TSEC, Mumbai

Department of Computer Engineering

DATABASE MANAGEMENT SYSTEM

Sample questions for DBMS Practical Exam

SE/COMP/SEMIV

Do the practice for all questions given below

Refer the following links :

For Functions :
https://www.plsqltutorial.com/plsql-function/

For Procedure:
https://www.plsqltutorial.com/plsql-procedure/

Create table with Constraints

```
CREATE TABLE employees (
employee_id NUMBER(5) PRIMARY KEY,
first_name VARCHAR2(50) NOT NULL,
last_name VARCHAR2(50) NOT NULL,
email VARCHAR2(100) UNIQUE,
hire_date DATE DEFAULT SYSDATE,
salary NUMBER(10,2) CHECK (salary > 0),
department_id NUMBER(5) REFERENCES departments(department_id) ON DELETE SET NULL
);
```

In this example, we create a table called "employees" with the following columns:

- employee_id: a numeric column with a maximum of 5 digits, set as the primary key of the table.
- first_name and last_name: string columns that cannot be null.
- email: a string column with a maximum length of 100 characters, with a unique constraint applied to ensure no duplicate emails are stored.
- hire_date: a date column that defaults to the current system date when a new row is inserted.
- salary: a numeric column that must have a value greater than 0.
- department_id: a numeric column that references the department_id column in another table called "departments", with the constraint that if a department is deleted, the corresponding department_id value in the "employees" table should be set to NULL.

```
CREATE TABLE employees9 (
employee_id NUMBER(5) PRIMARY KEY,
first_name VARCHAR2(50) NOT NULL,
last_name VARCHAR2(50) NOT NULL,
hire_date DATE CHECK (hire_date BETWEEN TO_DATE('01-JAN-2000', 'DD-MON-YYYY') AND
TO_DATE('31-DEC-2022', 'DD-MON-YYYY'))
);
```

Date constraints: Date constraints can be used to restrict the values that can be entered into a date column. For example, the following statement creates a table with a hire_date column that must be between January 1, 2000 and December 31, 2022

```
CREATE TABLE employees (
employee_id NUMBER(5) PRIMARY KEY,
first_name VARCHAR2(50) NOT NULL,
last_name VARCHAR2(50) NOT NULL,
salary NUMBER(10,2) CHECK (salary BETWEEN 1000 AND 100000)
);
```

Range constraints: Range constraints can be used to restrict the values that can be entered into a numeric column to a specific range. For example, the following statement creates a table with a salary column that must be between 1000 and 100000

```
CREATE TABLE customers (
customer_id NUMBER(5) PRIMARY KEY,
first_name VARCHAR2(50) NOT NULL,
last_name VARCHAR2(50) NOT NULL,
email VARCHAR2(100) UNIQUE
);
```

Unique constraint: A unique constraint can be used to ensure that no duplicate values are entered into a column. For example, the following statement creates a table with an email column that must be unique

Not null constraint: A NOT NULL constraint can be used to ensure that a column must have a value entered into it when a row is inserted or updated. For example, the following statement creates a table with a phone_number column that cannot be null

creating a table with a constraint that requires a specific column value to start with a certain number

```
CREATE TABLE phone_numbers (
   id INT PRIMARY KEY,
   phone_number VARCHAR(20) CONSTRAINT phone_starts_with CHECK (phone_number LIKE '1%')
);
```

Example of creating a table with a constraint that requires a specific column value to start with a certain string (An)

```
CREATE TABLE example_table (
   id INT PRIMARY KEY,
   name VARCHAR(50) CONSTRAINT name_starts_with CHECK (name LIKE 'An%')
);
```

creating a table with a check constraint for city value to be one of Pune or Mumbai and Cname must be in capital letters

```
CREATE TABLE customers (
    id INT PRIMARY KEY,
    cname VARCHAR(50) CHECK (cname = UPPER(cname)),
    city VARCHAR(50) CHECK (city IN ('Pune', 'Mumbai"))
);
```

```
CREATE TABLE employees (
  employee_id NUMBER(6) PRIMARY KEY,
  first_name VARCHAR2(20),
  last_name VARCHAR2(25),
  email VARCHAR2(25),
  phone_number VARCHAR2(20),
  hire_date DATE,
  job_id VARCHAR2(10),
  salary NUMBER(8,2),
  commission_pct NUMBER(2,2),
  manager_id NUMBER(6),
  department_id NUMBER(4)
);
```

Inserting Date

```
INSERT INTO employees (employee_id, first_name, last_name, email, phone_number, hire_date, job_id,
salary, commission_pct, manager_id, department_id)
VALUES (100, 'Steven', 'King', 'steven.king@example.com', '515.123.4567', TO_DATE('17-JUN-1987',
'DD-MON-YYYY'), 'AD_PRES', 24000, NULL, NULL, 90);
```

Triggers Queries

1. BEFORE INSERT trigger to automatically set a new employee's hire date to the current date:

   ```
   CREATE OR REPLACE TRIGGER set_hire_date
   BEFORE INSERT ON employees
   FOR EACH ROW
   BEGIN
     :NEW.hire_date := SYSDATE;
   END;
   /
   ```

2. Create a trigger that will prevent a record from being inserted into the employees table if the salary is less than 1000.

