**DBMS LAB RECORD**

NAME – SUDESHNA BHUSHAN

USN – 1BM19CS189

SEMESTER – 4

SECTION – D

PROGRAM 1: INSURANCE DATABASE

Consider the Insurance database given below. The data types are specified.

PERSON (driver\_id: String, name: String, address: String)

CAR (reg\_num: String, model: String, year: int)

ACCIDENT (report\_num: int, accident\_date: date, location: String)

OWNS (driver\_id: String, reg\_num: String)

PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)

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

ii)Enter at least five tuples for each relation.

iii)Demonstrate how you

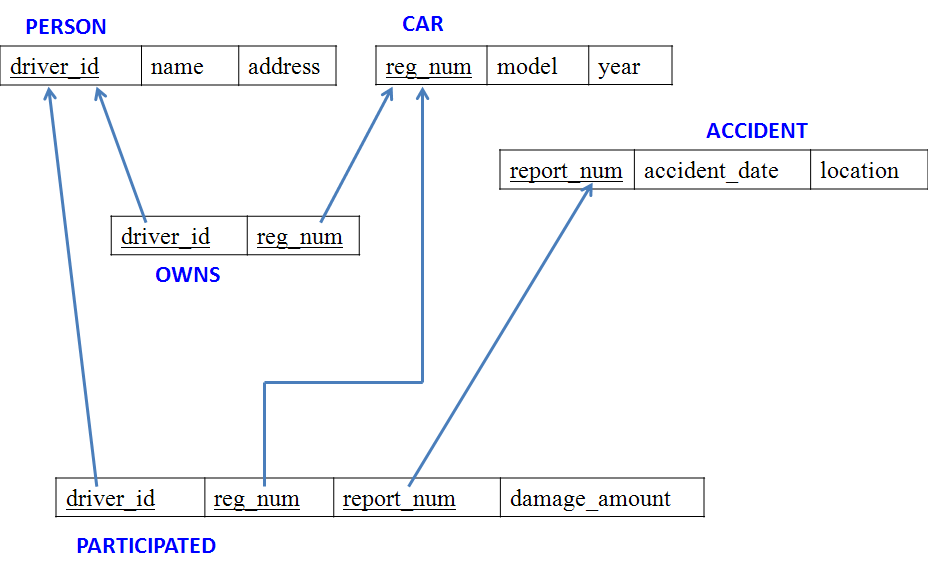
a.Update the damage amount to 25000 for the car with a specific reg-num(example 'K A053408') for which the accident report number was 12.

b.Add a new accident to the database.

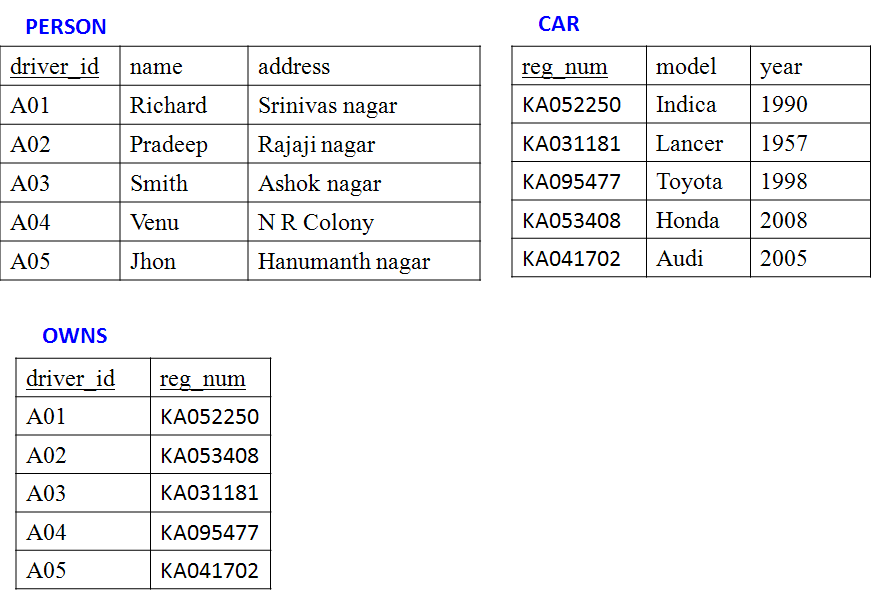
iv)Find the total number of people who owned cars that involved in accidents in 2008.

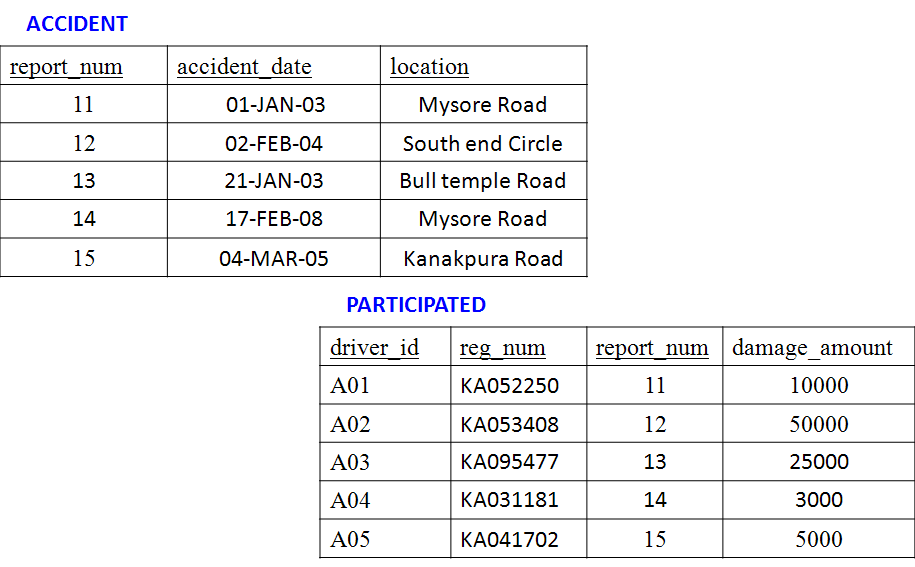
v)Find the number of accidents in which cars belonging to a specific model (example )were involved.

**Schema diagram**



**Tables**





create database insurance;

use insurance;

create table person(

driver\_id varchar(10),

name varchar(20),

address varchar(30),

primary key(driver\_id)

);

desc person;

create table car(

reg\_num varchar(10),

model varchar(10),

year int,

primary key(reg\_num)

);

desc car;

create table accident(

report\_num int,

accident\_date date,

location varchar(20),

primary key(report\_num)

);

desc accident;

create table owns(

driver\_id varchar(10),

reg\_num varchar(10),

primary key(driver\_id,reg\_num),

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

foreign key(reg\_num) references car(reg\_num)

);

desc owns;

create table participated(

driver\_id varchar(10),

reg\_num varchar(10),

report\_num int,

damage\_amount int,

primary key(driver\_id,reg\_num,report\_num),

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

foreign key(reg\_num) references car(reg\_num),

foreign key(report\_num) references accident(report\_num)

);

desc participated;

insert into person values('A01','Raghu','Electronic City');

insert into person values('A02','Rishab','Orange County');

insert into person values('A03','Rufus','NR Colony');

insert into person values('A04','Jamal','Lawrence Park');

insert into person values('A05','Kevin','Rosedale');

commit;

select \* from person;

insert into car values('KA031111','Accord',2005);

insert into car values('KA041122','MX-5',2019);

insert into car values('KA051133','Indica',2010);

insert into car values('KA061144','Prius',2015);

insert into car values('KA071155','Camry',2020);

insert into car values('KA01010', 'Accord', 2002);

commit;

select \* from car;

insert into accident values(111,'2020-01-01','NR Road');

insert into accident values(122,'2020-02-02','Dalhousie Road');

insert into accident values(133,'2020-03-03','Henry Road');

insert into accident values(144,'2020-04-04','Beehive Road');

insert into accident values(155,'2020-05-05','Orange Street');

insert into accident values(200, '2008-12-01', 'Pinto Road');

commit;

select \* from accident;

insert into owns values ('A01','KA031111');

insert into owns values ('A02','KA041122');

insert into owns values ('A03','KA051133');

insert into owns values ('A04','KA061144');

insert into owns values ('A05','KA071155');

insert into owns values('A02', 'KA01010');

commit;

select \* from owns;

insert into participated values ('A01','KA031111',111, 10000);

insert into participated values ('A02','KA041122',122, 20000);

insert into participated values ('A03','KA051133',133, 30000);

insert into participated values ('A04','KA061144',144, 40000);

insert into participated values ('A05','KA071155',155, 50000);

insert into participated values('A02', 'KA01010', 200, 500);

commit;

select \* from participated;

-- Query 3a

update participated

set damage\_amount = 2500

where reg\_num='KA031111';

select \* from participated;

-- Query 3b

insert into accident values(101,'2020-12-01','Xavier Road');

insert into participated values('A01','KA031111',101, 1001);

commit;

select \* from accident;

