

**B.M.S.COLLEGE OF ENGINEERING,**

**BANGALORE-19**

**(Autonomous College under VTU)**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**dATABASE MANAGEMENT SYSTEM**

**LABORATORY RECORD**

**NAME: SUPRIYA. M. LINGDAL**

**USN: 1BM20CS415**

**PROGRAM: BACHELOR OF ENGINEERING**

**SEMESTER: IV**

**SESSION: APRIL-JULY 2021**

**COURSE CODE: 19CS4PCDBM**

**COURSE TITLE: DATABASE MANAGEMENt SYSTEM**

**CREDITS: 4**

**DBMS Lab List**

|  |  |
| --- | --- |
| **Experiments** | **Name of Experiment** |
| 1 | Insurance Database |
| 2 | Banking Enterprise Database |
| 3 | Supplier Database |
| 4 | Student Faculty Database |
| 5 | Airline Flight Database |
| 6 | Order Processing Database |
| 7 | Book dealer Database |
| 8 | Student Enrolment Database |
| 9 | Movie Database |
| 10 | College Database |

**PROGRAM 1**

**INSURANCE DATABASE**

Consider the Insurance database given below. The primary keys are underlined and the data types are specified.

PERSON (driver-id #: String, name: String, address: String)

CAR (Regno: String, model: String, year: int)

ACCIDENT (report-number: int, date: date, location: String)

OWNS (driver-id #: String, Regno: String)

PARTICIPATED (driver-id: String, Regno: String, report-number: int, damage-amount: int

**i. Create the above tables by properly specifying the primary keys and the foreign keys**

create database Supriya;

use Supriya;

create table person(

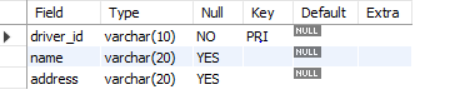
driver\_id varchar(10),

name varchar(20),

address varchar(20),

primary key(driver\_id));

desc person;



create table car

(

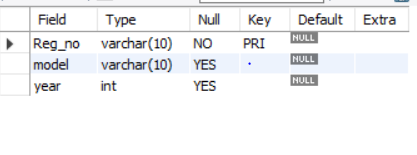
Reg\_no varchar(10),

model varchar(10),

year int(10),

primary key(Reg\_no));

desc car;



create table accident(

Report\_no int(10),

accident\_date date,

location varchar(20),

primary key(Report\_no));

desc accident;

create table owns

(

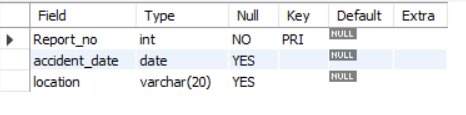
driver\_id varchar(10),

Reg\_no varchar(10),

primary key(driver\_id,Reg\_no),

foreign key(driver\_id)references person(driver\_id),

foreign key(Reg\_no)references car(Reg\_no));



create table participated

(

driver\_id varchar(10),

Reg\_no varchar(10),

report\_num int,

damage\_amount int,

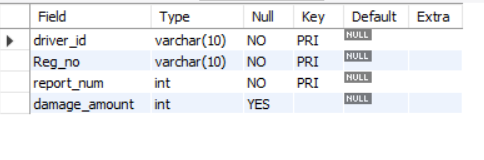
primary key(driver\_id,Reg\_no,report\_num),

foreign key(driver\_id)references person(driver\_id),

foreign key(Reg\_no)references car(Reg\_no),

foreign key(report\_num)references accident(Report\_no));

desc participated;



**ii. Enter at least five tuples for each relation**

use Supriya;

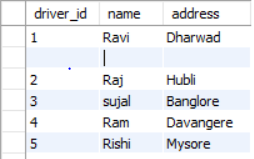
insert into person values(1,'Ravi','Dharwad');

insert into person values(2,'Raj','Hubli');

insert into person values(3,'sujal','Banglore');

insert into person values(4,'Ram','Davangere');

insert into person values(5,'Rishi','Mysore');



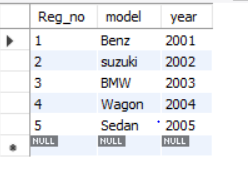
insert into car values(1,'Benz','2001');

insert into car values(2,'suzuki','2002');

insert into car values(3,'BMW','2003');

insert into car values(4,'Wagon','2004');

insert into car values(5,'Sedan','2005');



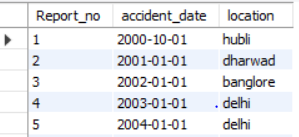
insert into accident values(1,'Ravi','Dharwad');

insert into accident values(2,'Raj','Hubli');

insert into accident values(3,'sujal','Banglore');

insert into accident values(4,'Ram','Davangere');

insert into accident values(5,'Rishi','Mysore');



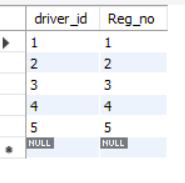
insert into owns values(1,1);

insert into owns values(2,2);

insert into owns values(3,3);

insert into owns values(4,4);

insert into owns values(5,5);

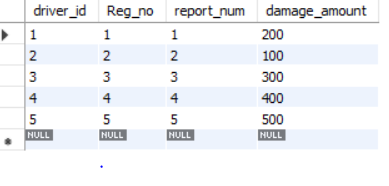


insert into participated values(1,1,1,200);

insert into participated values(2,2,2,100);

insert into participated values(3,3,3,300);

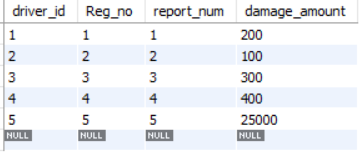
insert into participated values(4,4,4,400);

insert into participated values(5,5,5,500);

**iii) a.Update the damage amount for the car with a specific Reg\_no in the accident with report number 5 to 25000.**

use Supriya;

update participated set damage\_amount=25000 where Reg\_no=5 AND report\_num=5;



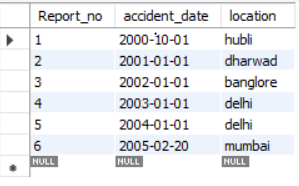
**b. Add a new accident to the database.**

use Supriya;

select \* from accident;

insert into accident values(6,'2005-02-20','mumbai');

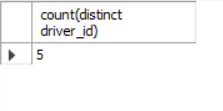
select \* from accident;



**iv) Find the total number of people who owned cars that involved in accidents in 2008.**

use Supriya;

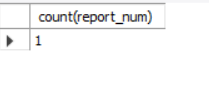
select count(distinct driver\_id)from participated p,accident a where p.report\_num and a.accident\_date like '2000-10-01';



**v) Find the number of accidents in which cars belonging to a specific model were involved**

use Supriya;

select count(report\_num)from car c,participated p where c.Reg\_no=p.Reg\_no and model='Benz';



**PROGRAM 2**

**BANKING ENTERPRISE DATABASE**

**Consider the following database for a banking enterprise.**

**BRANCH (branch-name: String, branch-city: String, assets: real)**

**ACCOUNTS (accno: int, branch-name: String, balance: real)**

**DEPOSITOR (customer-name: String, customer-street: String, customer-city: String)**

**LOAN (loan-number: int, branch-name: String, amount: real)**

**BORROWER (customer-name: String, loan-number: int)**

**i. Create the above tables by properly specifying the primary keys and the foreign keys**

use Supriya;

create table branch

(

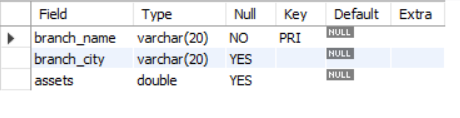
branch\_name varchar(20),

branch\_city varchar(20),

assets real,

primary key(branch\_name));

desc branch;



use Supriya;

create table accounts

(

branch\_name varchar(20),

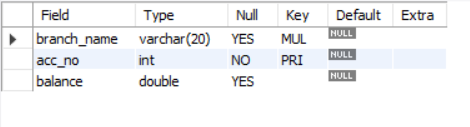
acc\_no integer,

balance real,

primary key(acc\_no),

foreign key(branch\_name)references branch(branch\_name));

desc accounts;



create table depositor

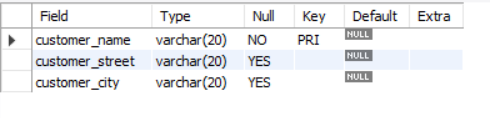
(

customer\_name varchar(20),

customer\_street varchar(20),

customer\_city varchar(20),

primary key(customer\_name));



create table loan

(

loan\_number int,

branch\_name varchar(20),

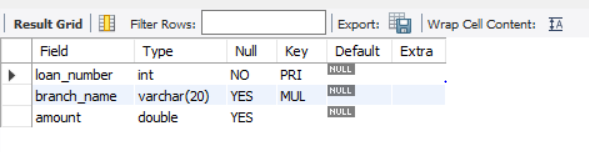
amount real,

primary key(loan\_number),

foreign key(branch\_name)references branch(branch\_name)

);

desc loan;



create table borrower

(

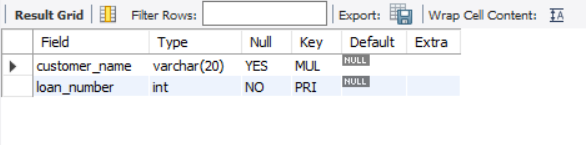
customer\_name varchar(20),

loan\_number int,

primary key(loan\_number),

foreign key(customer\_name)references depositor(customer\_name),

foreign key(loan\_number)references loan(loan\_number));



