN  | DNO |
|----------|-------|-------|----------|-----------|------------|--------------------------|-----|--------|-----------|-----|
| John     | B     |       | Smith    | 123456789 | 1965-01-09 | 731 Fondren, Houston, TX | M   | 30000  | 333445555 | 5   |
| Franklin | T     |       | Wong     | 333445555 | 1955-12-08 | 638 Voss, Houston, TX    | M   | 40000  | 888995555 | 5   |
| Alicia   | J     |       | Zelenski | 999997777 | 1969-07-19 | 3321 Castle, Spring, TX  | F   | 25000  | 987654321 | 4   |
| Jennifer | S     |       | Wallace  | 987654321 | 1941-06-20 | 291 Berry, Belaire, TX   | F   | 43000  | 888995555 | 4   |
| Ramesh   | K     |       | Narsyan  | 666884444 | 1982-09-15 | 975 Fire Oak, Humble, TX | M   | 39000  | 333445555 | 5   |
| Joyce    | A     |       | English  | 453453453 | 1972-07-31 | 5831 Rice, Houston, TX   | F   | 25000  | 333445555 | 5   |
| Ahmad    | V     |       | Jabbar   | 987987987 | 1959-03-29 | 960 Dallas, Houston, TX  | M   | 25000  | 987654321 | 4   |
| James    | E     |       | Borg     | 888995555 | 1937-11-10 | 450 Stone, Houston, TX   | M   | 55000  | null      | 1   |

Perform the Following

1. Creating Views (With and Without Check Option),
2. Selecting from a View
3. Dropping Views,

**SOLUTION:**

```
SQL> CREATE TABLE EMPLOYEE (
  SSN VARCHAR2 (20) PRIMARY KEY,
  FNAME VARCHAR2 (20),
  LNAME VARCHAR2 (20),
  ADDRESS VARCHAR2 (20),
  SEX CHAR (1),
  SALARY INTEGER,
  SUPERSSN REFERENCES EMPLOYEE (SSN),
  DNO REFERENCES DEPARTMENT (DNO));
```

```
SQL> DESC EMPLOYEE;
```

| Name     | Null?    | Type         |
|----------|----------|--------------|
| SSN      | NOT NULL | VARCHAR2(20) |
| FNAME    |          | VARCHAR2(20) |
| LNAME    |          | VARCHAR2(20) |
| ADDRESS  |          | VARCHAR2(20) |
| SEX      |          | CHAR(1)      |
| SALARY   |          | NUMBER(38)   |
| SUPERSSN |          | VARCHAR2(20) |
| DNO      |          | NUMBER(38)   |

```
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSECE01', 'JOHN', 'SCOTT', 'BANGALORE', 'M', 450000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE01', 'JAMES', 'SMITH', 'BANGALORE', 'M', 500000);
```



```

INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE02','HEARN','BAKER','BANGALORE','M', 700000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE03','EDWARD','SCOTT','MYSORE','M', 500000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE04','PAVAN','HEGDE','MANGALORE','M', 650000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE05','GIRISH','MALYA','MYSORE','M', 450000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSCSE06','NEHA','SN','BANGALORE','F', 800000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSACC01','AHANA','K','MANGALORE','F', 350000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSACC02','SANTHOSH','KUMAR','MANGALORE','M', 300000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSISE01','VEENA','M','MYSORE','M', 600000);
INSERT INTO EMPLOYEE (SSN, FNAME, LNAME, ADDRESS, SEX, SALARY)
VALUES ('RNSIT01','NAGESH','HR','BANGALORE','M', 500000);

```

### Creating View

The query that defines the sales\_staffview references only rows in department 5. Furthermore, the CHECK OPTION creates the view with the constraint (named sales\_staff\_cnst) that INSERT and UPDATE statements issued against the view cannot result in rows that the view cannot select.

#### 1. Creating Views (With and Without Check Option)

```

SQL> CREATE VIEW sales_staff AS
2   SELECT fname, ssn, dno
3   FROM employee
4   WHERE dno =5
5   WITH CHECK OPTION CONSTRAINT sales_staff_cnst;

```

View created.