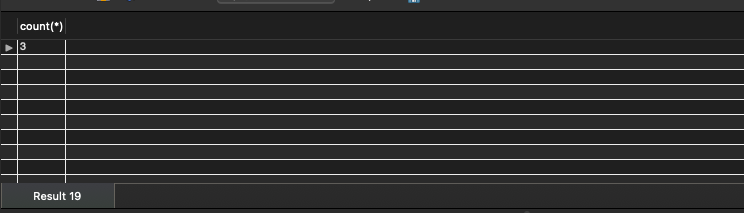
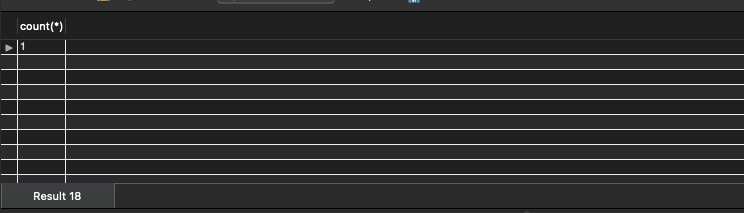
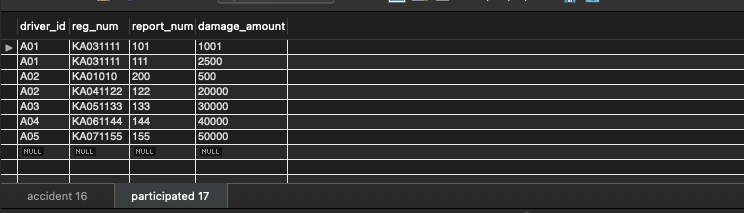
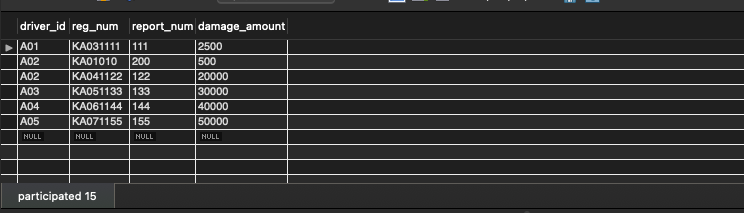
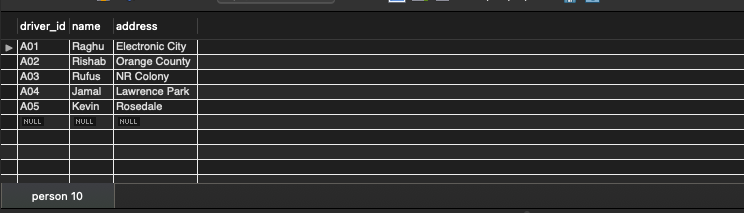
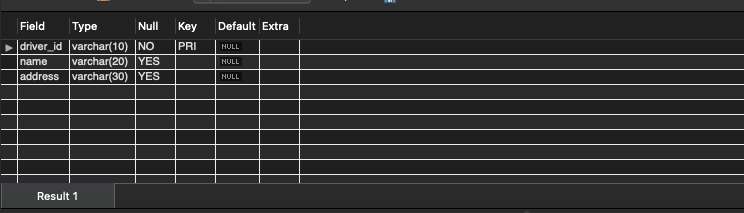
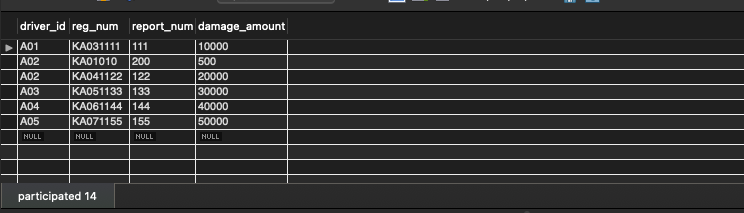
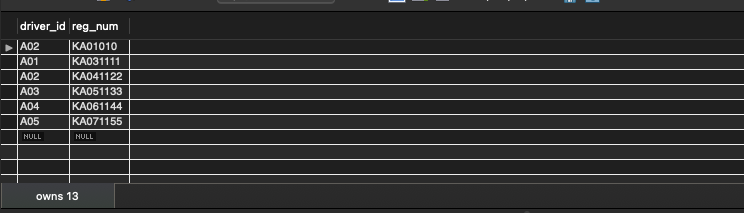
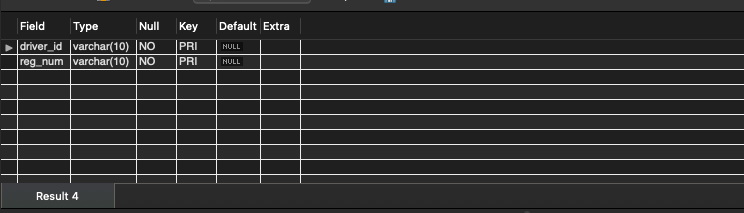
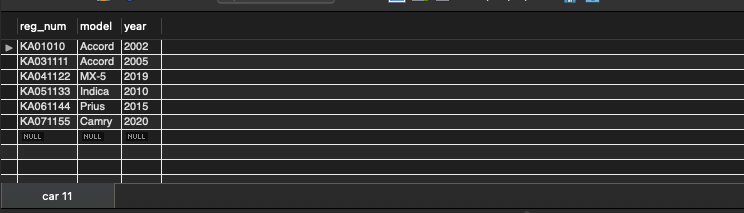
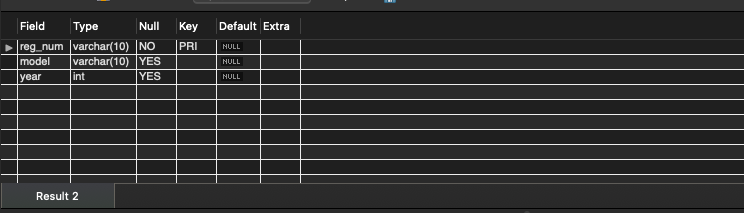
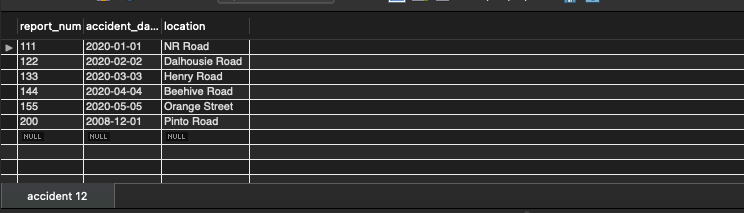
select \* from participated;

-- Query 4

select count(\*) from accident where year(accident\_date)=2008;

-- Query 5

select count(\*) from participated where reg\_num in ( select reg\_num from car where model="Accord");



**PROGRAM 2: BANKING ENTERPRISE DATABASE**

Consider the following database for a banking enterprise.

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

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

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

**Depositer**(customer-name: String, accno: int)

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

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

foreign keys.

ii. Enter at least five tuples for each relation.

iii. Find all the customers who have at least two accounts at the *Main* branch (ex. SBI\_ResidencyRoad).

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

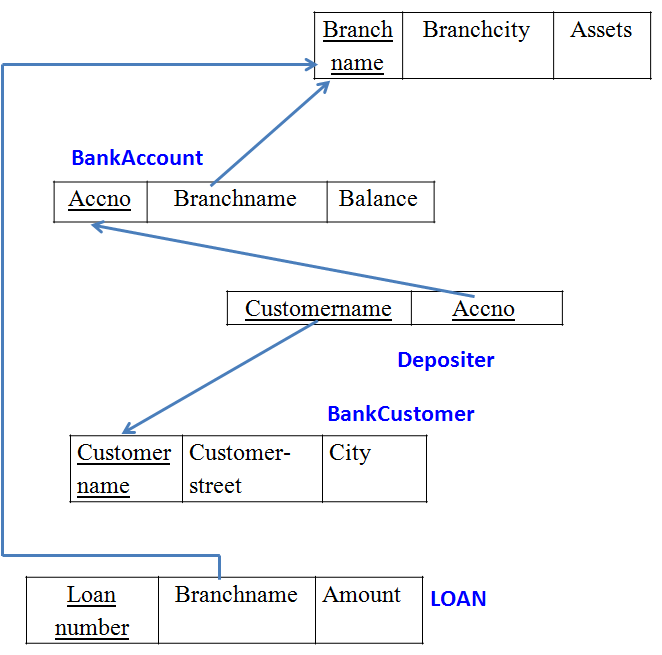
specific city (Ex. Delhi).

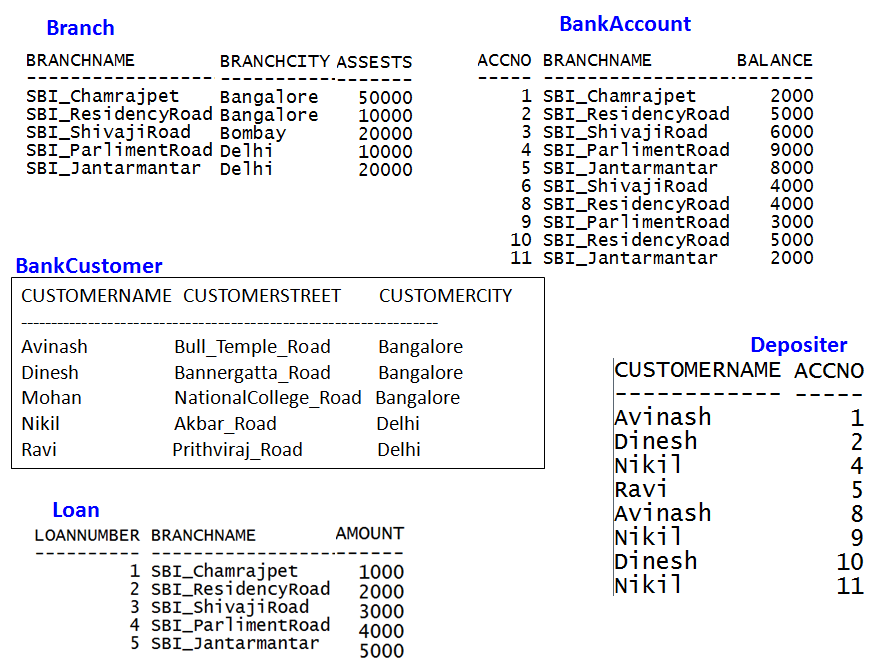
v. Demonstrate how you delete all account tuples at every branch located in

a specific city (Ex. Bombay).

**INTRODUCTION:** This database is developed for supporting banking facilities. Details of the branch along with the accounts and loans handled by them are recorded. Also details of the depositors of the corresponding branches are maintained.