**ii. Enter at least five tuples for each relation.**

use Supriya;

insert into branch values('SBI','Bangalore','25.6');

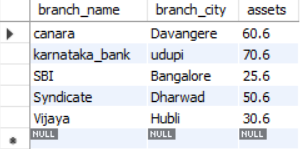
insert into branch values('Vijaya','Hubli','30.6');

insert into branch values('Syndicate','Dharwad','50.6');

insert into branch values('canara','Davangere','60.6');

insert into branch values('karnataka\_bank','udupi','70.6');

select \* from branch;



insert into accounts values('SBI','12345','25000.6');

insert into accounts values('Vijaya','23456','30000.6');

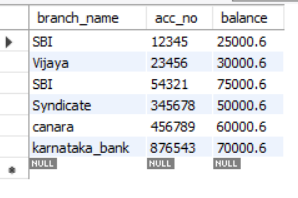
insert into accounts values('Syndicate','345678','50000.6');

insert into accounts values('canara','456789','60000.6');

insert into accounts values('karnataka\_bank','876543','70000.6');

insert into accounts values('SBI','54321','75000.6');

select \* from accounts;



insert into depositor values('Ravi','vidyagiri','Dharwad');

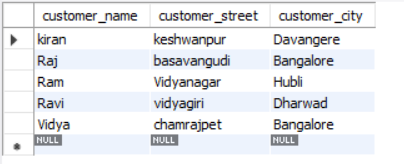
insert into depositor values('Vidya','chamrajpet','Bangalore');

insert into depositor values('Ram','Vidyanagar','Hubli');

insert into depositor values('Raj','basavangudi','Bangalore');

insert into depositor values('kiran','keshwanpur','Davangere');

select \* from depositor;



insert into loan values('1','SBI','25000.6');

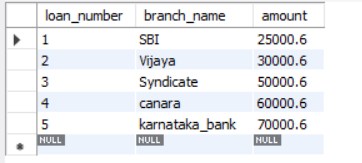
insert into loan values('2','Vijaya','30000.6');

insert into loan values('3','Syndicate','50000.6');

insert into loan values('4','canara','60000.6');

insert into loan values('5','karnataka\_bank','70000.6');

select \* from loan;



insert into borrower values('kiran',1);

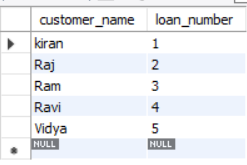
insert into borrower values('Raj',2);

insert into borrower values('Ram',3);

insert into borrower values('Ravi',4);

insert into borrower values('Vidya',5);

select \* from borrower;



**iii. Find all the customers who have at least two accounts at the Main branch.**

select c.customer\_name

from bank\_customer c

where exists(

select d.cust\_name

from depositer d, bank\_account ba

where

d.accno=ba.accno and

c.customer\_name=d.cust\_name and

ba.branch\_name='SBI\_chandralayout'

group by d.cust\_name

having count(d.cust\_name)>=2

);

**iv. Find all the customers who have an account at all the branches located in a specific city.**

select d.cust\_name from depositer d,branch b,bank\_account ba

where b.branch\_name=ba.branch\_name

AND ba.accno=d.accno

and branch\_city='Delhi'

group by d.cust\_name

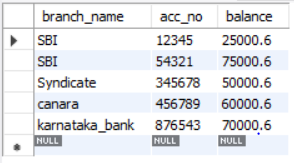
having COUNT(distinct b.branch\_name)=(select COUNT(branch\_name) from branch where branch\_city='Delhi');

**v. Demonstrate how you delete all account tuples at every branch located in a specific city.**

use Supriya;

delete from accounts where branch\_name IN(select branch\_name from branch where branch\_city='Hubli');

select \* from accounts;



**PROGRAM 3**

**SUPPLIER DATABASE**

**Consider the following schema:**

**SUPPLIERS (sid: integer, sname: string, address: string)**

**PARTS (pid: integer, pname: string, color: string)**

**CATALOG (sid: integer, pid: integer, cost: real)**

use Supriya;

create table suppliers

(

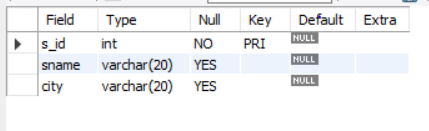
s\_id int(5),

sname varchar(20),

city varchar(20),

primary key (s\_id));

desc suppliers;



use Supriya;

create table parts

(

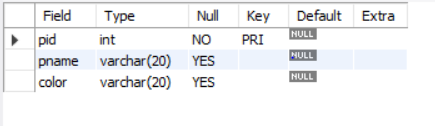
pid int(10),

pname varchar(20),

color varchar(20),

primary key(pid));

desc parts;



use Supriya;

create table catalog

(

s\_id int(10),

pid int(10),

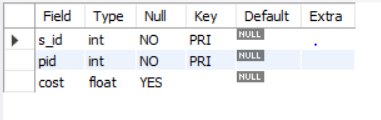
cost float(10),

primary key(pid,s\_id),

foreign key(s\_id)references suppliers(s\_id),

foreign key(pid)references parts(pid));

desc catalog;



use Supriya;

insert into suppliers values(1,'Reliance','Shrinagar');

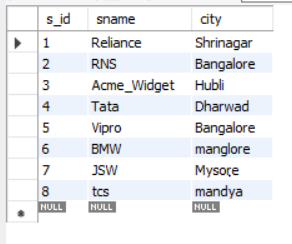
insert into suppliers values(2,'RNS','Bangalore');

insert into suppliers values(3,'Acme\_Widget','Hubli');

insert into suppliers values(4,'Tata','Dharwad');

insert into suppliers values(5,'Vipro','Bangalore');

select \* from suppliers;



use Supriya;

insert into parts values(1,'break','black');

insert into parts values(2,'engine','brown');

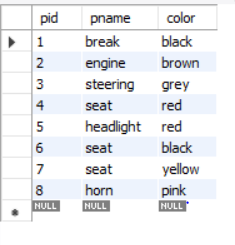
insert into parts values(3,'steering','grey');

insert into parts values(4,'seat','red');

insert into parts values(5,'headlight','red');

insert into parts values(6,'seat','black');

select \* from parts;



use Supriya;

desc catalog;

insert into catalog values(1,1,'1000.0');

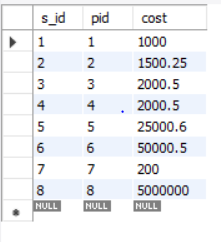
insert into catalog values(2,2,'1500.25');

insert into catalog values(3,3,'2000.50');

insert into catalog values(4,4,'2000.50');

insert into catalog values(5,5,'3000.25');

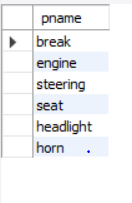
select \* from catalog;



**i)Find the pnames of parts for which there is some supplier.**

use Supriya;

SELECT DISTINCT P.pname FROM parts P, catalog C WHERE P.pid = C.pid;



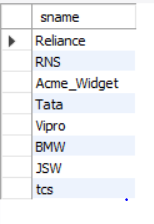
**ii)Find the snames of suppliers who supply every part.**

select s.sname

from suppliers s

where not exists(

select p.pid from parts p where not exists(select c.s\_id from catalog c where c.s\_id=s.s\_id));



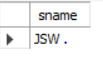
**iii)Find the snames of suppliers who supply every yellow part.**

select s.sname

from suppliers s

where not exists(select p.pid from parts p where p.color='yellow' and not exists(select c.s\_id from catalog c

where c.s\_id=s.s\_id and c.pid=p.pid));



**iv. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.**

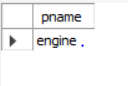
select p.pname

from parts p,catalog c,suppliers s

where p.pid=c.pid and

c.s\_id=s.s\_id and

s.sname='RNS' and not exists(select \* from catalog ca,suppliers su where p.pid=ca.pid and ca.s\_id=su.s\_id and su.sname<>'RNS');

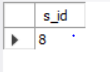


**v.Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).**

select distinct c.s\_id

from catalog c

where c.cost>(select avg(ca.cost) from catalog ca,catalog c where ca.pid=c.pid);



**vi. For each part, find the sname of the supplier who charges the most for that part.**

SELECT P.pid, S.sname

FROM Parts P, Suppliers S, Catalog C

WHERE C.pid = P.pid

AND C.sid = S.sid

AND C.cost = (SELECT MAX (C1.cost)

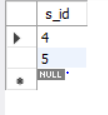
FROM Catalog C1

WHERE C1.pid = P.pid);

**vii. Find the sids of suppliers who supply only red parts**

select s.s\_id from suppliers s where exists(

select c.s\_id from catalog c,parts p where p.color='red' and p.pid=c.pid and c.s\_id=s.s\_id);