#### 2. Selecting from a View

```
SQL> select * from sales_staff;
```

#### 3. Drop View

```
SQL>DROP VIEW sales_staff;
```

### Experiment 7

Write a PL/SQL program to print integers from 1 to 10 by using PL/SQL FOR loop

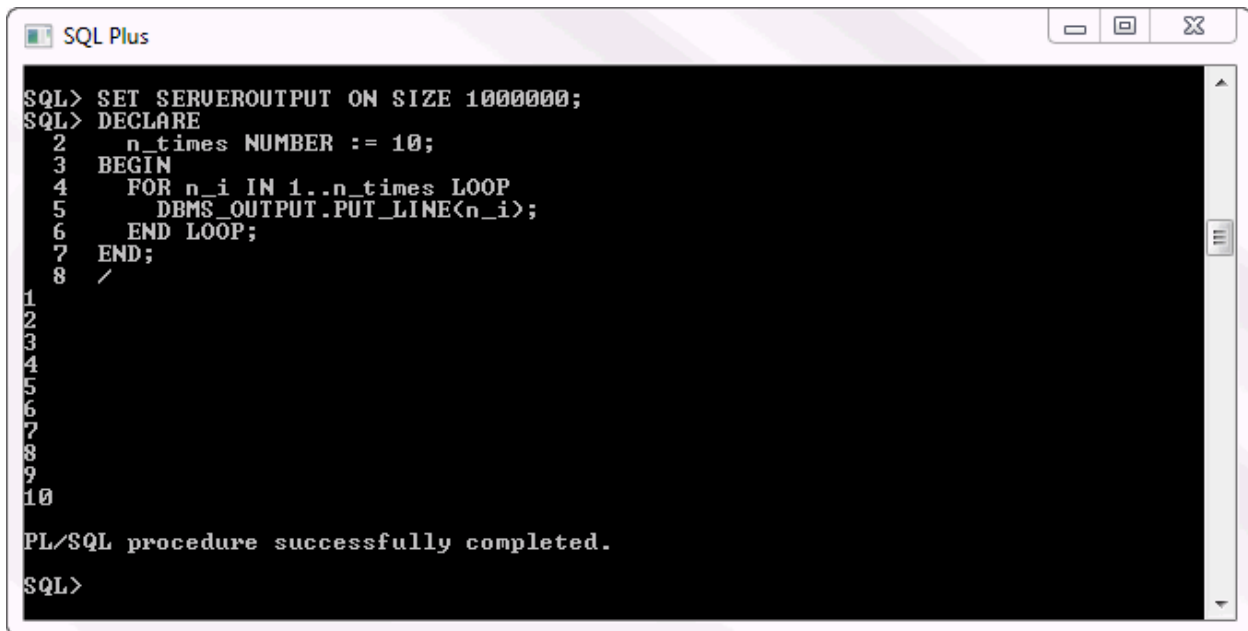
#### SOLUTION:

#### PL/SQL Block

```
SET SERVEROUTPUT ON SIZE 1000000;
DECLARE
  n_times NUMBER := 10;
BEGIN
  FOR n_i IN 1..n_times LOOP
    DBMS_OUTPUT.PUT_LINE(n_i);
  END LOOP;
END;
```

/

#### Output Table



```
SQL Plus
SQL> SET SERVEROUTPUT ON SIZE 1000000;
SQL> DECLARE
  2   n_times NUMBER := 10;
  3 BEGIN
  4   FOR n_i IN 1..n_times LOOP
  5     DBMS_OUTPUT.PUT_LINE(n_i);
  6   END LOOP;
  7 END;
  8 /
1
2
3
4
5
6
7
8
9
10
PL/SQL procedure successfully completed.
SQL>
```

**Experiment 8**

Given the table EMPLOYEE (EmpNo, Name, Salary, Designation, DeptID) write a cursor to select the five highest paid employees from the table.

EMPLOYEE (EmpNo, Name, Salary, Designation, DeptID)

**SOLUTION:**

```
CREATE TABLE EMPLOYEE
(EMPNO INTEGER PRIMARY KEY,
 NAME VARCHAR(20),
 SALARY NUMBER(7,2),
 DESIGNATION VARCHAR(10),
 DEPTID INTEGER);
```

```
get e:/p8.sql;
1 declare
2 i number:=0;
3 cursor ec is select empno,name,salary from employee order by gross_salary desc;
4 r ec%rowtype;
5 begin
6 open ec;
7 loop
8 exit when i=5;
9 fetch ec into r;
10 dbms_output.put_line(r.emp_no||' '||r.employee_name||' '||r.salary);
11 i:=i+1;
12 end loop;
13 close ec;
14* end;
15 .
```

```
SQL> /
```

```
1 rajesh 31000
```

```
2 paramesh 15000
```

```
3 pushpa 14000
```

```
4 vijaya 14000
```

```
5 keerthi 13000
```

PL/SQL procedure successfully completed.

**Experiment 10**

Given an integer i, write a PL/SQL procedure to insert the tuple (i, 'xxx') into a given relation.