**Schema Diagram**





create database bank;

use bank;

create table branch (

branch\_name varchar(25),

branch\_city varchar(15),

assets int,

primary key (branch\_name)

);

create table bank\_account (

accno int,

branch\_name varchar(25),

balance int,

primary key (accno),

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

);

create table bank\_customer (

customer\_name varchar(10),

customer\_street varchar(25),

customer\_city varchar(15),

primary key (customer\_name)

);

create table depositer (

customer\_name varchar(10),

accno int,

primary key(customer\_name, accno),

foreign key (customer\_name) references bank\_customer(customer\_name),

foreign key (accno) references bank\_account(accno)

);

create table loan (

loan\_number int,

branch\_name varchar(25),

amount int,

primary key (loan\_number),

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

);

insert into branch values('SBI\_Chamrajpet', 'Bangalore', 50000);

insert into branch values('SBI\_ResidencyRoad', 'Bangalore', 10000);

insert into branch values('SBI\_ShivajiRoad', 'Bombay', 20000);

insert into branch values('SBI\_ParliamentRoad', 'Delhi', 10000);

insert into branch values('SBI\_Jantarmantar', 'Delhi', 20000);

commit;

insert into bank\_account values(1, 'SBI\_Chamrajpet', 2000);

insert into bank\_account values(2, 'SBI\_ResidencyRoad', 5000);

insert into bank\_account values(3, 'SBI\_ShivajiRoad', 6000);

insert into bank\_account values(4, 'SBI\_ParliamentRoad', 9000);

insert into bank\_account values(5, 'SBI\_Jantarmantar', 8000);

insert into bank\_account values(6, 'SBI\_ShivajiRoad', 4000);

insert into bank\_account values(8, 'SBI\_ResidencyRoad', 4000);

insert into bank\_account values(9, 'SBI\_ParliamentRoad', 3000);

insert into bank\_account values(10, 'SBI\_ResidencyRoad', 5000);

insert into bank\_account values(11, 'SBI\_Jantarmantar', 2000);

commit;

insert into bank\_customer values ('Avinash', 'Bull\_Temple\_Road', 'Bangalore');

insert into bank\_customer values ('Dinesh', 'Bannergatta\_Road', 'Bangalore');

insert into bank\_customer values ('Mohan', 'National\_College\_Road', 'Bangalore');

insert into bank\_customer values ('Nikhil', 'Akbar\_Road', 'Delhi');

insert into bank\_customer values ('Ravi', 'Prithviraj\_Road', 'Delhi');

commit;

insert into depositer values('Avinash', 1);

insert into depositer values('Dinesh', 2);

insert into depositer values('Nikhil', 4);

insert into depositer values('Ravi', 5);

insert into depositer values('Avinash', 8);

insert into depositer values('Nikhil', 9);

insert into depositer values('Dinesh', 10);

insert into depositer values('Nikhil', 11);

commit;

insert into loan values(1, 'SBI\_Chamrajpet', 1000);

insert into loan values(2, 'SBI\_ResidencyRoad', 2000);

insert into loan values(3, 'SBI\_ShivajiRoad', 3000);

insert into loan values(4, 'SBI\_ParliamentRoad', 4000);

insert into loan values(5, 'SBI\_Jantarmantar', 5000);

commit;

select \* from branch;

select \* from bank\_account;

select \* from bank\_customer;

select \* from depositer;

select \* from loan;

select distinct c.customer\_name from bank\_customer c,bank\_account b where exists(select d.customer\_name,count(d.customer\_name) from depositer d,bank\_account ba where ba.accno = d.accno and

c.customer\_name = d.customer\_name and ba.branch\_name = 'SBI\_ResidencyRoad' group by d.customer\_name having count(d.customer\_name)>=2);

select d.customer\_name from depositer d,branch b,bank\_account a

where b.branch\_name=a.branch\_name

AND a.accno=d.accno

and branch\_city='Delhi'

group by d.customer\_name

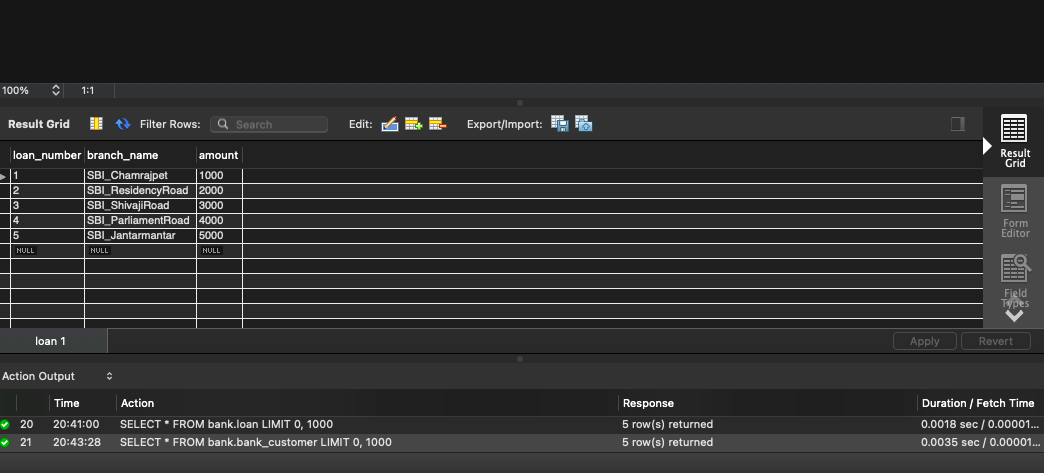
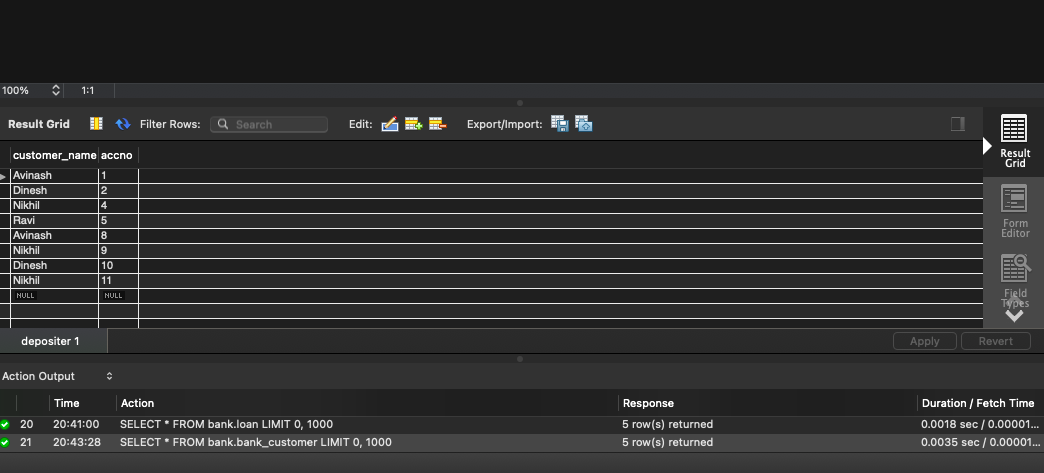
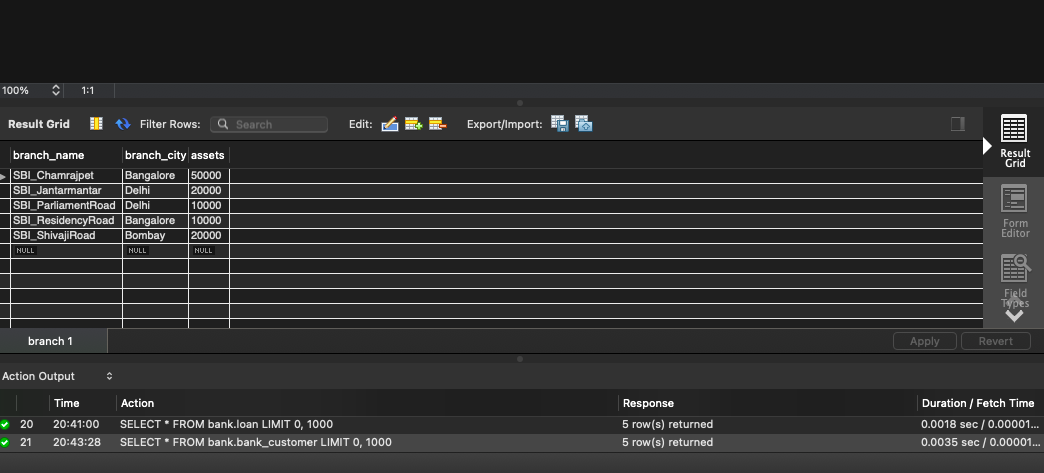
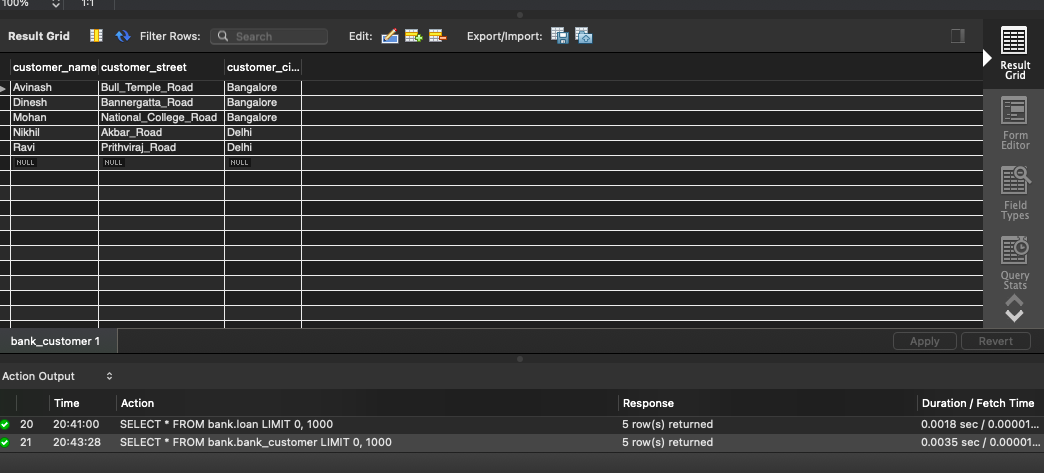
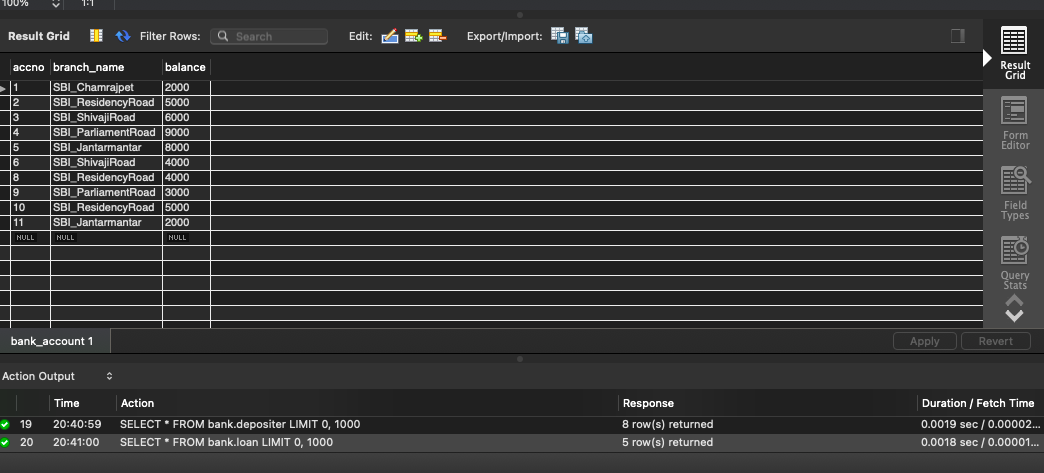
HAVING COUNT(distinct b.branch\_name)=(

SELECT COUNT(branch\_name)