   ```
   CREATE OR REPLACE TRIGGER check_employee_salary
   BEFORE INSERT ON employees
   FOR EACH ROW
   BEGIN
       IF :NEW.salary < 1000 THEN
           RAISE_APPLICATION_ERROR(-20001, 'Salary cannot be less than 1000');
       END IF;
   END;
   /
   ```

3. Create a trigger that outputs a message to the console, indicating the old and new salaries for the employee.

   ```
   CREATE OR REPLACE TRIGGER update_employee_salary
   BEFORE UPDATE OF salary ON employees
   FOR EACH ROW
   BEGIN
      IF :OLD.salary != :NEW.salary THEN
         DBMS_OUTPUT.PUT_LINE('Employee ' || :OLD.employee_id || ' salary is changing from ' ||
   :OLD.salary || ' to ' || :NEW.salary);
      END IF;
   END;
   /
   ```

Functions

1. Create a function that will calculate the age of a person based on their date of birth.

```
CREATE OR REPLACE FUNCTION calculate_age (dob IN DATE)
RETURN NUMBER
IS
  age NUMBER;
BEGIN
  age := TRUNC(MONTHS_BETWEEN(SYSDATE, dob) / 12);
  RETURN age;
END;
/
```

To call the function, you can use the following syntax:

```
DECLARE
  emp_age NUMBER;
BEGIN
  emp_age := calculate_age(TO_DATE('01-JAN-1980', 'DD-MON-YYYY'));
  DBMS_OUTPUT.PUT_LINE('Employee Age: ' || emp_age);
END;
/
```

2. Create a function that will return the total number of employees in a department based on the department ID.

```
CREATE OR REPLACE FUNCTION get_department_size (dept_id IN NUMBER)
RETURN NUMBER
IS
  emp_count NUMBER;
BEGIN
  SELECT COUNT(*) INTO emp_count
  FROM employees e
  WHERE e.department_id = dept_id;
  RETURN emp_count;
END;
/
```

To call the function, you can use the following syntax:

```
DECLARE
  dept_size NUMBER;
BEGIN
  dept_size := get_department_size(60);
  DBMS_OUTPUT.PUT_LINE('Department Size: ' || dept_size);
END;
/
```

Procedure

1. Create a procedure that will insert a new record into the employees table.

```
CREATE OR REPLACE PROCEDURE add_employee (first_name IN VARCHAR2, last_name IN
VARCHAR2, email IN VARCHAR2, hire_date IN DATE, job_id IN VARCHAR2, salary IN
NUMBER, manager_id IN NUMBER, department_id IN NUMBER)
IS
BEGIN
   INSERT INTO employees (emp_id, first_name, last_name, email, hire_date, job_id, salary,
manager_id, department_id)
   VALUES (emp_id, first_name, last_name, email, hire_date, job_id, salary, manager_id,
department_id);
   COMMIT;
END;
/
```

To call the procedure, you can use the following syntax:

```
BEGIN
   add_employee(1,'John', 'Doe', 'johndoe@email.com', SYSDATE, 'IT_PROG', 5000, 100, 60);
END;
/
```

2. Create a procedure that will update the salary of an employee based on their employee ID.

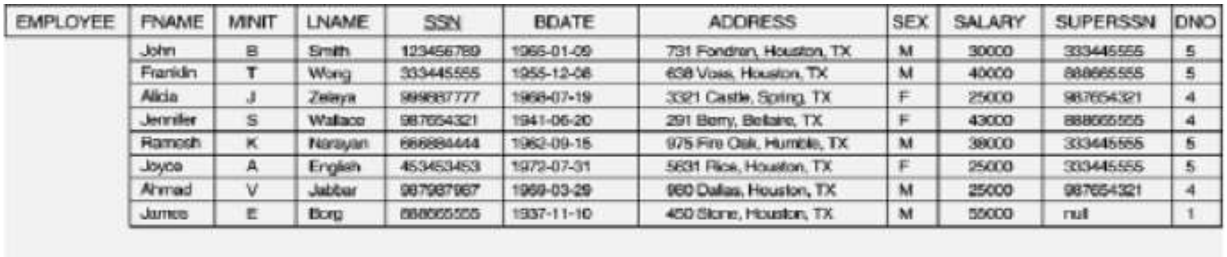
```
CREATE OR REPLACE PROCEDURE update_employee_salary (emp_id IN NUMBER,
new_salary IN NUMBER)
IS
BEGIN
   UPDATE employees SET salary = new_salary WHERE employee_id = emp_id;
   COMMIT;
END;
/
```

To call the procedure, you can use the following syntax:

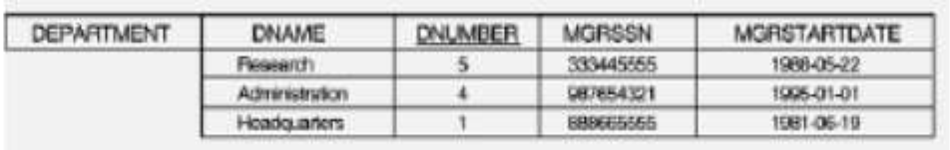
```
BEGIN
   update_employee_salary(100, 6000);
END;
/
```

For Practical

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	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 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.