**SOLUTION:**

```
CREATE TABLE T2 (  
    a INTEGER,  
    b CHAR(10));
```

```
CREATE OR REPLACE PROCEDURE addtuple1(x IN NUMBER)  
AS  
BEGIN  
    INSERT INTO T2 VALUES(x, 'xxx');  
END addtuple1;  
.  
run;
```

### Viva Questions

**1. What is SQL?**

Structured Query Language

**2. What is database?**

A database is a logically coherent collection of data with some inherent meaning, representing some aspect of real world and which is designed, built and populated with data for a specific purpose.

**3. What is DBMS?**

It is a collection of programs that enables user to create and maintain a database. In other words it is general-purpose software that provides the users with the processes of defining, constructing and manipulating the database for various applications.

**4. What is a Database system?**

The database and DBMS software together is called as Database system.

**5. Advantages of DBMS?**

Redundancy is controlled.

Unauthorized access is restricted.

Providing multiple user interfaces.

Enforcing integrity constraints.

Providing backup and recovery.

**6. Disadvantage in File Processing System?**

Data redundancy & inconsistency.

Difficult in accessing data.

Data isolation.

Data integrity.

Concurrent access is not possible.

Security Problems.

**7. Describe the three levels of data abstraction?**

There are three levels of abstraction:

Physical level: The lowest level of abstraction describes how data are stored.

Logical level: The next higher level of abstraction, describes what data are stored in database and what relationship among those data.

View level: The highest level of abstraction describes only part of entire database.

### 8. Define the "integrity rules"

There are two Integrity rules.

Entity Integrity: States that —Primary key cannot have NULL value

Referential Integrity: States that —Foreign Key can be either a NULL value or should be Primary Key value of other relation.

### 9. What is extension and intension?

Extension - It is the number of tuples present in a table at any instance. This is time dependent.

Intension - It is a constant value that gives the name, structure of table and the constraints laid on it.

### 10. What is Data Independence?

Data independence means that —the application is independent of the storage structure and access strategy of data. In other words, The ability to modify the schema definition in one level should not affect the schema definition in the next higher level.

Two types of Data Independence:

- Physical Data Independence: Modification in physical level should not affect the logical level.
- Logical Data Independence: Modification in logical level should affect the view level.

NOTE: Logical Data Independence is more difficult to achieve

### 11. What is a view? How it is related to data independence?

A view may be thought of as a virtual table, that is, a table that does not really exist in its own right but is instead derived from one or more underlying base table. In other words, there is no stored file that directly represents the view instead a definition of view is stored in data dictionary. Growth and restructuring of base tables is not reflected in views. Thus the view can insulate users from the effects of restructuring and growth in the database. Hence accounts for logical data independence.

### 12. What is Data Model?

A collection of conceptual tools for describing data, data relationships data semantics and constraints.

**13. What is E-R model?**

This data model is based on real world that consists of basic objects called entities and of relationship among these objects. Entities are described in a database by a set of attributes.

**14. What is Object Oriented model?**

This model is based on collection of objects. An object contains values stored in instance variables within the object. An object also contains bodies of code that operate on the object. These bodies of code are called methods. Objects that contain same types of values and the same methods are grouped together into classes.

**15. What is an Entity?**

It is an 'object' in the real world with an independent existence.

**16. What is an Entity type?**

It is a collection (set) of entities that have same attributes.

**17. What is an Entity set?**

It is a collection of all entities of particular entity type in the database.

**18. What is an Extension of entity type?**

The collections of entities of a particular entity type are grouped together into an entity set.

**19. What is an attribute?**

It is a particular property, which describes the entity.

**20. What is a Relation Schema and a Relation?**

A relation Schema denoted by  $R(A_1, A_2, \dots, A_n)$  is made up of the relation name  $R$  and the list of attributes  $A_i$  that it contains. A relation is defined as a set of tuples. Let  $r$  be the relation which contains set tuples  $(t_1, t_2, t_3, \dots, t_n)$ . Each tuple is an ordered list of  $n$ -values  $t=(v_1, v_2, \dots, v_n)$ .

**21. What is degree of a Relation?**

It is the number of attribute of its relation schema.

**22. What is Relationship?**

It is an association among two or more entities.

**23. What is Relationship set?**

The collection (or set) of similar relationships.

**24. What is Relationship type?**

Relationship type defines a set of associations or a relationship set among a given set of entity types.

**25. What is degree of Relationship type?**

It is the number of entity type participating.

**26. What is DDL (Data Definition Language)?**

A data base schema is specified by a set of definitions expressed by a special language called DDL.

**27. What is VDL (View Definition Language)?**

It specifies user views and their mappings to the conceptual schema.

**28. What is SDL (Storage Definition Language)?**

This language is to specify the internal schema. This language may specify the mapping between two schemas.