**PROGRAM 4.**

**STUDENT FACULTY DATABASE**

**Consider the following database for student enrolment for course:**

**STUDENT (snum: integer, sname: string, major: string, level: string, age: integer)**

**CLASS (name: string, meets at: time, room: string, fid: integer)**

**ENROLLED (snum: integer, cname: string)**

**FACULTY (fid: integer, fname: string, deptid: integer)**

use Supriya;

create table student(

snum INT,

sname VARCHAR(10),

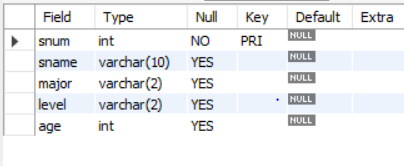
major VARCHAR(2),

level VARCHAR(2),

age INT,

primary key(snum));

desc student;



CREATE TABLE faculty(

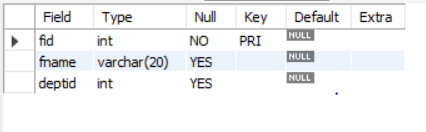
fid INT,

fname VARCHAR(20),

deptid INT,

PRIMARY KEY(fid));

desc faculty;



CREATE TABLE class(

cname VARCHAR(20),

meets\_at TIMESTAMP,

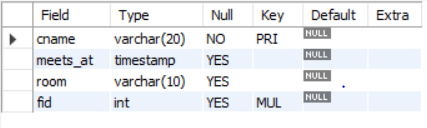
room VARCHAR(10),

fid INT,

PRIMARY KEY(cname),

FOREIGN KEY(fid) REFERENCES faculty(fid));

Desc class;



CREATE TABLE enrolled(

snum INT,

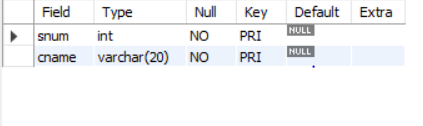
cname VARCHAR(20),

PRIMARY KEY(snum,cname),

FOREIGN KEY(snum) REFERENCES student(snum),

FOREIGN KEY(cname) REFERENCES class(cname));

desc enrolled;



insert into student values(1,'Ram','CS','JR',10);

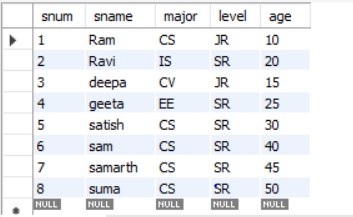
insert into student values(2,'Ravi','IS','SR',20);

insert into student values(3,'deepa','CV','JR',15);

insert into student values(4,'geeta','EE','SR',25);

insert into student values(5,'satish','CS','SR',30);

select \* from student;



insert into faculty values(1,'Seeta',10);

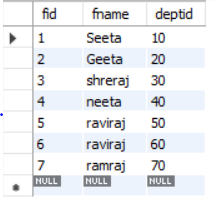
insert into faculty values(2,'Geeta',20);

insert into faculty values(3,'shreraj',30);

insert into faculty values(4,'neeta',40);

insert into faculty values(5,'raviraj',50);

select \* from faculty;



insert into class values('class1','12/11/15 10:15:16',1,1);

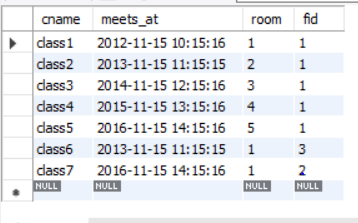
insert into class values('class2','13/11/15 11:15:15',2,2);

insert into class values('class3','14/11/15 12:15:16',3,3);

insert into class values('class4','15/11/15 13:15:16',4,4);

insert into class values('class5','16/11/15 14:15:16',5,5);

select \* from class;



insert into enrolled values(1,'class1');

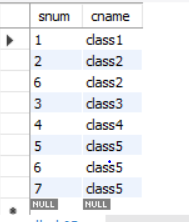
insert into enrolled values(2,'class2');

insert into enrolled values(3,'class3');

insert into enrolled values(4,'class4');

insert into enrolled values(5,'class5');

select \* from enrolled;



**QUERIES:**

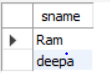
**i)Find the names of all Juniors (level= Jr) who are enrolled in a class taught by Seeta**

use Supriya;

select S.sname from student S where exists

(select S1.snum from student S1 where S.level='JR')AND EXISTS

(select F.fid from faculty F,enrolled E,class C where F.fname='Seeta' AND E.snum=S.snum AND E.cname = C.cname AND C.fid = F.fid );



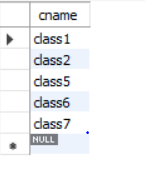
**ii. Find the names of all classes that either meet in room ‘ 1’ or have 2 or more Students enrolled.**

select c.cname

from class c

where c.room='1'

or c.cname in(select e.cname from enrolled e group by e.cname having count(\*)>=2);



**iii)Find the names of all students who are enrolled in two classes that meet at the same time.**

select distinct s.sname

from student s

where s.snum in(select e1.snum

               from enrolled e1,enrolled e2,class c1, class c2

   where e1.snum=e2.snum and e1.cname<>e2.cname

   and e1.cname=c1.cname

               and e2.cname=c2.cname

               and c1.meets\_at=c2.meets\_at);

**iv)Find the names of faculty members who teach in every room in which some class is taught.**

select f.fname

from faculty f

where f.fid in (select fid from class

group by fid having count(\*)=(select count(distinct room) from class));



**v)Find the names of faculty members for whom the combined enrolment of the courses that they teach is less than five.**

SELECT DISTINCT F.fname

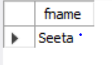
FROM faculty F

WHERE 5<(SELECT COUNT(E.snum)

FROM class C,enrolled E

WHERE C.cname = E.cname

AND C.fid = F.fid);



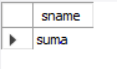
**vi)Find the names of students who are not enrolled in any class**

SELECT DISTINCT S.sname

FROM student S

WHERE S.snum NOT IN (SELECT E.snum

FROM enrolled E );



**vii. For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged**

**18 than SR, JR, or SO students aged 18, you should print the pair (18, FR)**

SELECT S.age, S.level

FROM student S

GROUP BY S.age, S.level

HAVING S.level IN(SELECT S1.level FROM student S1

WHERE S1.age = S.age

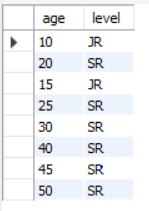
GROUP BY S1.level,S1.age

HAVING COUNT(\*)>= ALL(SELECT COUNT(\*)

FROM student S2

WHERE s1.age = S2.age

GROUP BY S2.level, S2.age));



**PROGRAM 5.**

**AIRLINE FLIGHT DATABASE**

**Consider the following database that keeps track of airline flight information:**

**FLIGHTS (flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer)**

**AIRCRAFT (aid: integer, aname: string, cruisingrange: integer)**

**CERTIFIED (eid: integer, aid: integer)**

**EMPLOYEE (eid: integer, ename: string, salary: integer)**

use Supriya;

CREATE TABLE flights

(

flno INTEGER PRIMARY KEY,

ffrom VARCHAR(15) NOT NULL,

tto VARCHAR(15) NOT NULL,

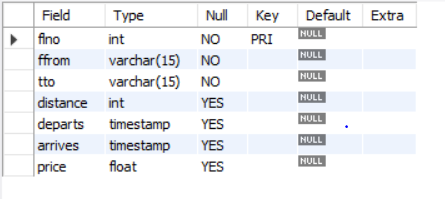
distance INTEGER,

departs TIMESTAMP,

arrives TIMESTAMP,

price float(10));

desc flights;



CREATE TABLE aircraft

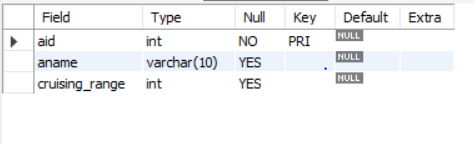
(

aid INTEGER PRIMARY KEY,

aname VARCHAR(10),

cruising\_range INTEGER);

desc aircraft;



CREATE TABLE employee

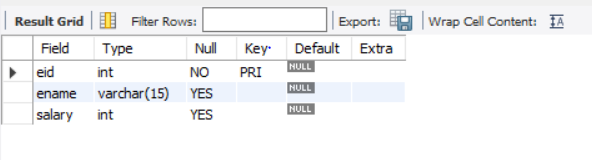
(

eid INTEGER PRIMARY KEY,

ename VARCHAR(15),

salary INTEGER(10));

desc employee;



CREATE TABLE certified

(

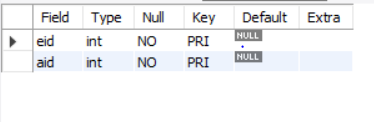
eid INTEGER NOT NULL,

aid INTEGER NOT NULL,

PRIMARY KEY (eid,aid),

FOREIGN KEY (eid) REFERENCES employee (eid), FOREIGN KEY (aid) REFERENCES aircraft (aid));

desc certified;