For Practical

Consider Dept table

DEPTNO	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)
   ```

3. 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)
```

4. 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)
```

5. Change the datatype of colunm 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)
```

6. Delete table

```
SQL> DROP TABLE DEPARTMENT;
Table dropped.
```

For Practical

Consider Employee table

EMPNO	EMP_NAME	DEPT	SALARY	DOJ	BRANCH
E101	Amit	oduction	45000	12-Mar-00	Bangalore
E102	Amit	HR	70000	03-Jul-02	Bangalore
E103	sunita	anagemen	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;
EMPN  EMP_NAME  DEPT          SALARY      DOJ          BRANCH
O
E101   Amit     Production    45000       12-MAR-00    Bangalor
                                                       e
E102  Amit      HR            70000       03-JUL-02    Bangalor
                                                       e
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   12000
       0
E105    67000
```

```
E106   14500
       0
```

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(SALAR  COUNT(*
                       Y)         )
-------------------------------------------------
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(SALA

RY)

mahesh                145000
sunita               187000
```

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

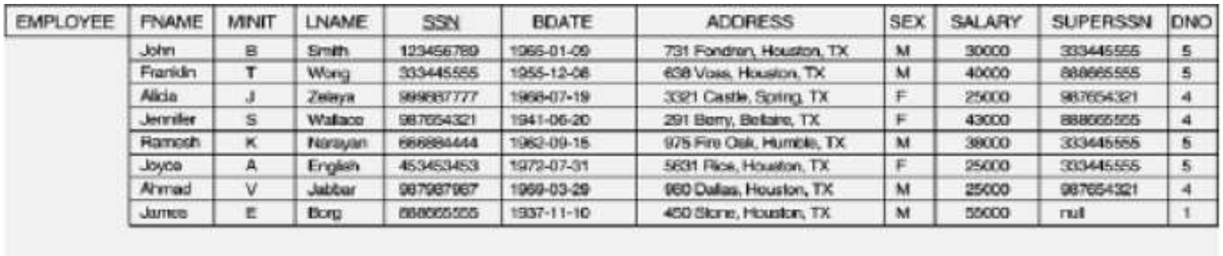
```
SQL> select emp_name from
  employee 2 order by emp_name
  desc;

EMP_NAME
--------------------
sunita
sunita
mahes
h Amit
Amit
```

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

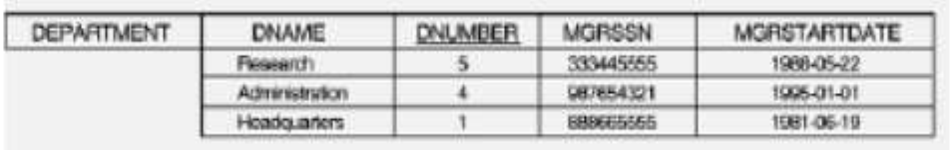
```
SQL> select * from employee
  2 where emp_name='Amit' and salary>50000;

EMPNO         EMP_NAME              DEPT      SALARY      DOJ          BRANCH
E102           Amit                 HR        70000      03-JUL-02     Bangalor
                                                                       e
```

For Practical

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	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 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

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	DATE
MGRSSN	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‘);
```

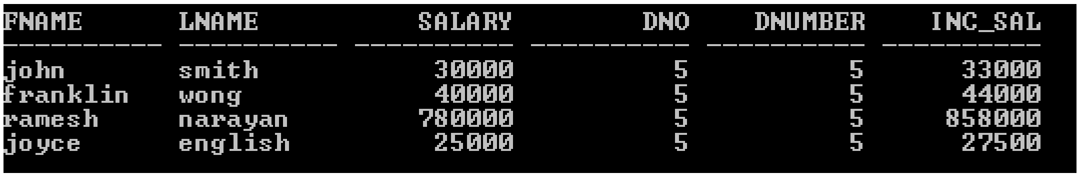
Note: update entries of employee table to fill missing fields SUPERSSN and DNO

```
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.**

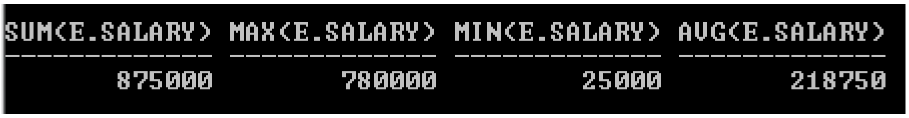
```
SQL> SELECT E.FNAME,E.LNAME, 1.1*E.SALARY AS INCR_SAL
  2 FROM EMPLOYEE1 E,DEPARTMENT D,EMPLOYEE1 W
  3  WHERE E.SSN=W.SSN
  4  AND E.DNO=D.DNUMBER
  5  AND D.DNAME='research';
```



```
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';
```

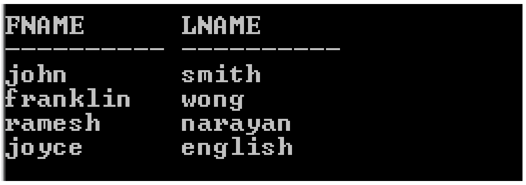


```
SUM(E.SALARY) MAX(E.SALARY) MIN(E.SALARY) AVG(E.SALARY)
------------- ------------- ------------- -------------
       875000        780000         25000        218750
```