FROM branch

WHERE branch\_city='Delhi');

delete from bank\_account where branch\_name in (select branch\_name from branch where branch\_city = 'Bombay');

**Sample Table data**

**PROGRAM 3: SUPPLIER DATABASE**

**Consider the following schema:**

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

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

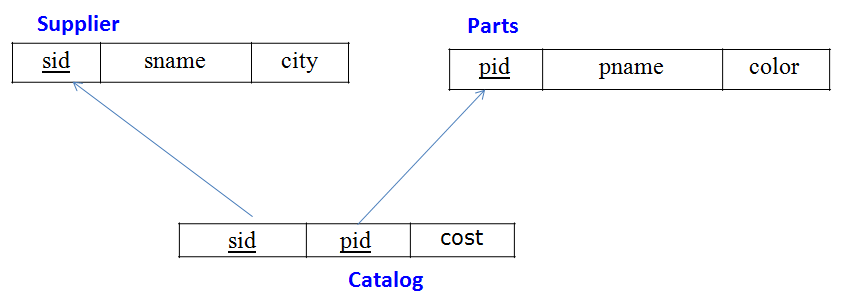
**CATALOG(sid: integer, pid: integer, cost: real)**

**The Catalog relation lists the prices charged for parts by Suppliers.**

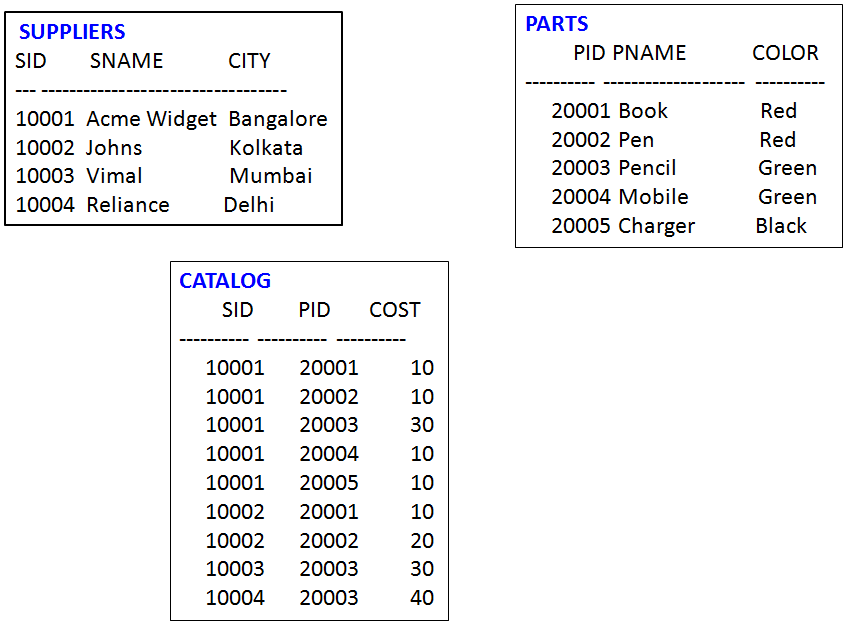
**Write the following queries in SQL:**

1. Find the pnames of parts for which there is some supplier.
2. Find the snames of suppliers who supply every part.
3. Find the snames of suppliers who supply every red part.
4. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
5. 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).
6. For each part, find the sname of the supplier who charges the most for that part.

**Schema Diagram**



**Table Data**

****

**create database supplier;**

**use supplier;**

**create table suppliers(**

**sid int primary key,**

**sname varchar(30),**

**address varchar(30)**

**);**

**create table parts(**

**pid int primary key,**

**pname varchar(30),**

**color varchar(30)**

**);**

**create table catalog (**

**sid int ,**

**pid int ,**

**cost real,**

**constraint c\_sid foreign key(sid) references suppliers(sid) ,**

**constraint c\_pid foreign key(pid) references parts(pid)**

**);**

**insert into suppliers values(1,'Acme Widget','kolkata') ;**

**insert into suppliers values(2,'Tata','bengaluru') ;**

**insert into suppliers values(3,'Reebok','delhi') ;**

**insert into suppliers values(4,'Nike','delhi') ;**

**insert into suppliers values(5,'Reliance','delhi') ;**

**insert into parts values(1,'paint','red') ;**

**insert into parts values(2,'steel','black') ;**

**insert into parts values(3,'spray','red') ;**

**insert into parts values(4,'sheet','green');**

**insert into parts values(5,'tiles','blue');**

**delete from parts where pid=5;**

**insert into catalog values(1,1,100);**

**insert into catalog values(1,2,200);**

**insert into catalog values(1,3,200);**

**insert into catalog values(1,4,100);**

**insert into catalog values(2,1,300);**

**insert into catalog values(2,2,100);**

**insert into catalog values(3,2,90);**

**insert into catalog values(3,3,110);**

**insert into catalog values(3,4,110);**

**insert into catalog values(4,1,100);**

**insert into catalog values(4,3,120);**

**insert into catalog values(4,4,130);**

**select \* from catalog;**

**select \* from parts;**

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

**insert into parts values(5,'tiles','blue');**

**select p.pname from parts p where p.pid in (select pid from catalog c group by c.pid having count(c.sid)>0);**

**insert into catalog values(1,5,140);**

**select p.pname from parts p where p.pid in (select pid from catalog c group by c.pid having count(c.sid)>0);**

**delete from catalog where pid=5;**

**delete from parts where pid=5;**

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

**select s.sname from suppliers s where s.sid in (select c.sid from catalog c group by c.sid having count(distinct (c.pid))=(select count(p.pid) from parts p));**

**-- iii. Find the snames of suppliers who supply every red part.**

**select s.sname from suppliers s where s.sid in (select ca.sid from catalog ca,parts p where ca.pid=p.pid and p.color='red' group by ca.sid having count(ca.pid)=(select count(\*) from parts p where p.color='red'));**

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

**select ca.pid from catalog ca where ca.sid=(select s.sid from suppliers s where s.sname ='Acme Widget') having (select count(c.pid) from catalog c where c.pid=ca.pid)=1;**

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

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

**select s.sname from suppliers s where s.sid in (select c.sid from catalog c where c.cost=(select max(cost) from catalog ca where ca.pid=c.pid));**

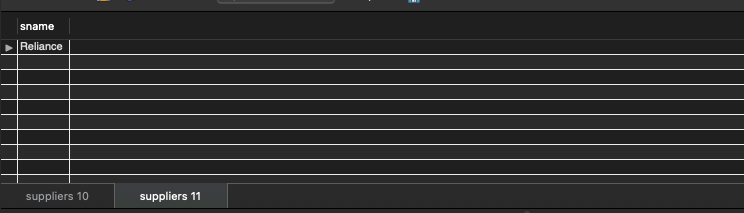
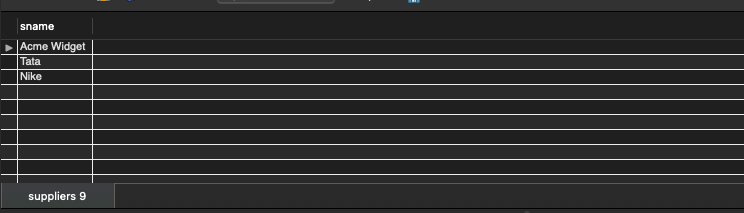
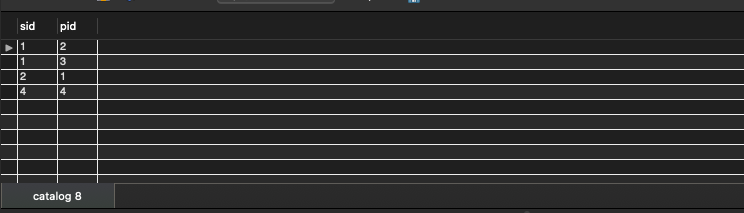
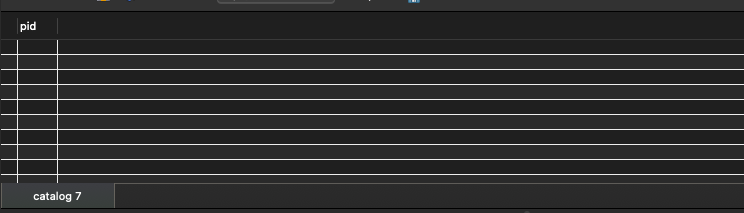
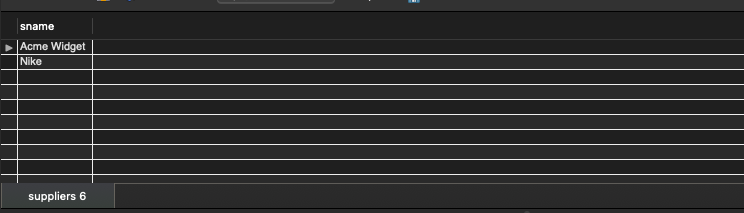
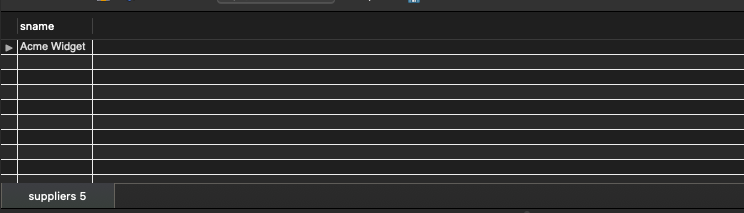
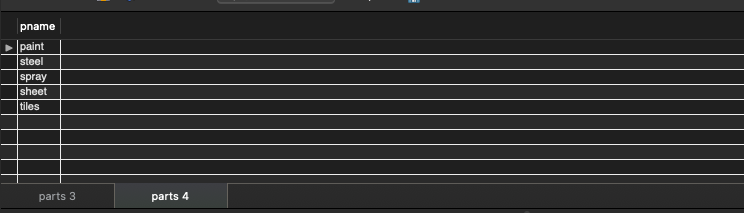
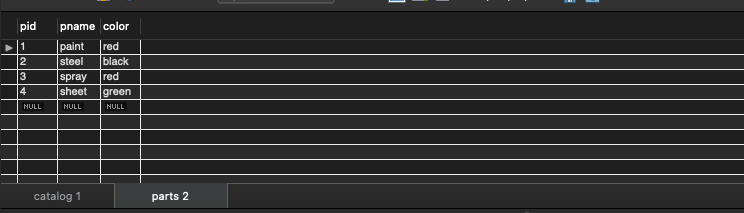
**-- vii. select supplier who sell only red parts**