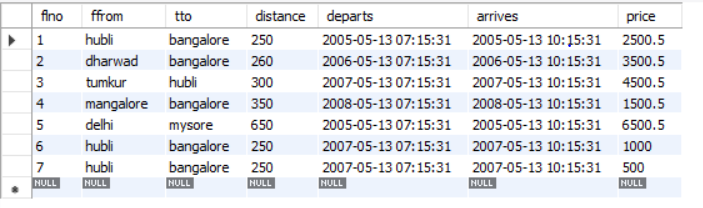
insert into flights values(1,'hubli','bangalore',250,'2005-05-13 07:15:31','2005-05-13 10:15:31','2500.5');

insert into flights values(2,'dharwad','bangalore',260,'2006-05-13 07:15:31','2006-05-13 10:15:31','3500.5');

insert into flights values(3,'tumkur','hubli',300,'2007-05-13 07:15:31','2007-05-13 10:15:31','4500.5');

insert into flights values(4,'mangalore','bangalore',350,'2008-05-13 07:15:31','2008-05-13 10:15:31','1500.5');

insert into flights values(5,'delhi','mysore',650,'2005-05-13 07:15:31','2005-05-13 10:15:31','6500.5');



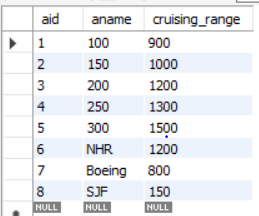
insert into aircraft values(1,'100','900');

insert into aircraft values(2,'150','1000');

insert into aircraft values(3,'200','1200');

insert into aircraft values(4,'250','1300');

insert into aircraft values(5,'300','1500');



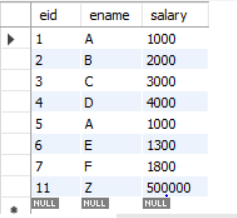
insert into employee values(1,'A',1000);

insert into employee values(2,'B',2000);

insert into employee values(3,'C',3000);

insert into employee values(4,'D',4000);

insert into employee values(5,'A',1000);



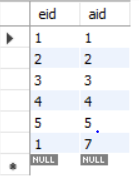
insert into certified values(1,1);

insert into certified values(2,2);

insert into certified values(3,3);

insert into certified values(4,4);

insert into certified values(5,5);



**i)Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.2000**

SELECT DISTINCT A.aname

FROM aircraft A

WHERE A.aid IN (SELECT C.aid

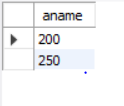
FROM certified C,employee E

WHERE C.eid = E.eid AND

EXISTS( SELECT \*

FROM employee E1

WHERE E1.eid = E.eid AND E1.salary>2000));



**ii)For each pilot who is certified for more than one aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified**

SELECT C.eid, MAX(A.cruising\_range)

FROM certified C,aircraft A

WHERE C.aid = A.aid

GROUP BY C.eid

HAVING COUNT(\*)>1;



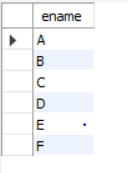
**iii)Find the names of pilots whose salary is less than the price of the cheapest route from delhi to mysore.**

SELECT DISTINCT E.ename

FROM employee E

WHERE E.salary<( SELECT MIN(F.price)

FROM flights F WHERE F.ffrom='delhi' AND F.tto='mysore');



**iv)For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft.**

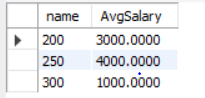
SELECT Temp.name,Temp.AvgSalary

FROM (SELECT A.aid, A.aname AS name, AVG(E.salary) AS AvgSalary

FROM aircraft A,certified C,employee E

WHERE A.aid = C.aid AND C.eid = E.eid AND A.cruising\_range>1000

GROUP BY A.aid, A.aname )Temp;

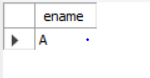


**v)Find the names of pilots certified for ‘300’ named aircraft.**

SELECT DISTINCT E.ename

FROM employee E, certified C, aircraft A

WHERE E.eid = C.eid AND C.aid = A.aid AND A.aname LIKE '300';



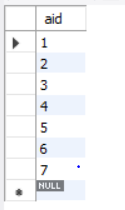
**vi)Find the aids of all aircraft that can be used on routes from tumkut to hubli.**

SELECT A.aid

FROM aircraft A

WHERE A.cruising\_range>(SELECT MIN(F.distance)

FROM flights F WHERE F.ffrom ='tumkur' AND F.tto='hubli');



**vii)A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.**

SELECT F.departs

FROM flights F

WHERE F.flno IN (( SELECT F0.flno

FROM flights F0

WHERE F0.ffrom ='hubli' AND F0.tto = 'bangalore'

AND extract(hour from F0.arrives)<18 )

UNION

( SELECT F0.flno

FROM flights F0,flights F1

WHERE F0.ffrom = 'hubli' AND F0.tto <>'bangalore'

AND F0.tto = F1.ffrom AND F1.tto ='bangalore'

AND F1.departs > F0.arrives

AND extract(hour from F1.arrives) < 18)

UNION

( SELECT F0.flno

FROM flights F0,flights F1,flights F2

WHERE F0.ffrom ='hubli'

AND F0.tto = F1.ffrom

AND F1.tto = F2.ffrom

AND F2.tto = 'bangalore'

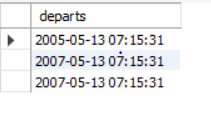
AND F0.tto <> 'bangalore'

AND F1.tto <> 'bangalore'

AND F1.departs > F0.arrives

AND F2.departs > F1.arrives

AND extract(hour from F2.arrives) < 18));



**viii. Print the name and salary of every non-pilot whose salary is more than the average salary for pilots**

SELECT E.ename, E.salary

FROM employee E

WHERE E.eid NOT IN( SELECT DISTINCT C.eid

FROM certified C)

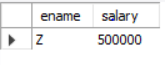
AND E.salary>(SELECT AVG(E1.salary)

FROM employee E1

WHERE E1.eid IN

( SELECT DISTINCT C1.eid

FROM certified C1 ) );



**PROGRAM 6.**

**ORDER PROCESSING DATABASE**

CREATE DATABASE ORDPROCESS;

USE ORDPROCESS;

CREATE TABLE SALESMAN(SALESMAN\_ID INT,

NAME VARCHAR (20),

CITY VARCHAR (20),

COMMISSION VARCHAR (20),

PRIMARY KEY (SALESMAN\_ID));

CREATE TABLE CUSTOMER

(CUSTOMER\_ID INT,

CUST\_NAME VARCHAR (20),

CITY VARCHAR (20),

GRADE INT,

SALESMAN\_ID INT,

PRIMARY KEY (CUSTOMER\_ID),

FOREIGN KEY(SALESMAN\_ID) REFERENCES SALESMAN (SALESMAN\_ID) ON DELETE SET NULL);

CREATE TABLE ORDERS

(ORD\_NO INT,

PURCHASE\_AMT REAL,

ORD\_DATE DATE,

CUSTOMER\_ID INT,

SALESMAN\_ID INT,

PRIMARY KEY (ORD\_NO),

FOREIGN KEY(CUSTOMER\_ID) REFERENCES CUSTOMER(CUSTOMER\_ID) ON DELETE CASCADE,

FOREIGN KEY(SALESMAN\_ID) REFERENCES SALESMAN(SALESMAN\_ID) ON DELETE CASCADE);

INSERT INTO SALESMAN VALUES (1000,'JOHN','BANGALORE','25 %');

INSERT INTO SALESMAN VALUES (2000,'RAVI','BANGALORE','20 %');

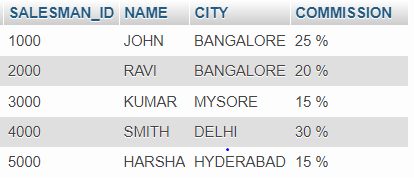
INSERT INTO SALESMAN VALUES (3000,'KUMAR','MYSORE','15 %');

INSERT INTO SALESMAN VALUES (4000,'SMITH','DELHI','30 %');

INSERT INTO SALESMAN VALUES (5000,'HARSHA','HYDERABAD','15 %');

COMMIT;

Select \* from SALESMAN;



INSERT INTO CUSTOMER VALUES (10,'PREETHI','BANGALORE', 100, 1000);

INSERT INTO CUSTOMER VALUES (11,'VIVEK','MANGALORE', 300, 1000);

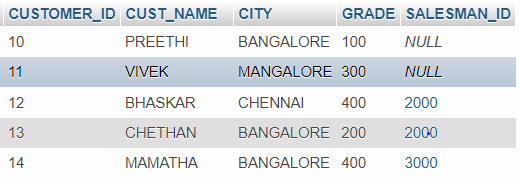
INSERT INTO CUSTOMER VALUES (12,'BHASKAR','CHENNAI', 400, 2000);

INSERT INTO CUSTOMER VALUES (13,'CHETHAN','BANGALORE', 200, 2000);

INSERT INTO CUSTOMER VALUES (14,'MAMATHA','BANGALORE', 400, 3000);

COMMIT;

SELECT \* FROM CUSTOMER;



INSERT INTO ORDERS VALUES (50, 5000,'2017-05-04',10,1000);

INSERT INTO ORDERS VALUES (51, 450,'2017-01-20',10,2000);

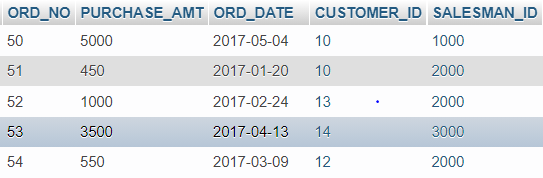
INSERT INTO ORDERS VALUES (52, 1000,'2017-02-24',13,2000);

INSERT INTO ORDERS VALUES (53, 3500,'2017-04-13',14,3000);