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);
```



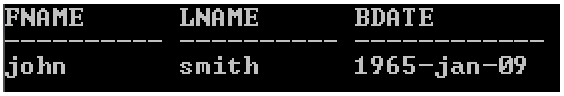
```
FNAME      LNAME
---------- ----------
john       smith
franklin   wong
ramesh     narayan
joyce      english
```

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%';
```



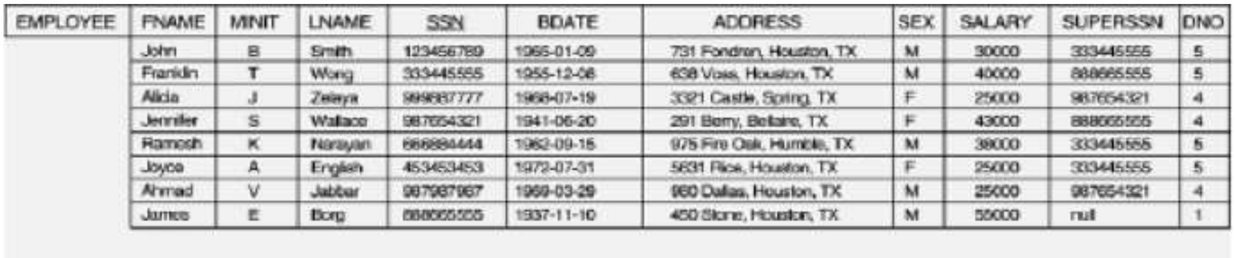
```
FNAME      LNAME      BDATE
---------- ---------- ------------
john       smith      1965-jan-09
```

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;
```

For Practical View

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	888665555	5
	Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
	Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
	Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
	James	E	Borg	888665555	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;
```

For Practical Procedure

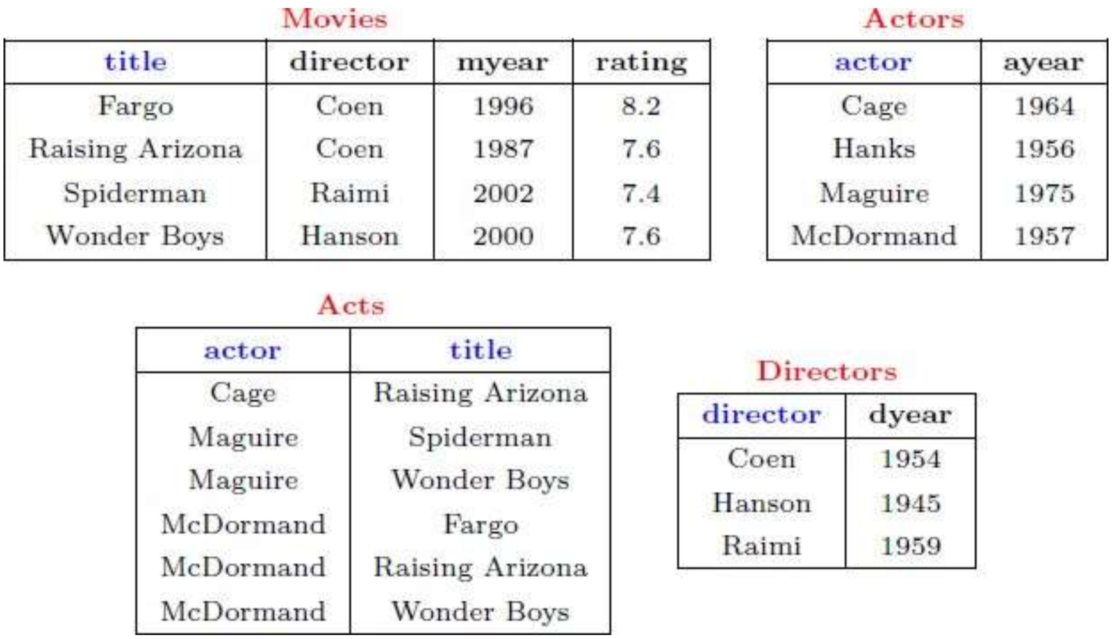
Given an integer i, write a PL/SQL procedure to insert the tuple (i, 'xxx') into a given relation.