**select s.sname from suppliers s where s.sid in(select c.sid from catalog c where c.sid not in (select distinct(ca.sid) from catalog ca,parts p where ca.pid=p.pid and p.color!='red'));**

**insert into catalog values(5,1,140);**

**select s.sname from suppliers s where s.sid in(select c.sid from catalog c where c.sid not in (select distinct(ca.sid) from catalog ca,parts p where ca.pid=p.pid and p.color!='red'));**

**delete from catalog where sid=5;**

****

**PROGRAM 4: STUDENT FACULTY DATABASE**

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

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

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

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

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

**The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Level(lvl) is a two character code with 4 different values (example: Junior: JR etc)**

**Write the following queries in SQL. No duplicates should be printed in any of the answers.**

1. Find the names of all Juniors (level = JR) who are enrolled in a class taught by
2. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.
3. Find the names of all students who are enrolled in two classes that meet at the same time.
4. Find the names of faculty members who teach in every room in which some class is taught.
5. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.
6. Find the names of students who are not enrolled in any class.
7. 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).

create database studentdb;

use studentdb;

create table student(

snum int,

sname varchar(10),

major varchar(2),

lvl varchar(2),

age int,

primary key(snum)

);

desc student;

create table class(

cname varchar(8),

meetsat time,

room varchar(5),

fid int,

primary key (cname)

);

desc class;

create table enrolled(

snum int,

cname varchar(10),

primary key (snum, cname),

foreign key (snum) references student(snum),

foreign key (cname) references class(cname)

);

desc enrolled;

create table faculty(

fid int,

fname varchar(10),

deptid int,

primary key (fid)

);

desc faculty;

insert into student values(1,'John', 'CS','Jr',19);

insert into student values(2,'Smith', 'CS','Jr',20);

insert into student values(3,'Jacob', 'CV','Sr',20);

insert into student values(4,'Tom', 'CS','Jr',20);

insert into student values(5,'Rahul', 'CS','Jr',20);

insert into student values(6,'Rita', 'CS','Sr',21);

insert into student values(7,'McGregor', 'CV','Sr',22);

insert into student values(8,'Smilga', 'CS','Jr',19);

insert into student values(9,'Price', 'CV','Sr',22);

commit;

insert into faculty values (11, 'Harish', 1000);

insert into faculty values (12, 'MV', 1000);

insert into faculty values (13, 'Mira', 1001);

insert into faculty values (14, 'Shiva', 1002);

insert into faculty values (15, 'Nupur', 1000);

commit;

insert into class values('Class1', '10:15:15' , 'R1', 14);

insert into class values('Class10', '10:15:16' , 'R128', 11);

insert into class values('Class11', '10:15:16' , 'R1', 11);

insert into class values('Class3', '10:15:16' , 'R3', 11);

insert into class values('Class13', '10:15:16' , 'R2', 11);

insert into class values('Class12', '10:15:16' , 'R4', 11);

insert into class values('Class2', '10:15:20' , 'R2', 12);

insert into class values('Class3', '10:15:45' , 'R3', 11);

insert into class values('Class4', '10:15:20' , 'R4', 12);

insert into class values('Class5', '20:15:20' , 'R3', 15);

insert into class values('Class6', '13:20:20' , 'R2', 12);

insert into class values('Class7', '10:10:10' , 'R3', 12);

commit;

select \* from class;

insert into enrolled values(1, 'Class1');

insert into enrolled values(2, 'Class1');

insert into enrolled values(6, 'Class1');

insert into enrolled values(7, 'Class1');

insert into enrolled values(8, 'Class1');

insert into enrolled values(3, 'Class3');

insert into enrolled values(4, 'Class2');

insert into enrolled values(5, 'Class4');

commit;

select \* from student;

select \* from faculty;

select \* from class;

select \* from enrolled;

-- Query 1

select sname from student where lvl='Jr' and snum in

(select snum from enrolled where cname in

(select cname from class where fid in

(select fid from faculty where fname='Shiva')

));

-- Query 2

select cname from class where cname in(

select cname from class where room = 'R128') or cname in

(select cname from enrolled group by cname having count(cname)>=5);

-- Query 3

select sname from student where snum in(

select snum from enrolled where cname in(

select cname from class where meetsat in (select meetsat from class group by meetsat having count(meetsat)>1)));

-- Query 4

SELECT f.fname,f.fid

FROM faculty f

WHERE f.fid in ( SELECT fid FROM class

GROUP BY fid HAVING COUNT(\*)=(SELECT COUNT(DISTINCT room) FROM class) );

-- Query 5

select distinct fid from class where cname in (select cname from enrolled group by cname having count(cname)<5) or cname not in (select distinct cname from enrolled);

-- Query 6

select sname from student where snum not in (select distinct snum from enrolled);

-- Query 7

SELECT S.age, S.lvl

FROM student S

GROUP BY S.age, S.lvl

HAVING S.lvl IN(SELECT S1.lvl

FROM student S1

WHERE S1.age=S.age

GROUP BY S1.age, S1.lvl

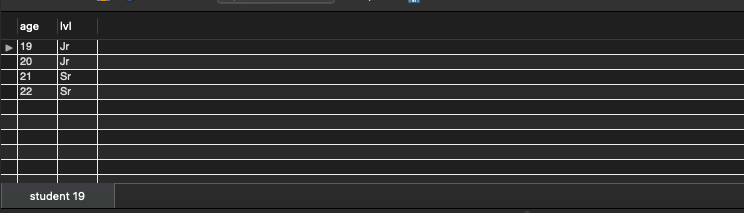
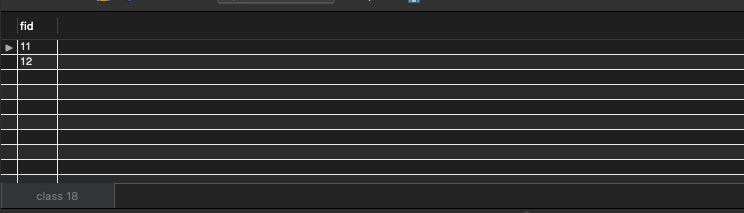
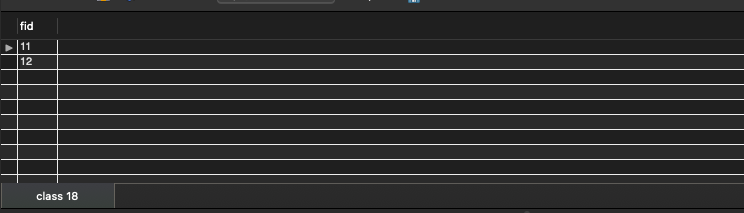
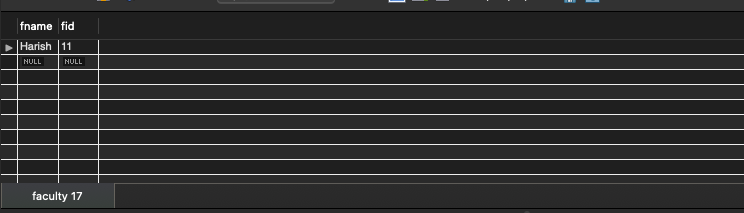
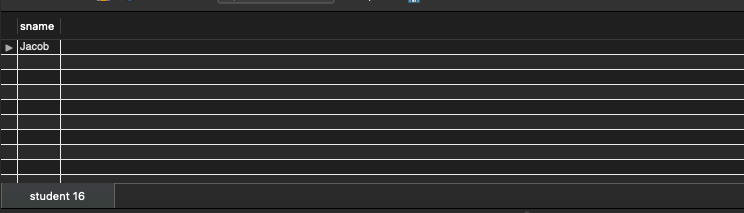
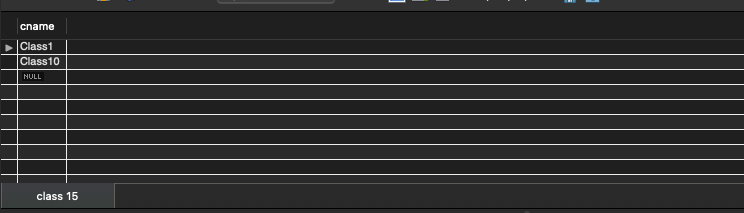
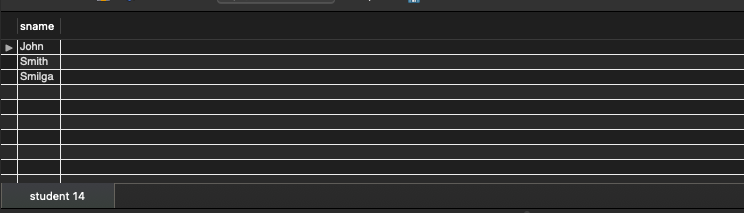
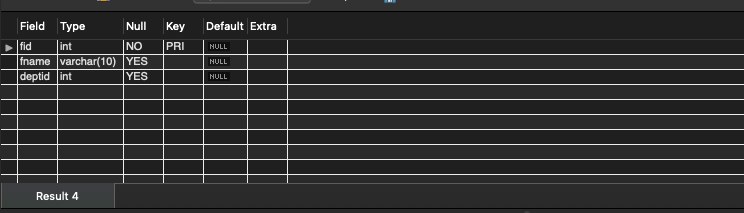
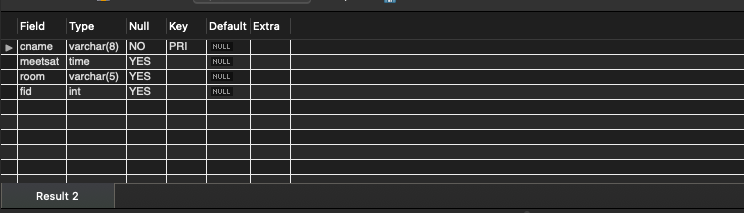
HAVING COUNT(\*) >= ALL (SELECT COUNT(\*)

FROM student S2

WHERE S1.age=S2.age

GROUP BY S2.lvl, S2.age))

ORDER BY S.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)**

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

**Note that the Employees relation describes pilots and other kinds of employees as well; Every pilot is certified for some aircraft, and only pilots are certified to fly.**

**Write each of the following queries in SQL.**

1. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.
2. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruisingrange of the aircraft for which she or he is certified.
3. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.
4. For all aircraft with cruisingrange over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft.
5. Find the names of pilots certified for some Boeing aircraft.
6. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.
7. 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.

create database flightdb;

use flightdb;

create table flights(

flno int,

fromplace varchar(15),

toplace varchar(15),

distance int,

departs datetime,

arrives datetime,

price int,

primary key (flno)