INSERT INTO ORDERS VALUES (54, 550,'2017-03-09',12,2000);

COMMIT;

SELECT \* FROM ORDERS;



**QUERIES:**

**1)Count the customers with grades above Bangalore’s average.**

SELECT GRADE, COUNT(DISTINCT CUSTOMER\_ID)

FROM CUSTOMER

GROUP BY GRADE

HAVING GRADE > (SELECT AVG(GRADE) FROM CUSTOMER WHERE CITY='BANGALORE');



**2)Find the name and numbers of all salesmen who had more than one customer.**

SELECT SALESMAN\_ID, NAME

FROM SALESMAN S1

WHERE 1 < (SELECT COUNT(\*)

FROM CUSTOMER

WHERE SALESMAN\_ID=S1.SALESMAN\_ID);



**3)List all salesmen and indicate those who have and don’t have customers in their cities**

SELECT S.SALESMAN\_ID, NAME, CUST\_NAME, COMMISSION

FROM SALESMAN S, CUSTOMER C

WHERE S.CITY = C.CITY

UNION

SELECT SALESMAN\_ID, NAME, 'NO MATCH', COMMISSION

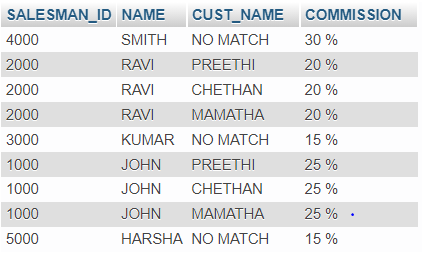
FROM SALESMAN

WHERE NOT CITY = ANY

(SELECT CITY

FROM CUSTOMER)

ORDER BY 2 DESC;



**4)Create a view that finds the salesman who has the customer with the highest order of a day.**

CREATE VIEW ELITESALESMAN AS

SELECT O.ORD\_DATE, S.SALESMAN\_ID, S.NAME

FROM SALESMAN S, ORDERS O

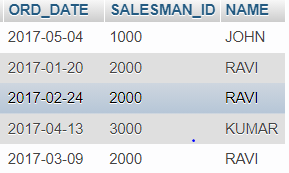
WHERE S.SALESMAN\_ID = O.SALESMAN\_ID

AND O.PURCHASE\_AMT=(SELECT MAX(PURCHASE\_AMT)

FROM ORDERS O1

WHERE O1.ORD\_DATE = O.ORD\_DATE);

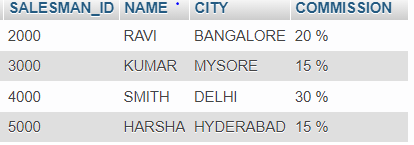
SELECT \* FROM ELITESALESMAN;



**5)  Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.**

DELETE FROM SALESMAN

WHERE SALESMAN\_ID=1000;



**PROGRAM 7**

**BOOK DEALER DATABASE**

CREATE TABLE author

(

author\_id int(20),

author\_name varchar(20),

city varchar(10),

country varchar(10),

PRIMARY key(author\_id)

);

create table publisher

(

pid int(10),

pname varchar(20),

city varchar(20),

country varchar(10),

primary key(pid));

create table catalog

(

book\_id int(10),

title varchar(20),

author\_id int(20),

pid int(10),

category\_id int(10),

year int,

price int,

primary key(author\_id,pid,book\_id,category\_id),

foreign key(author\_id)references author(author\_id),

foreign key(pid)references publisher(pid));

CREATE TABLE order\_details

(

order\_id INT,

book\_id INT,

quantity INT,

PRIMARY KEY(order\_id),

FOREIGN KEY(book\_id) REFERENCES catalog(book\_id));

CREATE TABLE category(

category\_id INT,

category\_desc VARCHAR(30),

PRIMARY KEY(category\_id));

INSERT INTO author VALUES(1001,'JK Rowling','London','England');

INSERT INTO author VALUES(1002,'Chetan Bhagat','Delhi','India');

INSERT INTO author VALUES(1003,'John McCarthy','Chicago','USA');

INSERT INTO author VALUES(1004,'Dan Brown','California','USA');

COMMIT;

SELECT \* FROM author;



INSERT INTO publisher VALUES(2001,'Bloomsbury','London','England');

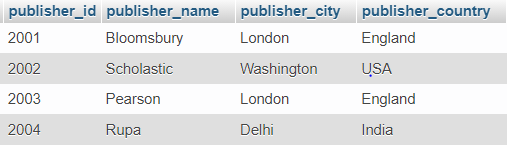
INSERT INTO publisher VALUES(2002,'Scholastic','Washington','USA');

INSERT INTO publisher VALUES(2003,'Pearson','London','England');

INSERT INTO publisher VALUES(2004,'Rupa','Delhi','India') ;

COMMIT;

SELECT \* FROM publisher;



INSERT INTO category VALUES(3001,'Fiction');

INSERT INTO category VALUES(3002,'Non-Fiction');

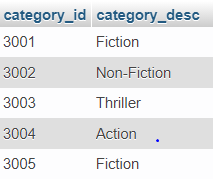
INSERT INTO category VALUES(3003,'Thriller');

INSERT INTO category VALUES(3004,'Action');

INSERT INTO category VALUES(3005,'Fiction');

COMMIT;

SELECT \* FROM category;



INSERT INTO catalog VALUES(4001,'HP and Goblet Of Fire',1001,2001,3001,2002,600);

INSERT INTO catalog VALUES(4002,'HP and Order Of Phoenix',1001,2002,3001,2005,650);

INSERT INTO catalog VALUES(4003,'Two States',1002,2004,3001,2009,65);

INSERT INTO catalog VALUES(4004,'3 Mistakes of my life',1002,2004,3001,2007,55);

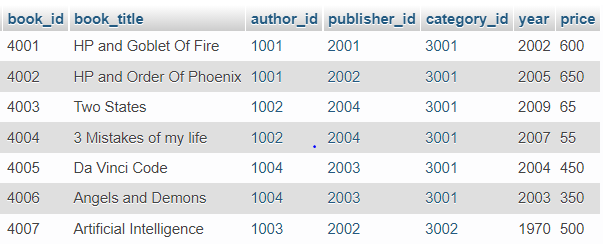
INSERT INTO catalog VALUES(4005,'Da Vinci Code',1004,2003,3001,2004,450);

INSERT INTO catalog VALUES(4006,'Angels and Demons',1004,2003,3001,2003,350);

INSERT INTO catalog VALUES(4007,'Artificial Intelligence',1003,2002,3002,1970,500);

COMMIT;

SELECT \* FROM catalog;



INSERT INTO order\_details VALUES(5001,4001,5);

INSERT INTO order\_details VALUES(5002,4002,7);

INSERT INTO order\_details VALUES(5003,4003,15);

INSERT INTO order\_details VALUES(5004,4004,11);

INSERT INTO order\_details VALUES(5005,4005,9);

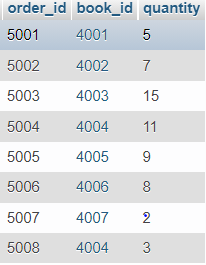
INSERT INTO order\_details VALUES(5006,4006,8);

INSERT INTO order\_details VALUES(5007,4007,2);

INSERT INTO order\_details VALUES(5008,4004,3);

COMMIT;

SELECT \* FROM order\_details;



**QUERIES:**

**1)Give the details of the authors who have 2 or more books in the catalog and the price of the books is greater than the average price of the books in the catalog and the year of publication is after 2000.**

SELECT \* FROM author

WHERE author\_id IN

(SELECT author\_id FROM catalog

WHERE year>2000 AND price>(SELECT AVG(price) FROM catalog)

GROUP BY author\_id HAVING COUNT(\*)>1);



**2)Find the author of the book which has maximum sales.**

SELECT a.author\_name FROM author a,catalog c WHERE a.author\_id=c.author\_id AND book\_id IN (SELECT book\_id FROM order\_details WHERE quantity = (SELECT MAX(quantity) FROM order\_details));



**3)Demonstrate how you increase the price of books published by a specific publisher by 10%.**

SET SQL\_SAFE\_UPDATES = 0;

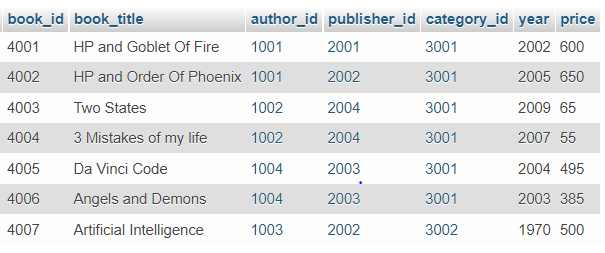
UPDATE catalog SET price=1.1\*price

WHERE publisher\_id IN

(SELECT publisher\_id FROM publisher WHERE publisher\_name='pearson');

SELECT \* FROM catalog;

SET SQL\_SAFE\_UPDATES = 1;



**PROGRAM 8**

**STUDENT ENROLLMENT DATABASE**

CREATE DATABASE student\_enroll;

USE student\_enroll;

CREATE TABLE student(

regno VARCHAR(15),

name VARCHAR(20),

major VARCHAR(20),

bdate DATE,

PRIMARY KEY (regno));