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;
```

Study For Theory Only Not in Practical Exam

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

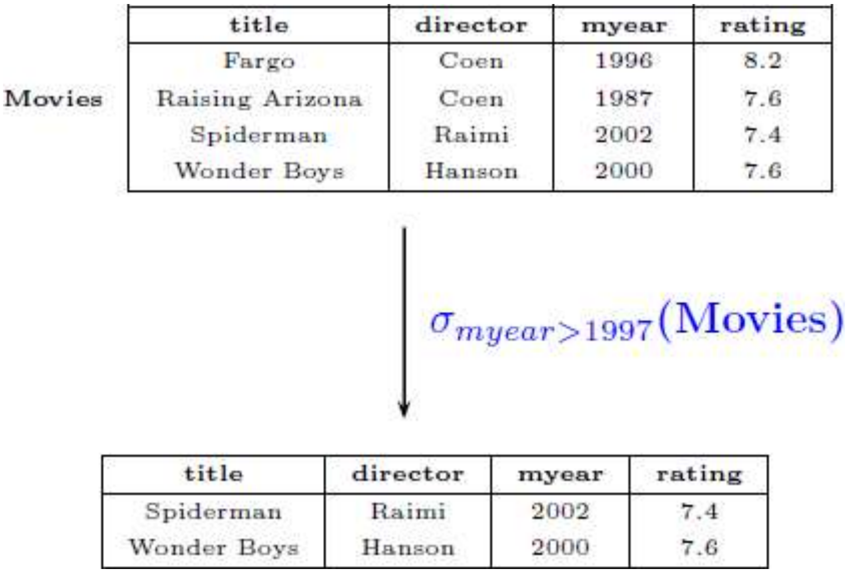
SOLUTION:

Common notations of Relational Algebra

Operation	**Purpose**
Select(σ)	The SELECT operation is used for selecting a subset of the tuples according to a given selection condition
Projection(π)	The projection eliminates all attributes of the input relation but those mentioned in the projection list.
Union Operation(∪)	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(∩)	Intersection defines a relation consisting of a set of all tuple that are in both A and B.
Cartesian Product(X)	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(θ)	The general case of JOIN operation is called a Theta join. It is denoted by symbol θ.
EQUI Join	When a theta join uses only equivalence condition, it becomes a equi join.
Natural Join(⋈)	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(⟕)	In the left outer join, operation allows keeping all tuple in the left relation.