);

desc flights;

create table aircraft(

aid int,

aname varchar(15),

cruisingrange int,

primary key (aid)

);

desc aircraft;

create table employees (

eid int,

ename varchar(15),

salary int,

primary key (eid)

);

desc employees;

create table certified (

eid int,

aid int,

foreign key (eid) references employees(eid),

foreign key (aid) references aircraft(aid)

);

desc certified;

insert into flights values(101, 'Bangalore', 'Delhi', 2500, '2005-05-13 07:15:31', '2005-05-13 18:15:31', 5000);

insert into flights values(102, 'Bangalore', 'Lucknow', 3000, '2013-05-05 07:15:31', '2013-05-05 11:15:31', 6000);

insert into flights values(103, 'Lucknow', 'Delhi', 500, '2013-05-05 12:15:31', '2013-05-05 17:15:31', 3000);

insert into flights values(107, 'Bangalore', 'Frankfurt', 8000, '2013-05-05 07:15:31', '2013-05-05 22:15:31', 60000);

insert into flights values(104, 'Bangalore', 'Frankfurt', 8500, '2013-05-05 07:15:31', '2013-05-05 23:15:31', 75000);

insert into flights values(105, 'Kolkata', 'Delhi', 3400, '2013-05-05 07:15:31', '2013-05-05 09:15:31', 7000);

insert into flights values(106, 'Bangalore', 'Kolkata', 1000, '2013-05-05 01:15:30', '2013-05-05 09:20:30', 10000);

insert into flights values(108, 'Lucknow', 'Kolkata', 1000, '2013-05-05 11:30:30', '2013-05-05 15:20:30', 10000);

commit;

select \* from flights;

insert into aircraft values(101, '747', 3000);

insert into aircraft values(102, 'Boeing', 900);

insert into aircraft values(103, '647', 800);

insert into aircraft values(104, 'Dreamliner', 10000);

insert into aircraft values(105, 'Boeing', 3500);

insert into aircraft values(106, '707', 1500);

insert into aircraft values(107, 'Dream', 120000);

insert into aircraft values(108, '707', 760);

insert into aircraft values(109, '747', 1000);

commit;

select \* from aircraft;

insert into employees values(701, 'A', 50000);

insert into employees values(702, 'B', 100000);

insert into employees values(703, 'C', 150000);

insert into employees values(704, 'D', 90000);

insert into employees values(705, 'E', 40000);

insert into employees values(706, 'F', 60000);

insert into employees values(707, 'G', 90000);

commit;

select \* from employees;

insert into certified values(701, 101);

insert into certified values(701, 102);

insert into certified values(701, 106);

insert into certified values(701, 105);

insert into certified values(702, 104);

insert into certified values(703, 104);

insert into certified values(704, 104);

insert into certified values(702, 107);

insert into certified values(703, 107);

insert into certified values(704, 107);

insert into certified values(702, 101);

insert into certified values(702, 108);

insert into certified values(701, 109);

commit;

select \* from certified;

-- Query 1

select distinct a.aname from aircraft a where a.aid in (

select c.aid from certified c, employees e where

c.eid = e.eid and not exists(

select \* from employees e1 where e1.eid=e.eid and e1.salary<80000

)

);

-- Query 2

select max(a.cruisingrange), c.eid from certified c, aircraft a where c.aid = a.aid group by c.eid having count(c.eid)>3;

-- Query 3

select ename from employees where salary <(

select min(price) from flights where fromplace='Bangalore' and toplace='Frankfurt');

-- Query 4

select avg(e.salary), c.aid from certified c, employees e where c.aid in(

select aid from aircraft where cruisingrange>1000) and e.eid = c.eid group by c.aid;

-- Query 5

select ename from employees where eid in(

select eid from certified where aid in(

select aid from aircraft where aname = 'Boeing'));

-- Query 6

select aname from aircraft where cruisingrange > any (select distance from flights where fromplace='Bangalore' and toplace='Delhi');

-- Query 7

SELECT F.flno, F.departs

FROM flights F

WHERE F.flno IN ( ( SELECT F0.flno

FROM flights F0

WHERE F0.fromplace = 'Bangalore' AND F0.toplace = 'Kolkata'

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

UNION

( SELECT F0.flno

FROM flights F0, flights F1

WHERE F0.fromplace = 'Bangalore' AND F0.toplace <> 'Kolkata'

AND F0.toplace = F1.fromplace AND F1.toplace = 'Kolkata'

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.fromplace = 'Bangalore'

AND F0.toplace = F1.fromplace

AND F1.toplace = F2.fromplace

AND F2.toplace = 'Kolkata'

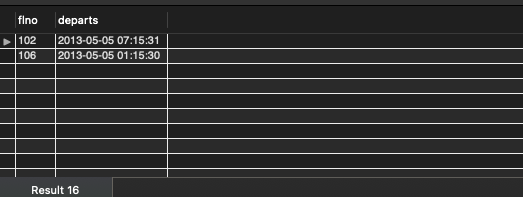
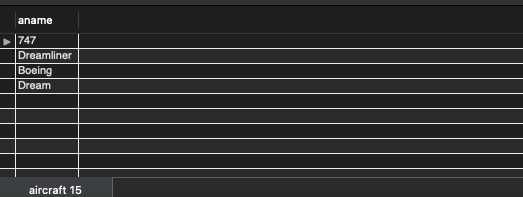
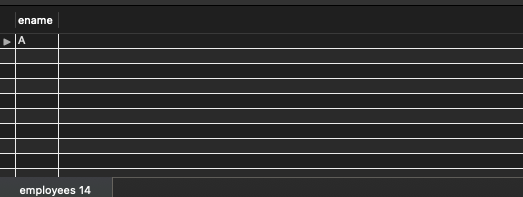
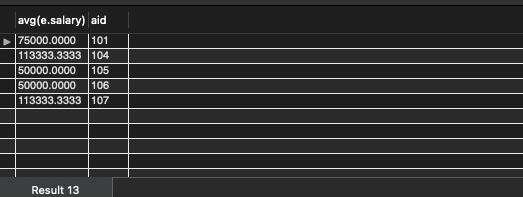
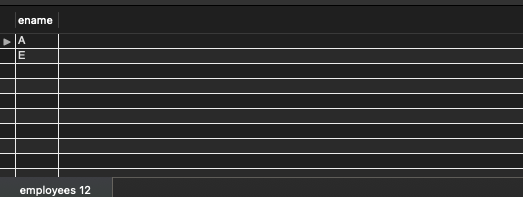
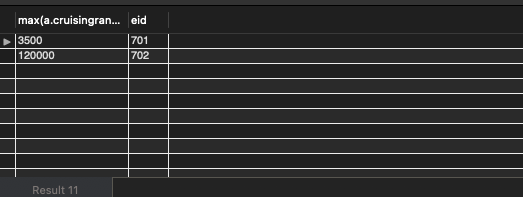
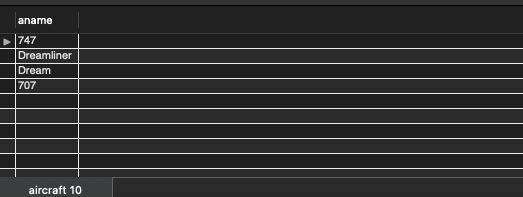
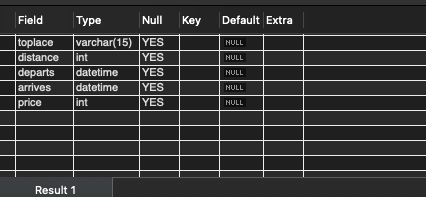
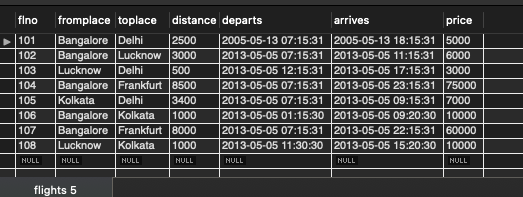
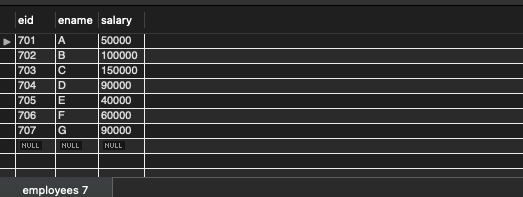
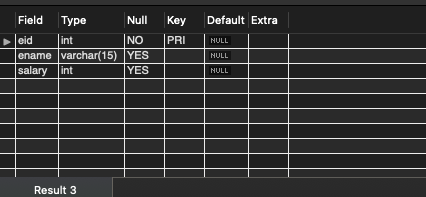
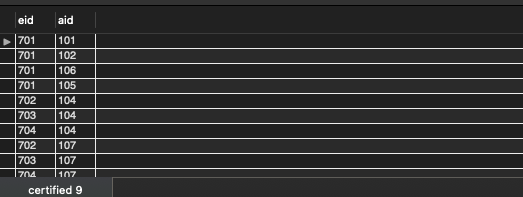
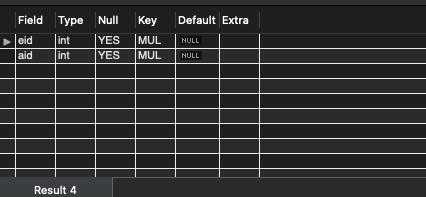
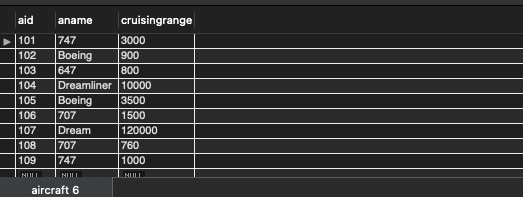
AND F0.toplace <> 'Kolkata'

AND F1.toplace <> 'Kolkata'

AND F1.departs > F0.arrives

AND F2.departs > F1.arrives

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



**Program 6 : Order Database**

**Consider the following schema for Order Database:**

**SALESMAN (*Salesman\_id, Name, City, Commission*)**

**CUSTOMER (*Customer\_id, Cust\_Name, City, Grade, Salesman\_id*)**

**ORDERS (*Ord\_No, Purchase\_Amt, Ord\_Date, Customer\_id, Salesman\_id*)**

**Write SQL queries to**

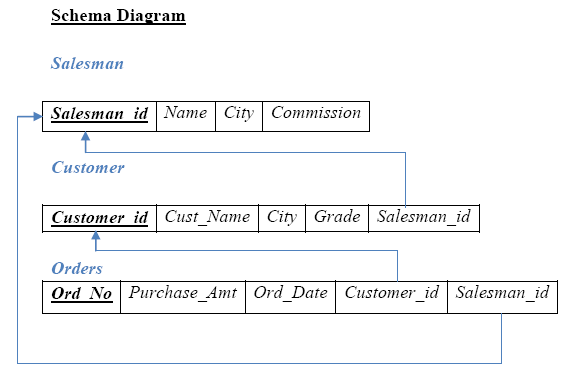
**1. Count the customers with grades above Bangalore’s average.**