CREATE TABLE course(

courseno INT,

cname VARCHAR(20),

dept VARCHAR(20),

PRIMARY KEY (courseno));

CREATE TABLE enroll(

regno VARCHAR(15),

courseno INT,

sem INT,

marks INT,

PRIMARY KEY (regno,courseno),

FOREIGN KEY (regno) REFERENCES student(regno),

FOREIGN KEY (courseno) REFERENCES course(courseno));

CREATE TABLE textbook(

book\_isbn INT,

book\_title VARCHAR(20),

publisher VARCHAR(20),

author VARCHAR(20),

PRIMARY KEY (book\_isbn));

CREATE TABLE book\_adoption(

courseno INT,

sem INT,

book\_isbn INT,

PRIMARY KEY (courseno,book\_isbn),

FOREIGN KEY (courseno) REFERENCES course (courseno),

FOREIGN KEY (book\_isbn) REFERENCES textbook(book\_isbn));

INSERT INTO student VALUES('1BM11CS001','A','Sr','19931230');

INSERT INTO student VALUES('1BM11CS002','B','Sr','19930924');

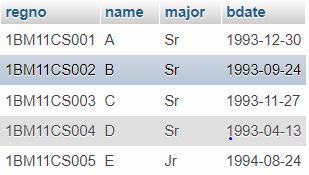
INSERT INTO student VALUES('1BM11CS003','C','Sr','19931127');

INSERT INTO student VALUES('1BM11CS004','D','Sr','19930413');

INSERT INTO student VALUES('1BM11CS005','E','Jr','19940824');

COMMIT;

SELECT \* FROM student;



INSERT INTO course VALUES(111,'OS','CSE');

INSERT INTO course VALUES(112,'EC','ECE');

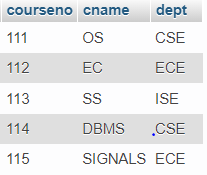
INSERT INTO course VALUES(113,'SS','ISE');

INSERT INTO course VALUES(114,'DBMS','CSE');

INSERT INTO course VALUES(115,'SIGNALS','ECE');

COMMIT;

SELECT \* FROM course;



INSERT INTO textbook VALUES(10,'DATABASE SYSTEMS','PEARSON','SCHIELD');

INSERT INTO textbook VALUES(900,'OPERATING SYSTEMS','PEARSON','LELAND');

INSERT INTO textbook VALUES(901,'CIRCUITS','HALL INDIA','BOB');

INSERT INTO textbook VALUES(902,'SYSTEM SOFTWARE','PETERSON','JACOB');

INSERT INTO textbook VALUES(903,'SCHEDULING','PEARSON','PATIL');

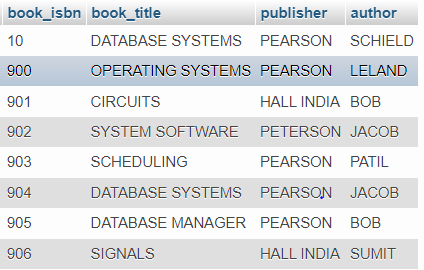
INSERT INTO textbook VALUES(904,'DATABASE SYSTEMS','PEARSON','JACOB');

INSERT INTO textbook VALUES(905,'DATABASE MANAGER','PEARSON','BOB');

INSERT INTO textbook VALUES(906,'SIGNALS','HALL INDIA','SUMIT');

COMMIT;

SELECT \* FROM textbook;



INSERT INTO enroll VALUES('1BM11CS001',115,3,100);

INSERT INTO enroll VALUES('1BM11CS002',114,5,100);

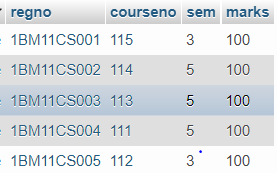
INSERT INTO enroll VALUES('1BM11CS003',113,5,100);

INSERT INTO enroll VALUES('1BM11CS004',111,5,100);

INSERT INTO enroll VALUES('1BM11CS005',112,3,100);

COMMIT;

SELECT \* FROM enroll;



INSERT INTO book\_adoption VALUES(111,5,900);

INSERT INTO book\_adoption VALUES(111,5,903);

INSERT INTO book\_adoption VALUES(111,5,904);

INSERT INTO book\_adoption VALUES(112,3,901);

INSERT INTO book\_adoption VALUES(113,3,10);

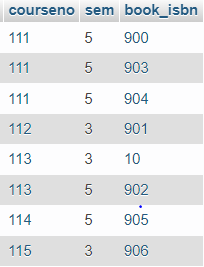
INSERT INTO book\_adoption VALUES(114,5,905);

INSERT INTO book\_adoption VALUES(113,5,902);

INSERT INTO book\_adoption VALUES(115,3,906);

COMMIT;

SELECT \* FROM book\_adoption;



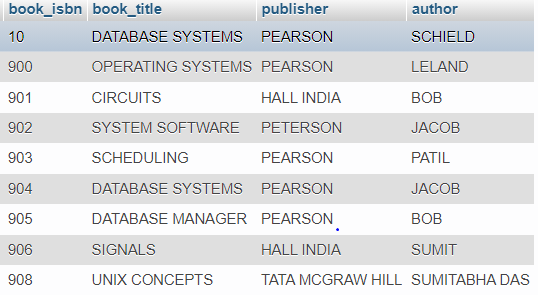
**QUERIES:**

**1)Demonstrate how you add a new text book to the database and make this book be adopted by some department.**

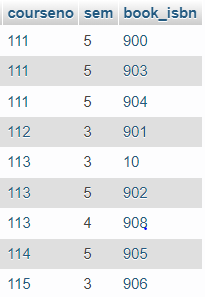
INSERT INTO textbook VALUES(908,'UNIX CONCEPTS','TATA MCGRAW HILL','SUMITABHA DAS');

INSERT INTO book\_adoption VALUES(113,4,908);

SELECT \* FROM textbook;



SELECT \* FROM book\_adoption;



**2)Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the ‘CS’ department that use more than two books.**

SELECT c.courseno,t.book\_isbn,t.book\_title

FROM course c,book\_adoption ba,textbook t

WHERE c.courseno=ba.courseno

AND ba.book\_isbn=t.book\_isbn

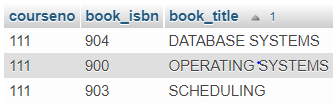
AND c.dept='CSE'

AND 2<(SELECT COUNT(book\_isbn)

FROM book\_adoption b

WHERE c.courseno=b.courseno)

ORDER BY t.book\_title;



**3)List any department that has all its adopted books published by a specific publisher.**

SELECT DISTINCT c.dept

FROM course c

WHERE c.dept IN

(SELECT c.dept

FROM course c,book\_adoption b,textbook t

WHERE c.courseno=b.courseno

AND t.book\_isbn=b.book\_isbn

AND t.publisher='PEARSON')

AND c.dept NOT IN

(SELECT c.dept

FROM course c,book\_adoption b,textbook t

WHERE c.courseno=b.courseno

AND t.book\_isbn=b.book\_isbn

AND t.publisher != 'PEARSON');



**PROGRAM 9**

**MOVIE DATABASE**

CREATE DATABASE MOVIE;

USE MOVIE;

CREATE TABLE ACTOR (

ACT\_ID INT,

ACT\_NAME VARCHAR (20),

ACT\_GENDER CHAR (1),

PRIMARY KEY (ACT\_ID));

DESC ACTOR;

CREATE TABLE DIRECTOR (

DIR\_ID INT,

DIR\_NAME VARCHAR (20),

DIR\_PHONE REAL,

PRIMARY KEY (DIR\_ID));

DESC DIRECTOR;

CREATE TABLE MOVIES (

MOV\_ID INT,

MOV\_TITLE VARCHAR (25),

MOV\_YEAR INT,

MOV\_LANG VARCHAR (12),

DIR\_ID INT,

PRIMARY KEY (MOV\_ID),

FOREIGN KEY (DIR\_ID) REFERENCES DIRECTOR (DIR\_ID));

DESC MOVIES;

CREATE TABLE MOVIE\_CAST (

ACT\_ID INT,

MOV\_ID INT,

ROLE VARCHAR (10),

PRIMARY KEY (ACT\_ID, MOV\_ID),

FOREIGN KEY (ACT\_ID) REFERENCES ACTOR (ACT\_ID),

FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

DESC MOVIE\_CAST;

CREATE TABLE RATING (

MOV\_ID INT,

REV\_STARS VARCHAR (25),

PRIMARY KEY (MOV\_ID),

FOREIGN KEY (MOV\_ID) REFERENCES MOVIES (MOV\_ID));

DESC RATING;

INSERT INTO ACTOR VALUES (301,'ANTHONY PERKINS','M');

INSERT INTO ACTOR VALUES (302,'SAOIRSE RONAN','F');

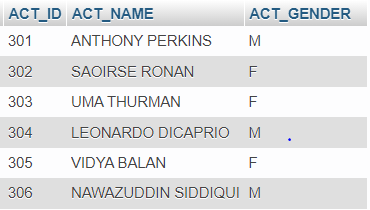
INSERT INTO ACTOR VALUES (303,'UMA THURMAN','F');

INSERT INTO ACTOR VALUES (304,'LEONARDO DICAPRIO','M');