Right Outer join(⟖)	In the right outer join, operation allows keeping all tuple in the right relation.
Full Outer Join(⟗)	In a full outer join, all tuples from both relations are included in the result irrespective of the matching condition.

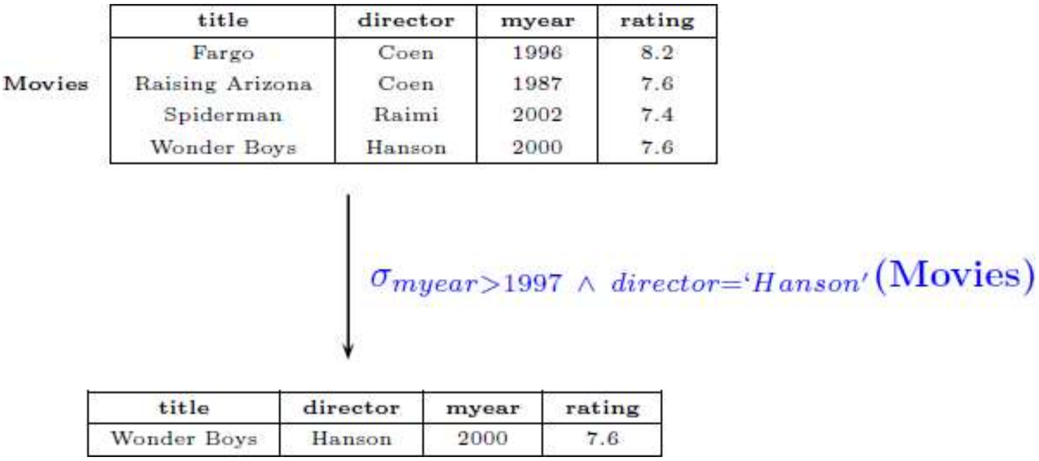
2. Find movies made after 1997
σmyear>1997**(Movies)**



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

title	director	myear	rating
Spiderman	Raimi	2002	7.4
Wonder Boys	Hanson	2000	7.6

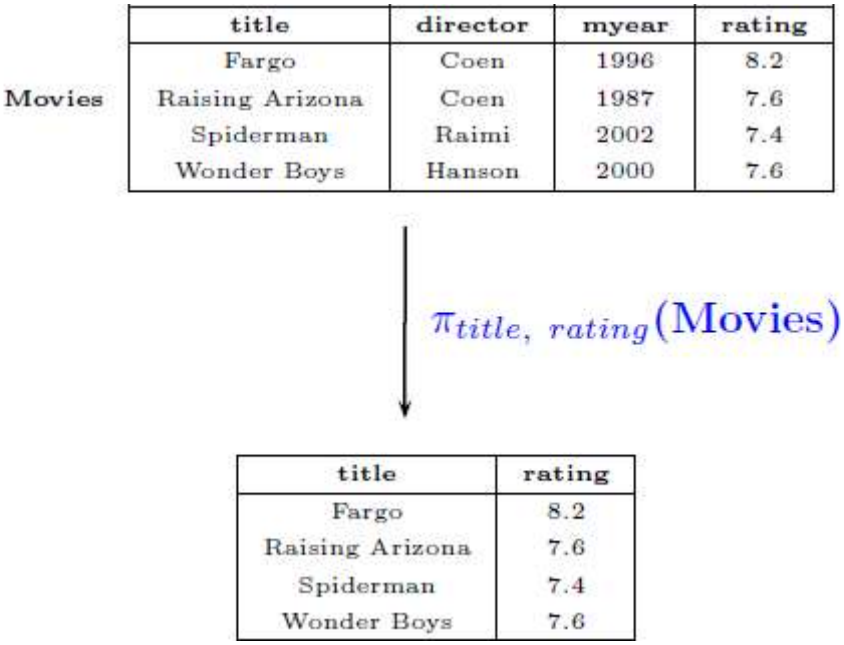
3. Find movies made by Hanson after 1997
σmyear>1997 ∧ director=‘Hanson_**(Movies)**



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

title	director	myear	rating
Wonder Boys	Hanson	2000	7.6

4. Find all movies and their ratings
πtitle, rating**(Movies)**



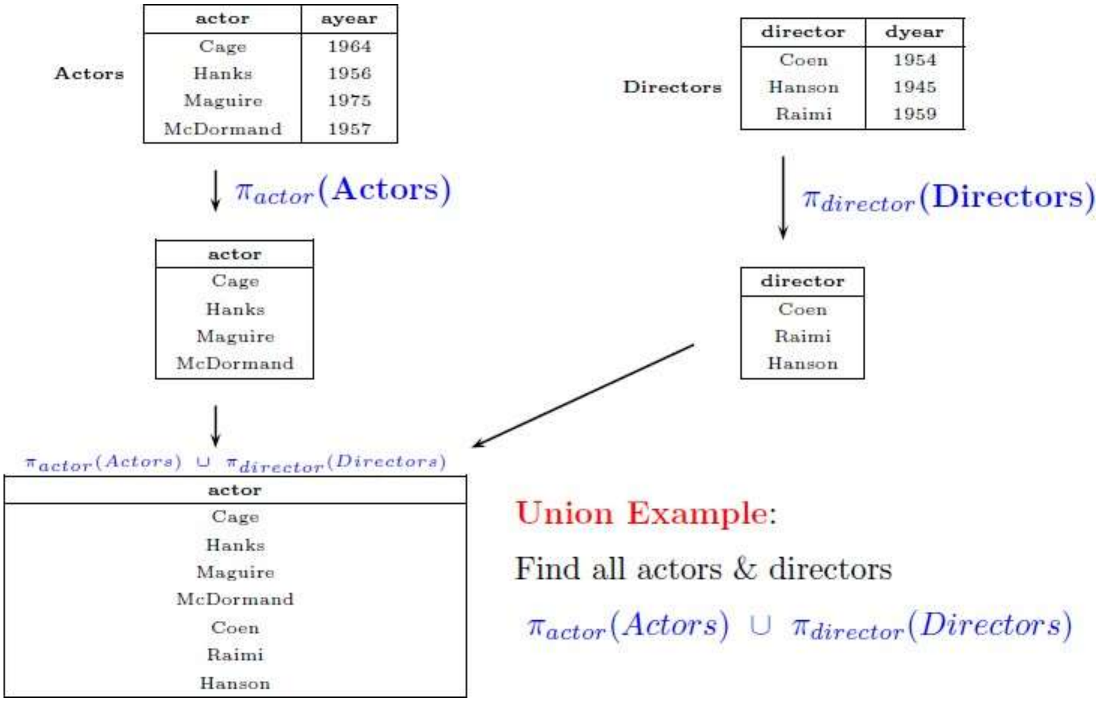
Movies

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

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

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

5. Find all actors and directors
πactor(Actors) ∪ πdirector(Directors)



Actors

actor	ayear
Cage	1964
Hanks	1956