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

**3. List all salesmen and indicate those who have and don’t have customers in their cities (Use UNION operation.)**

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

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



create database order\_processsing;

use order\_processsing;

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','hydrabad','15 %');

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);

insert into orders values (50, 5000, '04-05-17', 10, 1000);

insert into orders values (51, 450, '20-01-17', 10, 2000);

insert into orders values (52, 1000, '24-02-17', 13, 2000);

insert into orders values (53, 3500, '13-04-17', 14, 3000);

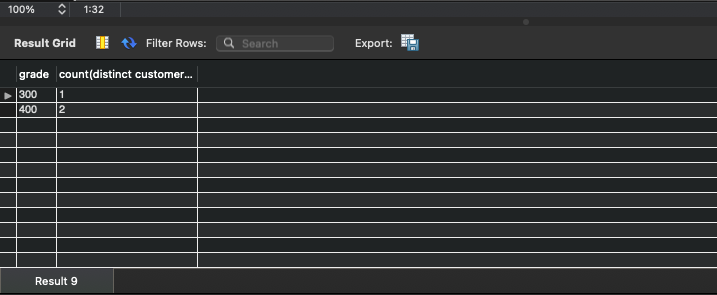
insert into orders values (54, 550, '09-03-17', 12, 2000);

select \* from salesman;

select \* from customer;

select \* from orders;

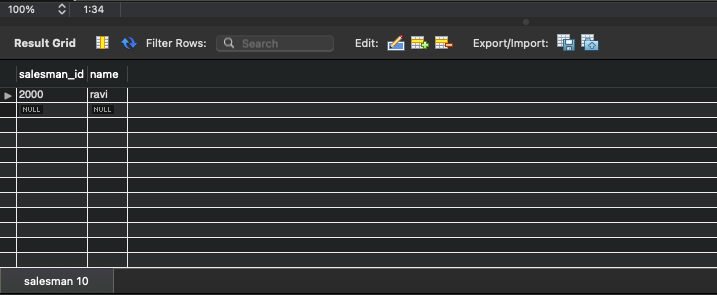
select grade, count(distinct customer\_id) from customer group by grade having grade > (select avg(grade) from customer where city='bangalore');



select salesman\_id, name from salesman a where 1 < (select count(\*) from customer where salesman\_id=a.salesman\_id);

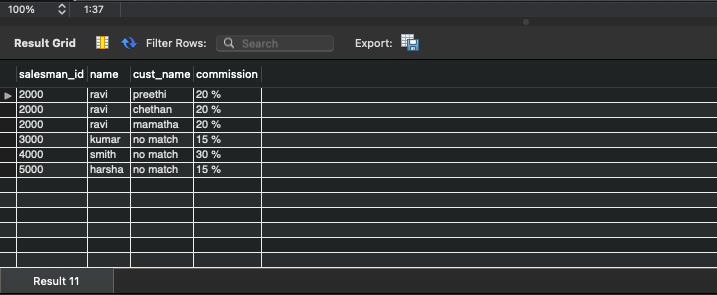
select salesman.salesman\_id, name, cust\_name, commission from salesman, customer where salesman.city = customer.city union

select salesman\_id, name, 'no match', commission from salesman where not city = any (select city from customer);



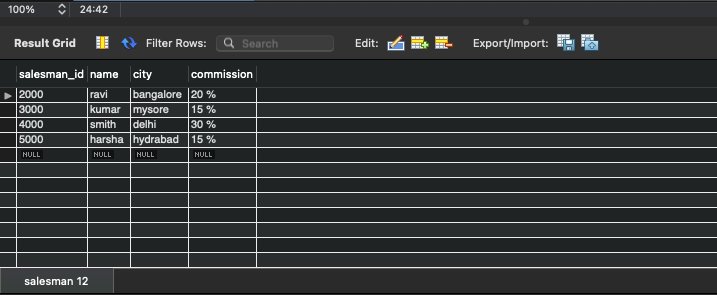
create view salesman\_view as select b.ord\_date, a.salesman\_id, a.name from salesman a, orders b where a.salesman\_id = b.salesman\_id and b.purchase\_amt=(select max(purchase\_amt) from orders c where c.ord\_date = b.ord\_date);

select \* from salesman\_view;



delete from salesman where salesman\_id=1000;

select \* from salesman;



**Program 7 : Book Database**

BOOK (Book\_id, Title, Publisher\_Name, Pub\_Year)

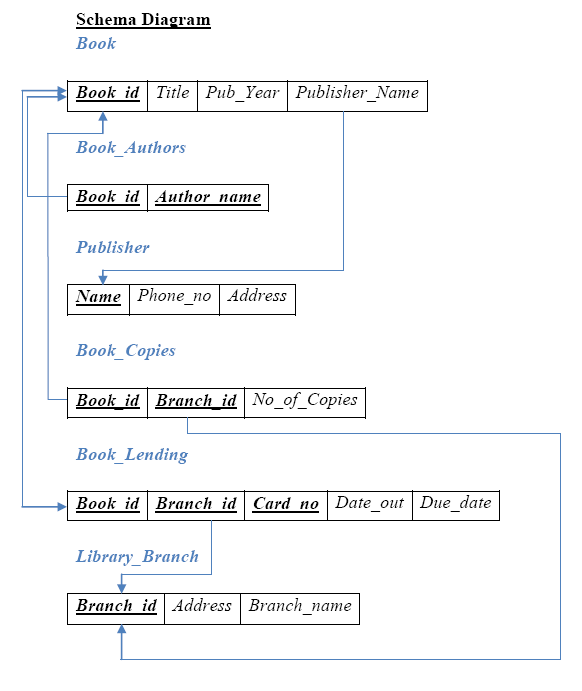
BOOK\_AUTHORS (Book\_id, Author\_Name)

PUBLISHER (Name, Address, Phone)

BOOK\_COPIES (Book\_id, Branch\_id, No-of\_Copies)

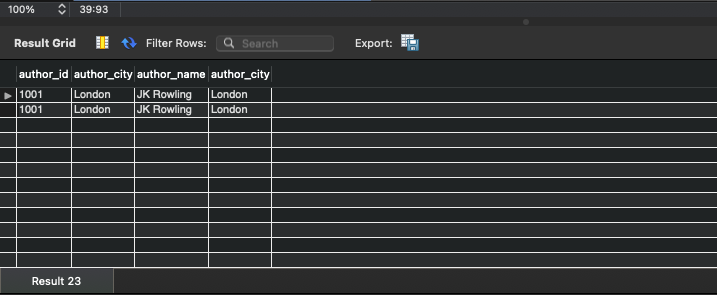
BOOK\_LENDING (Book\_id, Branch\_id, Card\_No, Date\_Out, Due\_Date)

LIBRARY\_BRANCH (Branch\_id, Branch\_Name, Address)



**Write SQL queries to**

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.

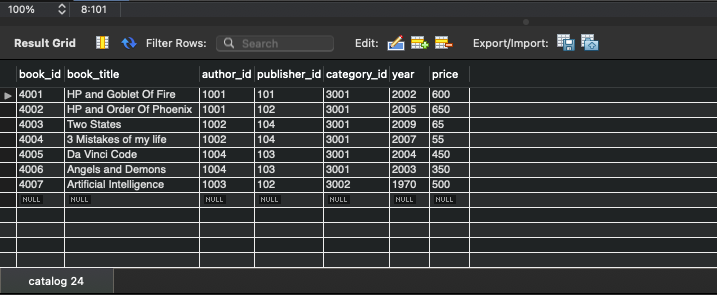


Get the particulars of borrowers who have borrowed more than 3 books, but            from Jan 2017 to Jun 2017

Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.

Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.

Create a view of all books and its number of copies that are currently available in the Library.



**Program 8:**

Consider the following database of student enrollment in courses & books adopted for each course.

STUDENT (regno: string, name: string, major: string, bdate:date)

COURSE (course #:int, cname:string, dept:string)

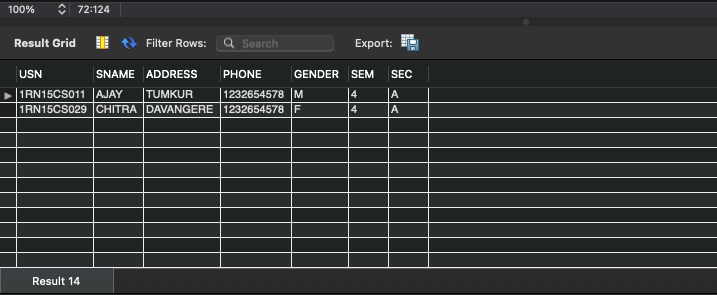
ENROLL ( regno:string, course#:int, sem:int, marks:int)

BOOK \_ ADOPTION (course# :int, sem:int, book-ISBN:int)

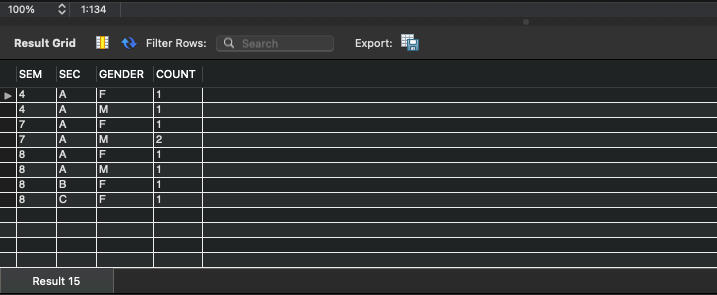
TEXT (book-ISBN:int, book-title:string, publisher:string, author:string)

Database applications laboratory GCEM DEPARTMENT OF CSE Page - 5 - 5th semester

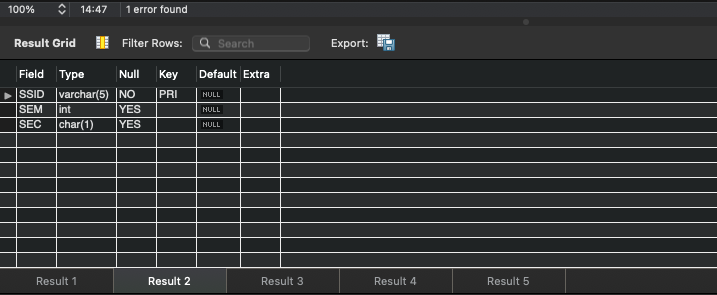
i. Create the above tables by properly specifying the primary keys and the foreign keys.