INSERT INTO ACTOR VALUES (305,'VIDYA BALAN','F');

INSERT INTO ACTOR VALUES (306,'NAWAZUDDIN SIDDIQUI','M');

SELECT \* FROM ACTOR;



INSERT INTO DIRECTOR VALUES (60,'GRETA GERWIG', 8751611001);

INSERT INTO DIRECTOR VALUES (61,'HITCHCOCK', 7766138911);

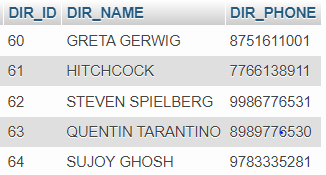
INSERT INTO DIRECTOR VALUES (62,'STEVEN SPIELBERG', 9986776531);

INSERT INTO DIRECTOR VALUES (63,'QUENTIN TARANTINO', 8989776530);

INSERT INTO DIRECTOR VALUES (64,'SUJOY GHOSH', 9783335281);

COMMIT;

SELECT \* FROM DIRECTOR;



INSERT INTO MOVIES VALUES (1001,'LITTLE WOMEN',2019,'ENGLISH',60);

INSERT INTO MOVIES VALUES (1002,'KAHAANI',2012,'HINDI',64);

INSERT INTO MOVIES VALUES (1003,'KILL BILL : VOL.1',2003,'ENGLISH',63);

INSERT INTO MOVIES VALUES (1004,'PSYCHO',1960,'ENGLISH',61);

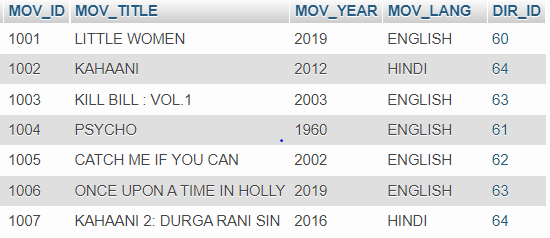
INSERT INTO MOVIES VALUES (1005,'CATCH ME IF YOU CAN',2002,'ENGLISH',62);

INSERT INTO MOVIES VALUES (1006,'ONCE UPON A TIME IN HOLLYWOOD',2019,'ENGLISH',63);

INSERT INTO MOVIES VALUES (1007,'KAHAANI 2: DURGA RANI SINGH',2016,'HINDI',64);

COMMIT;

SELECT \* FROM MOVIES;



INSERT INTO MOVIE\_CAST VALUES (301,1004,'HERO');

INSERT INTO MOVIE\_CAST VALUES (302,1001,'HEROINE');

INSERT INTO MOVIE\_CAST VALUES (303,1003,'HEROINE');

INSERT INTO MOVIE\_CAST VALUES (304,1005,'HERO');

INSERT INTO MOVIE\_CAST VALUES (305,1002,'HEROINE');

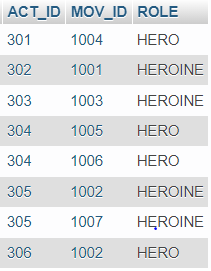
INSERT INTO MOVIE\_CAST VALUES (306,1002,'HERO');

INSERT INTO MOVIE\_CAST VALUES (304,1006,'HERO');

INSERT INTO MOVIE\_CAST VALUES (305,1007,'HEROINE');

COMMIT;

SELECT \* FROM MOVIE\_CAST;



INSERT INTO RATING VALUES (1001, 4);

INSERT INTO RATING VALUES (1002, 4);

INSERT INTO RATING VALUES (1003, 3);

INSERT INTO RATING VALUES (1004, 5);

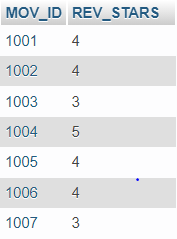
INSERT INTO RATING VALUES (1005, 4);

INSERT INTO RATING VALUES (1006, 4);

INSERT INTO RATING VALUES (1007, 3);

COMMIT;

SELECT \* FROM RATING;



**QUERIES:**

**1)List the titles of all movies directed by 'SUJOY GHOSH'.**

SELECT MOV\_TITLE

FROM MOVIES

WHERE DIR\_ID IN (SELECT DIR\_ID

FROM DIRECTOR

WHERE DIR\_NAME = 'SUJOY GHOSH');



**2)Find the movie names where one or more actors acted in two or more movies.**

SELECT MOV\_TITLE

FROM MOVIES M, MOVIE\_CAST MV

WHERE M.MOV\_ID=MV.MOV\_ID AND ACT\_ID IN (SELECT ACT\_ID

FROM MOVIE\_CAST GROUP BY ACT\_ID

HAVING COUNT(ACT\_ID)>=1)

GROUP BY MOV\_TITLE

HAVING COUNT(\*)>1;



**3) List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**

SELECT ACT\_NAME, MOV\_TITLE, MOV\_YEAR

FROM ACTOR A

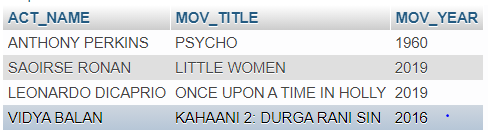
JOIN MOVIE\_CAST C

ON A.ACT\_ID=C.ACT\_ID

JOIN MOVIES M

ON C.MOV\_ID=M.MOV\_ID

WHERE M.MOV\_YEAR NOT BETWEEN 2000 AND 2015;



**4)Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**

SELECT MOV\_TITLE, MAX(REV\_STARS)

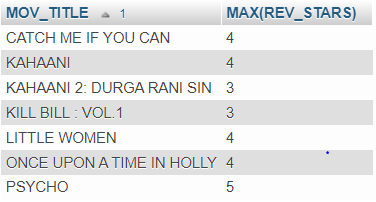
FROM MOVIES

INNER JOIN RATING USING (MOV\_ID)

GROUP BY MOV\_TITLE

HAVING MAX(REV\_STARS)>0

ORDER BY MOV\_TITLE;



**5)Update rating of all movies directed by ‘Steven Spielberg’ to 5**

SET SQL\_SAFE\_UPDATES = 0;

UPDATE RATING

SET REV\_STARS=5

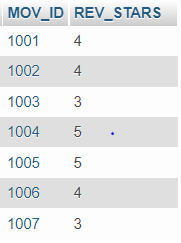
WHERE MOV\_ID IN (SELECT MOV\_ID FROM MOVIES

WHERE DIR\_ID IN (SELECT DIR\_ID FROM DIRECTOR

WHERE DIR\_NAME = 'STEVEN SPIELBERG'));

SELECT \* FROM RATING;

SET SQL\_SAFE\_UPDATES = 1;



**PROGRAM 10**

**COLLEGE DATABASE**

create database college;

use college;

create table student(

usn varchar(10) primary key,

sname varchar(25),

address varchar(25),

phone bigint,

gender char(1));

desc student;

create table semsec(

ssid varchar(5) primary key,

sem int,

sec char(1));

desc semsec;

create table class(

usn varchar(10),

ssid varchar(5),

primary key(usn,ssid),

foreign key(usn) references student(usn),

foreign key(ssid) references semsec(ssid));

desc class;

create table subject(

subcode varchar(8),

title varchar(20),

sem int,

credits int,

primary key(subcode));

desc subject;

create table iamarks(

usn varchar(10),

subcode varchar(8),

ssid varchar(5),

test1 int,

test2 int,

test3 int,

finalia int,

primary key (usn,subcode,ssid),

foreign key (usn) references student (usn),

foreign key (subcode) references subject (subcode),

foreign key (ssid) references semsec (ssid));

desc iamarks;

INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI', 8877881122,'M');

INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU', 7722829912,'F');

INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU', 7712312312,'F');

INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU', 8877881122,'F');

INSERT INTO STUDENT VALUES ('1RN14CS010','ABHAY','BENGALURU', 9900211201,'M');

INSERT INTO STUDENT VALUES ('1RN14CS032','BHASKAR','BENGALURU', 9923211099,'M');

INSERT INTO STUDENT VALUES ('1RN14CS025','ASMI','BENGALURU', 7894737377,'F');

INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');

INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE', 7696772121,'F');

INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 9944850121,'M');

INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU', 8812332201,'M');

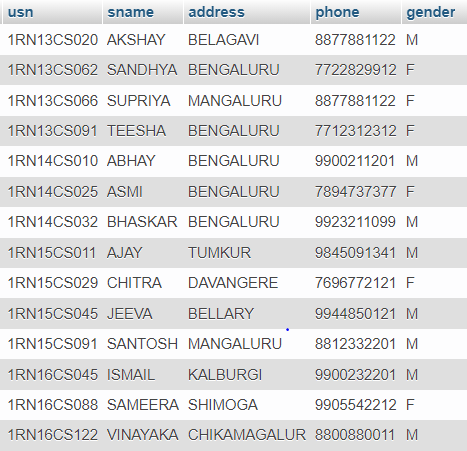
INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI', 9900232201,'M');

INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA', 9905542212,'F');

INSERT INTO STUDENT VALUES ('1RN16CS122','VINAYAKA','CHIKAMAGALUR', 8800880011,'M');

COMMIT;

SELECT \* FROM STUDENT;



INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');