Maguire	1975
McDormand	1957

Directors

director	dyear
Coen	1954
Hanson	1945
Raimi	1959

$\pi_{actor}(\text{Actors})$

actor
Cage
Hanks
Maguire
McDormand

$\pi_{director}(\text{Directors})$

director
Coen
Raimi
Hanson

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

actor
Cage
Hanks
Maguire
McDormand
Coen
Raimi
Hanson

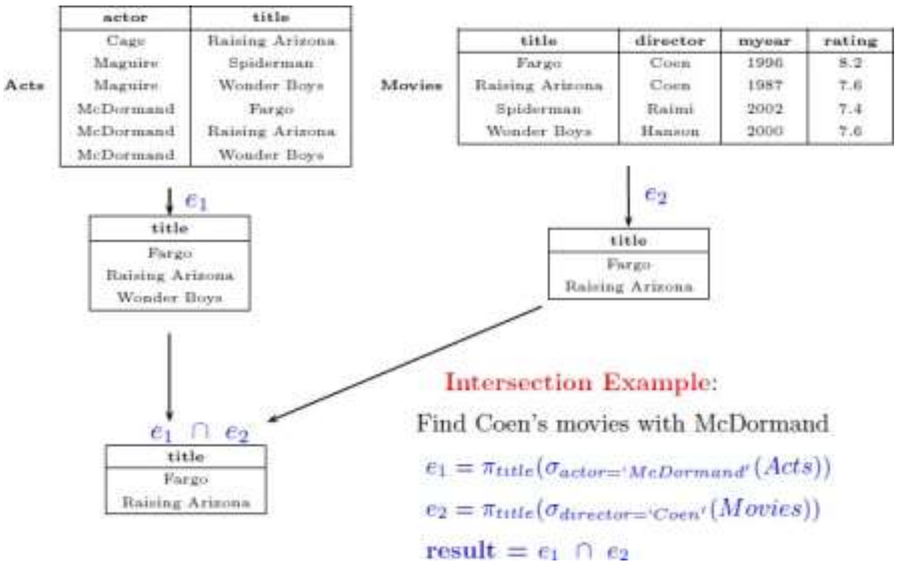
Union Example:

Find all actors & directors

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

6. Find Coen’s movies with McDormand

   e1 = πtitle(σactor=‘McDormand_ (Acts))
   e2 = πtitle(σdirector=‘Coen_ (Movies))
   result = e1 ∩ e2



Acts

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

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

e_1

title
Fargo
Raising Arizona
Wonder Boys

e_2

title
Fargo
Raising Arizona

$e_1 \cap e_2$

title
Fargo
Raising Arizona

Intersection Example:

Find Coen’s movies with McDormand

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

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

result $= e_1 \cap e_2$

For theory Exam Sample 1

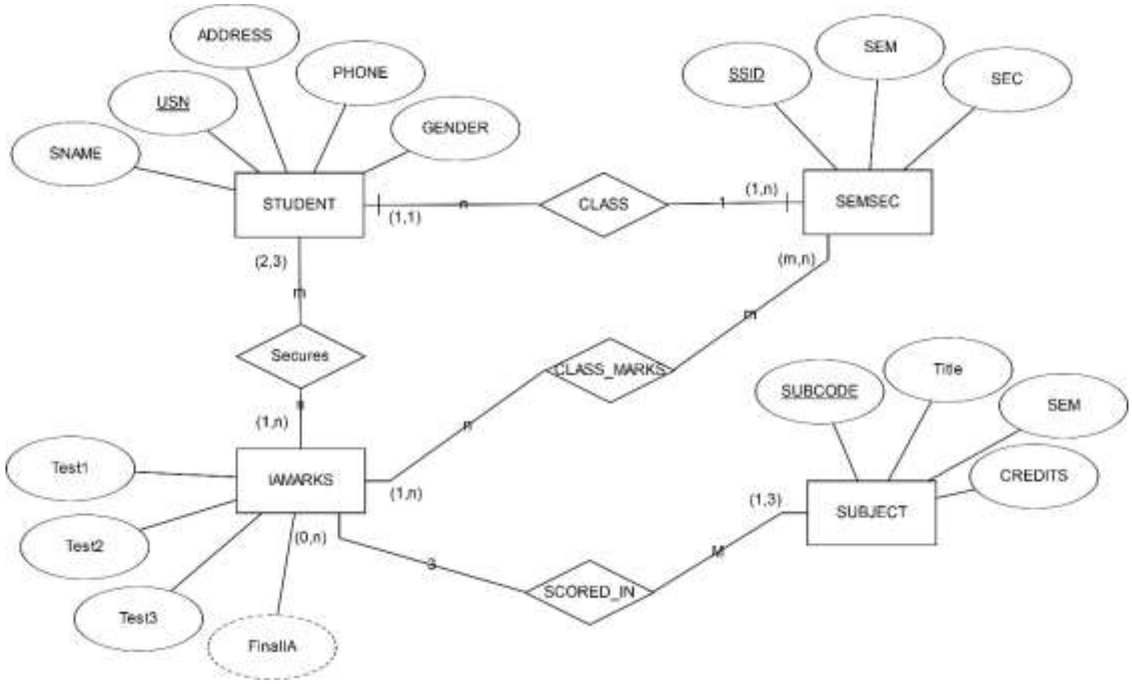