**29. What is Data Storage - Definition Language?**

The storage structures and access methods used by database system are specified by a set of definition in a special type of DDL called data storage- definition language.

**30. What is DML (Data Manipulation Language)?**

This language that enable user to access or manipulate data as organized by appropriate data model.

Procedural DML or Low level: DML requires a user to specify what data are needed and how to get those data.

Non-Procedural DML or High level: DML requires a user to specify what data are needed without specifying how to get those data.

**31. What is DML Compiler?**

It translates DML statements in a query language into low-level instruction that the query evaluation engine can understand.

**32. What is Relational Algebra?**

It is a procedural query language. It consists of a set of operations that take one or two relations as input and produce a new relation.

**33. What is Relational Calculus?**



It is an applied predicate calculus specifically tailored for relational databases proposed by E.F. Codd. E.g. of languages based on it are DSL, ALPHA, QUEL.

### 34. What is normalization?

It is a process of analyzing the given relation schemas based on their Functional Dependencies (FDs) and primary key to achieve the properties

- Minimizing redundancy
- Minimizing insertion, deletion and update anomalies.

### 35. What is Functional Dependency?

A Functional dependency is denoted by  $X \rightarrow Y$  between two sets of attributes  $X$  and  $Y$  that are subsets of  $R$  specifies a constraint on the possible tuple that can form a relation state  $r$  of  $R$ . The constraint is for any two tuples  $t_1$  and  $t_2$  in  $r$  if  $t_1[X] = t_2[X]$  then they have  $t_1[Y] = t_2[Y]$ . This means the value of  $X$  component of a tuple uniquely determines the value of component  $Y$ .

### 36. When is a functional dependency $F$ said to be minimal?

- Every dependency in  $F$  has a single attribute for its right hand side.
- We cannot replace any dependency  $X \rightarrow A$  in  $F$  with a dependency  $Y \rightarrow A$  where  $Y$  is a proper subset of  $X$  and still have a set of dependency that is equivalent to  $F$ .
- We cannot remove any dependency from  $F$  and still have set of dependency that is equivalent to  $F$ .

### 37. What is Multivalued dependency?

Multivalued dependency denoted by  $X \twoheadrightarrow Y$  specified on relation schema  $R$ , where  $X$  and  $Y$  are both subsets of  $R$ , specifies the following constraint on any relation  $r$  of  $R$ : if two tuples  $t_1$  and  $t_2$  exist in  $r$  such that  $t_1[X] = t_2[X]$  then  $t_3$  and  $t_4$  should also exist in  $r$  with the following properties

- $t_3[x] = t_4[X] = t_1[X] = t_2[X]$
- $t_3[Y] = t_1[Y]$  and  $t_4[Y] = t_2[Y]$
- $t_3[Z] = t_2[Z]$  and  $t_4[Z] = t_1[Z]$

where  $Z = (R - (X \cup Y))$

### 38. What is Lossless join property?

It guarantees that the spurious tuple generation does not occur with respect to relation schemas after decomposition.

**39. What is 1 NF (Normal Form)?**

The domain of attribute must include only atomic (simple, indivisible) values.

**40. What is Fully Functional dependency?**

It is based on concept of full functional dependency. A functional dependency  $X \rightarrow Y$  is fully functional dependency if removal of any attribute  $A$  from  $X$  means that the dependency does not hold any more.

**41. What is 2NF?**

A relation schema  $R$  is in 2NF if it is in 1NF and every non-prime attribute  $A$  in  $R$  is fully functionally dependent on primary key.

**42. What is 3NF?**

A relation schema  $R$  is in 3NF if it is in 2NF and for every FD  $X \rightarrow A$  either of the following is true

$X$  is a Super-key of  $R$ .

$A$  is a prime attribute of  $R$ .

In other words, if every non prime attribute is non-transitively dependent on primary key.

**43. What is BCNF (Boyce-Codd Normal Form)?**

A relation schema  $R$  is in BCNF if it is in 3NF and satisfies additional constraints that for every FD  $X \rightarrow A$ ,  $X$  must be a candidate key.

**44. What is 4NF?**

A relation schema  $R$  is said to be in 4NF if for every Multivalued dependency  $X \twoheadrightarrow Y$  that holds over  $R$ , one of following is true

$X$  is subset or equal to (or)  $XY = R$ .

$X$  is a super key.

**45. What is 5NF?**

A Relation schema  $R$  is said to be 5NF if for every join dependency  $\{R_1, R_2, \dots, R_n\}$  that holds  $R$ , one the following is true

$R_i = R$  for some  $i$ .

The join dependency is implied by the set of FD, over  $R$  in which the left side is key of  $R$ .