INSERT INTO SEMSEC VALUES ('CSE8B', 8,'B');

INSERT INTO SEMSEC VALUES ('CSE8C', 8,'C');

INSERT INTO SEMSEC VALUES ('CSE7A', 7,'A');

INSERT INTO SEMSEC VALUES ('CSE7B', 7,'B');

INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C');

INSERT INTO SEMSEC VALUES ('CSE6A', 6,'A');

INSERT INTO SEMSEC VALUES ('CSE6B', 6,'B');

INSERT INTO SEMSEC VALUES ('CSE6C', 6,'C');

INSERT INTO SEMSEC VALUES ('CSE5A', 5,'A');

INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');

INSERT INTO SEMSEC VALUES ('CSE5C', 5,'C');

INSERT INTO SEMSEC VALUES ('CSE4A', 4,'A');

INSERT INTO SEMSEC VALUES ('CSE4B', 4,'B');

INSERT INTO SEMSEC VALUES ('CSE4C', 4,'C');

INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');

INSERT INTO SEMSEC VALUES ('CSE3B', 3,'B');

INSERT INTO SEMSEC VALUES ('CSE3C', 3,'C');

INSERT INTO SEMSEC VALUES ('CSE2A', 2,'A');

INSERT INTO SEMSEC VALUES ('CSE2B', 2,'B');

INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');

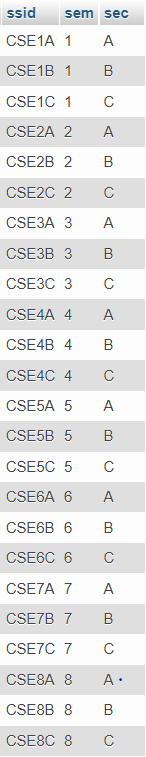
INSERT INTO SEMSEC VALUES ('CSE1A', 1,'A');

INSERT INTO SEMSEC VALUES ('CSE1B', 1,'B');

INSERT INTO SEMSEC VALUES ('CSE1C', 1,'C');

COMMIT;

SELECT \* FROM SEMSEC;



INSERT INTO CLASS VALUES ('1RN13CS020','CSE8A');

INSERT INTO CLASS VALUES ('1RN13CS062','CSE8A');

INSERT INTO CLASS VALUES ('1RN13CS066','CSE8B');

INSERT INTO CLASS VALUES ('1RN13CS091','CSE8C');

INSERT INTO CLASS VALUES ('1RN14CS010','CSE7A');

INSERT INTO CLASS VALUES ('1RN14CS025','CSE7A');

INSERT INTO CLASS VALUES ('1RN14CS032','CSE7A');

INSERT INTO CLASS VALUES ('1RN15CS011','CSE4A');

INSERT INTO CLASS VALUES ('1RN15CS029','CSE4A');

INSERT INTO CLASS VALUES ('1RN15CS045','CSE4B');

INSERT INTO CLASS VALUES ('1RN15CS091','CSE4C');

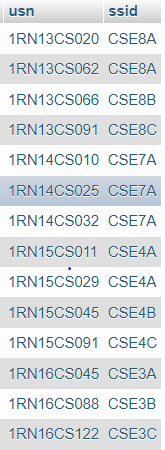
INSERT INTO CLASS VALUES ('1RN16CS045','CSE3A');

INSERT INTO CLASS VALUES ('1RN16CS088','CSE3B');

INSERT INTO CLASS VALUES ('1RN16CS122','CSE3C');

COMMIT;

SELECT \* FROM CLASS;



INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4);

INSERT INTO SUBJECT VALUES ('10CS82','SSM', 8, 4);

INSERT INTO SUBJECT VALUES ('10CS83','NM', 8, 4);

INSERT INTO SUBJECT VALUES ('10CS84','CC', 8, 4);

INSERT INTO SUBJECT VALUES ('10CS85','PW', 8, 4);

INSERT INTO SUBJECT VALUES ('10CS71','OOAD', 7, 4);

INSERT INTO SUBJECT VALUES ('10CS72','ECS', 7, 4);

INSERT INTO SUBJECT VALUES ('10CS73','PTW', 7, 4);

INSERT INTO SUBJECT VALUES ('10CS74','DWDM', 7, 4);

INSERT INTO SUBJECT VALUES ('10CS75','JAVA', 7, 4);

INSERT INTO SUBJECT VALUES ('10CS76','SAN', 7, 4);

INSERT INTO SUBJECT VALUES ('15CS51', 'ME', 5, 4);

INSERT INTO SUBJECT VALUES ('15CS52','CN', 5, 4);

INSERT INTO SUBJECT VALUES ('15CS53','DBMS', 5, 4);

INSERT INTO SUBJECT VALUES ('15CS54','ATC', 5, 4);

INSERT INTO SUBJECT VALUES ('15CS55','JAVA', 5, 3);

INSERT INTO SUBJECT VALUES ('15CS56','AI', 5, 3);

INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4);

INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4);

INSERT INTO SUBJECT VALUES ('15CS43','DAA', 4, 4);

INSERT INTO SUBJECT VALUES ('15CS44','MPMC', 4, 4);

INSERT INTO SUBJECT VALUES ('15CS45','OOC', 4, 3);

INSERT INTO SUBJECT VALUES ('15CS46','DC', 4, 3);

INSERT INTO SUBJECT VALUES ('15CS31','M3', 3, 4);

INSERT INTO SUBJECT VALUES ('15CS32','ADE', 3, 4);

INSERT INTO SUBJECT VALUES ('15CS33','DSA', 3, 4);

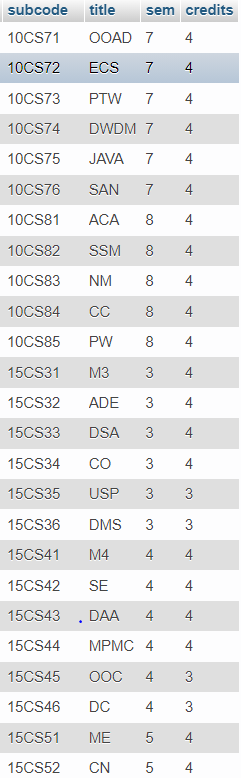
INSERT INTO SUBJECT VALUES ('15CS34','CO', 3, 4);

INSERT INTO SUBJECT VALUES ('15CS35','USP', 3, 3);

INSERT INTO SUBJECT VALUES ('15CS36','DMS', 3, 3);

COMMIT;

SELECT \* FROM SUBJECT;



INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS81','CSE8C', 15, 16, 18);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS82','CSE8C', 12, 19, 14);

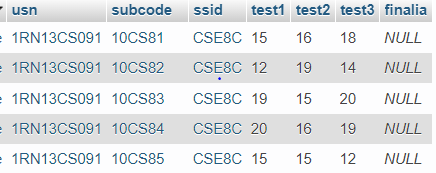
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS83','CSE8C', 19, 15, 20);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS84','CSE8C', 20, 16, 19);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES ('1RN13CS091','10CS85','CSE8C', 15, 15, 12);

COMMIT;

SELECT \* FROM IAMARKS;



**QUERIES:**

**1)List all the student details studying in fourth semester ‘A’ section.**

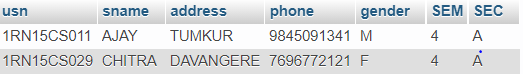
SELECT S.\*, SS.SEM, SS.SEC

FROM STUDENT S, SEMSEC SS, CLASS C

WHERE S.USN = C.USN AND

SS.SSID = C.SSID AND

SS.SEM = 4 AND SS.SEC='A';



**2)Compute the total number of male and female students in each semester and in each section**.

SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER) AS COUNT

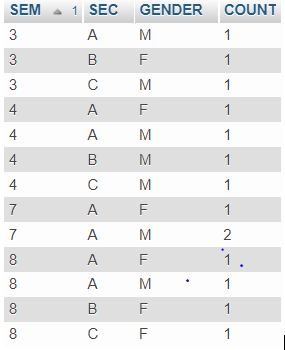
FROM STUDENT S, SEMSEC SS, CLASS C

WHERE S.USN = C.USN AND

SS.SSID = C.SSID

GROUP BY SS.SEM, SS.SEC, S.GENDER

ORDER BY SEM;



**3)Create a view of Test1 marks of student USN ‘1RN13CS091’ in all subjects.**

CREATE VIEW STU\_TEST1\_MARKS\_VIEW

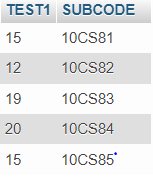
AS

SELECT TEST1, SUBCODE

FROM IAMARKS

WHERE USN = '1RN13CS091';

SELECT \* FROM STU\_TEST1\_MARKS\_VIEW;



**4)Categorize students based on the following criterion: If FinalIA = 17 to 20 then CAT = ‘Outstanding’ If FinalIA = 12 to 16 then CAT = ‘Average’ If FinalIA< 12 then CAT = ‘Weak’ Give these details only for 8th semester A, B, and C section students.**

SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,(CASE

WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'

WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'

ELSE 'WEAK'

END) AS CAT

FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB

WHERE S.USN = IA.USN AND

SS.SSID = IA.SSID AND

SUB.SUBCODE = IA.SUBCODE AND

SUB.SEM = 8;