Consider following databases and draw ER diagram and convert entities and relationships to relation table for a given scenario.

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*)

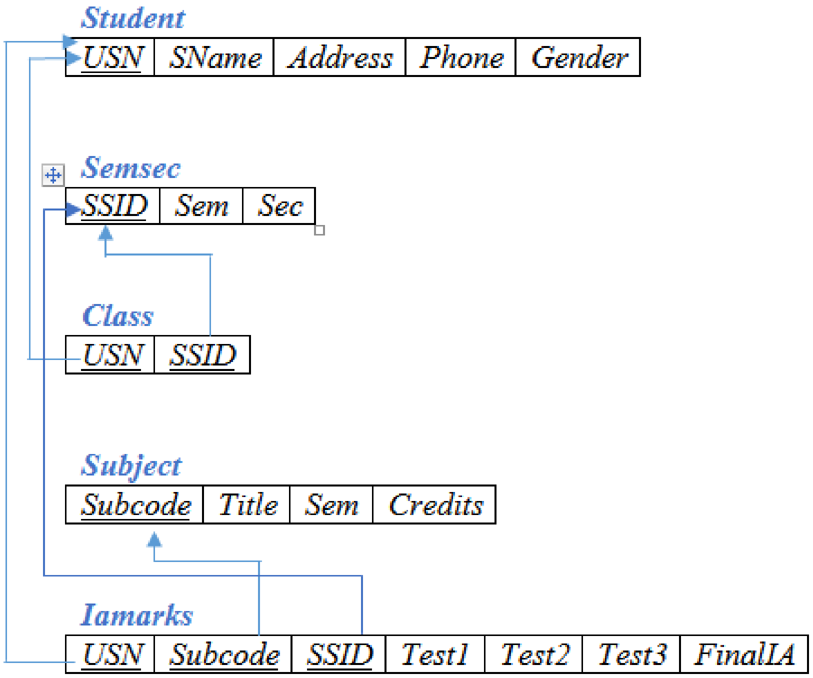
2. **COMPANY DATABASE:**
   EMPLOYEE (*SSN*, *Name*, *Address*, *Sex*, *Salary*, *SuperSSN*, *DNo*)
   DEPARTMENT (*DNo*, *DName*, *MgrSSN*, *MgrStartDate*)
   DLOCATION (*DNo*,*DLoc*)
   PROJECT (*PNo*, *PName*, *PLocation*, *DNo*)
   WORKS_ON (*SSN*, *PNo*, *Hours*)

SOLUTION:

College Database: E-R Diagram



Mapping entities and relationships to relation table (Schema Diagram)



Student

USN	SName	Address	Phone	Gender

Semsec

SSID	Sem	Sec

Class

USN	SSID

Subject

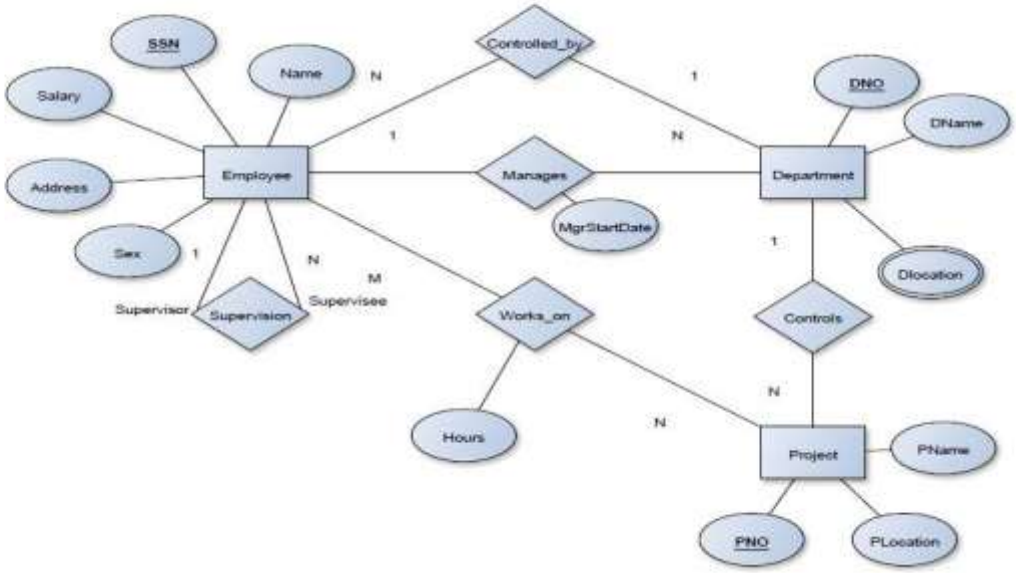
Subcode	Title	Sem	Credits

Iamarks

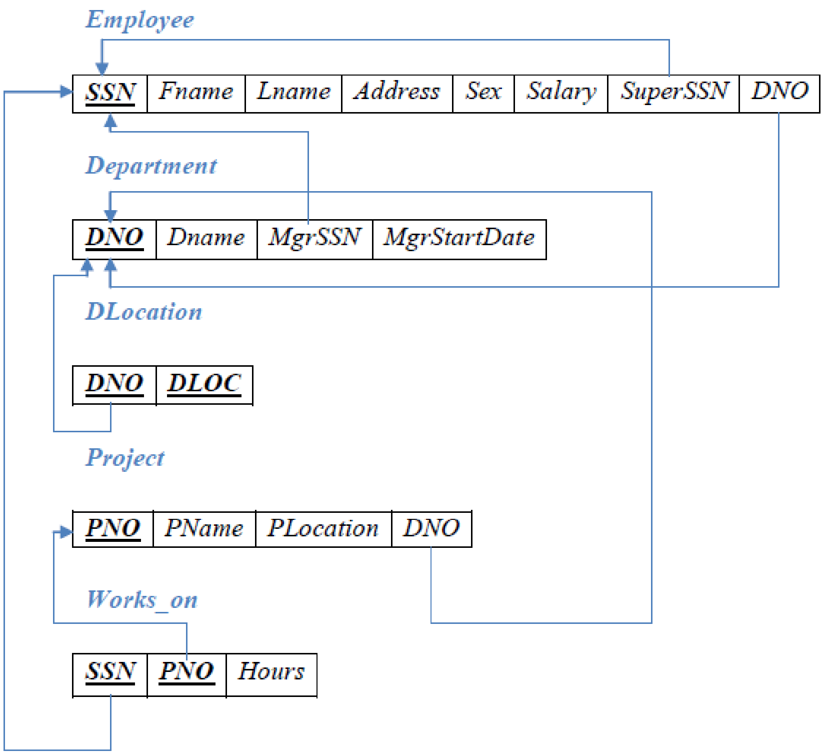
USN	Subcode	SSID	Test1	Test2	Test3	FinalIA

COMPANY DATABASE:

E-R Diagram



Schema Diagram



Employee

SSN	Fname	Lname	Address	Sex	Salary	SuperSSN	DNO

Department

DNO	Dname	MgrSSN	MgrStartDate

DLocation

DNO	DLOC

Project

PNO	PName	PLocation	DNO

Works_on

SSN	PNO	Hours