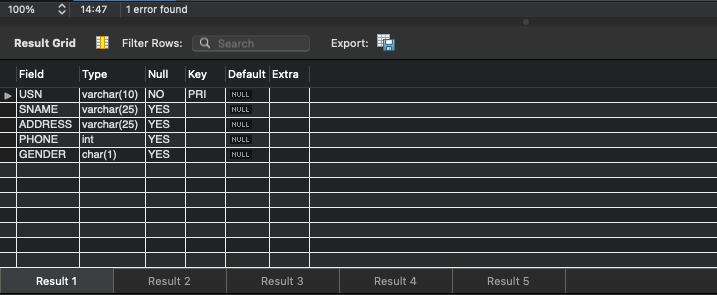
ii. Enter at least five tuples for each relation.



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

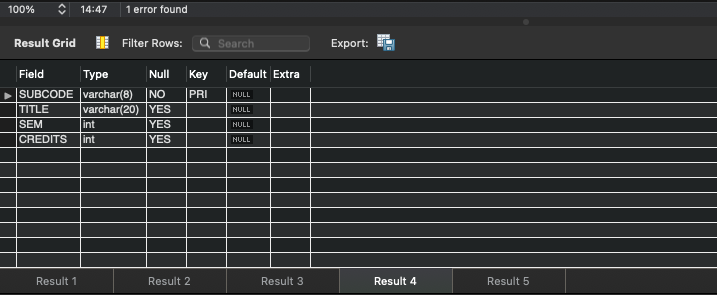


iv. Produce a list of text books (include Course #, Book-ISBN, Book-title) in the alphabetical order for courses offered by the ‘CS’ department that use more than two books.

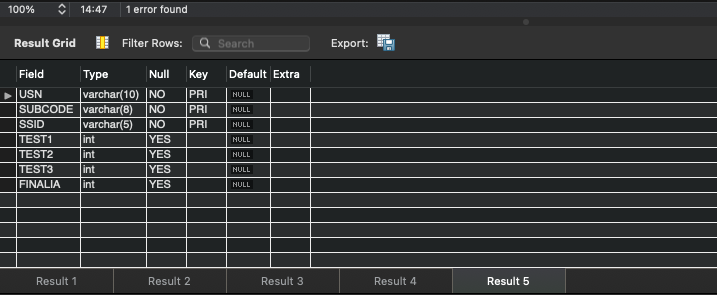


v. List any department that has all its adopted books published by a specific publisher.

vi. Generate suitable reports.



vii. Create suitable front end for querying and displaying the results.



**Program 9: Movie database**

**Consider the schema for Movie Database:**

**ACTOR (*Act\_id, Act\_Name, Act\_Gender*)**

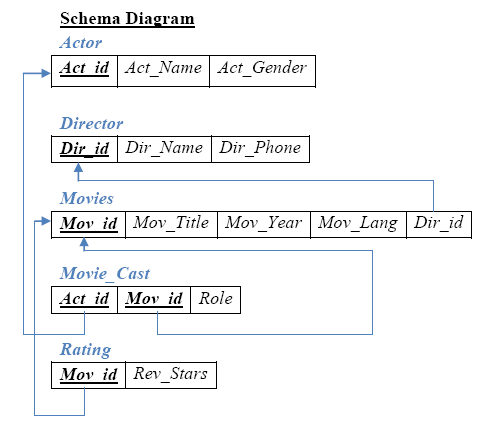
**DIRECTOR (*Dir\_id, Dir\_Name, Dir\_Phone*)**

**MOVIES (*Mov\_id, Mov\_Title, Mov\_Year, Mov\_Lang, Dir\_id*)**

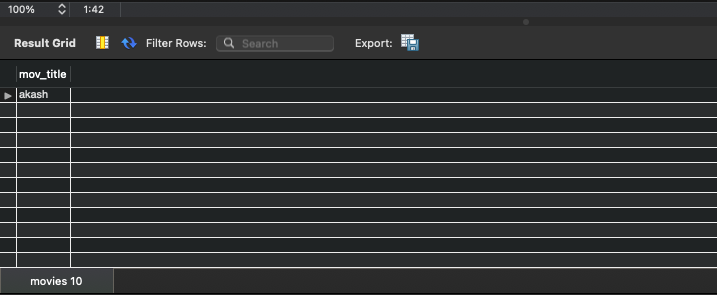
**MOVIE\_CAST (*Act\_id, Mov\_id, Role*)**

**RATING (*Mov\_id, Rev\_Stars*)**

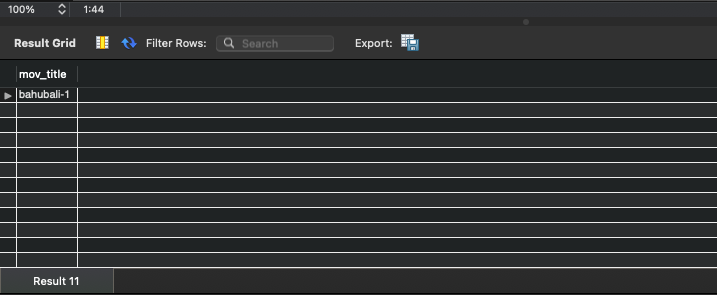
**Write SQL queries to**



**1. List the titles of all movies directed by ‘Hitchcock’.**

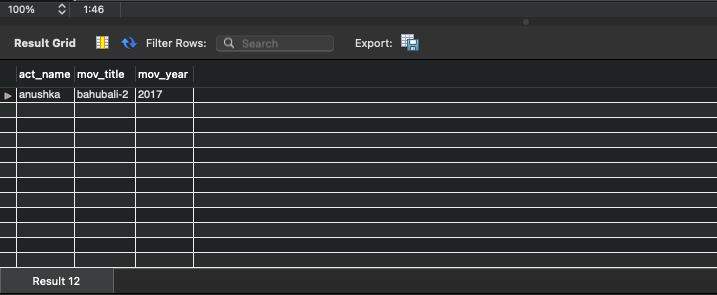


**2. Find the movie names where one or more actors acted in two or more movies.**

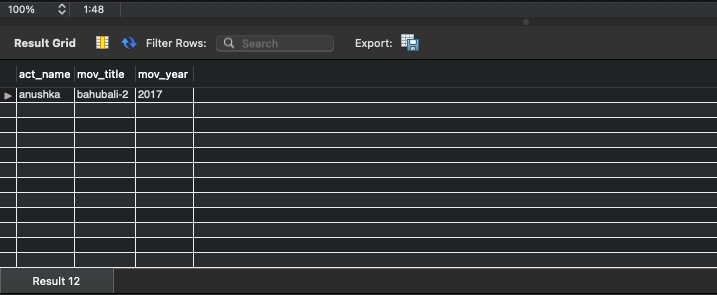
****

**3. List all actors who acted in a movie before 2000 and also in a movie after**

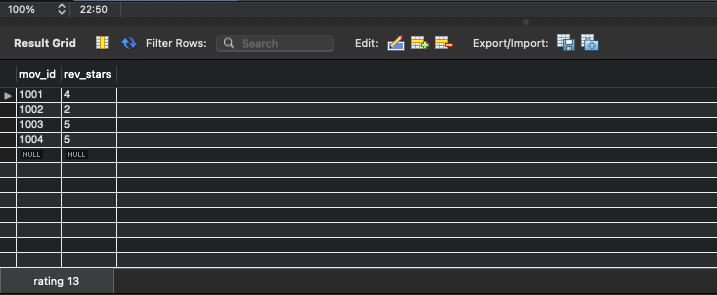
**2015 (use JOIN operation).**



**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.**

****

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

****

**Program 10**

**Consider the schema for College Database:**

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

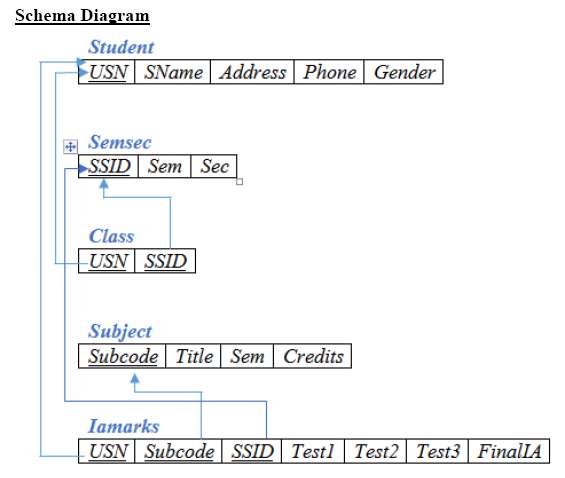
**SEMSEC (*SSID, Sem, Sec*)**

**CLASS (*USN, SSID*)**

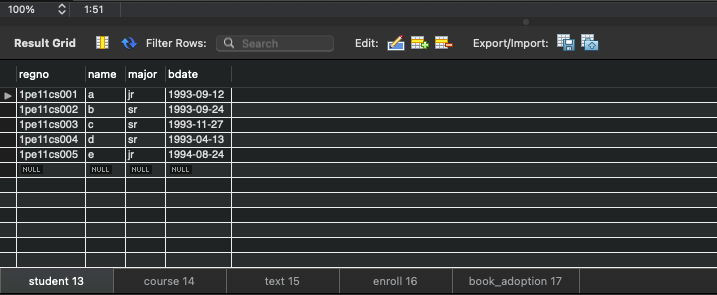
**SUBJECT (*Subcode, Title, Sem, Credits*)**

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

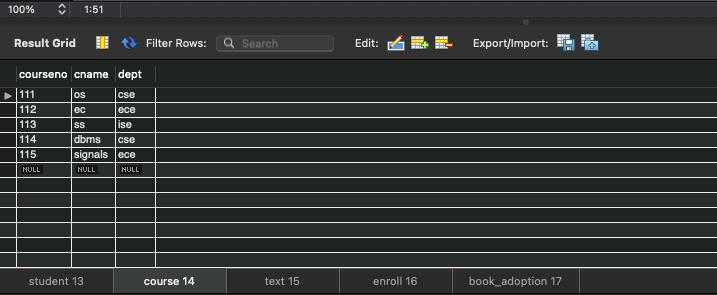
**Write SQL queries to**



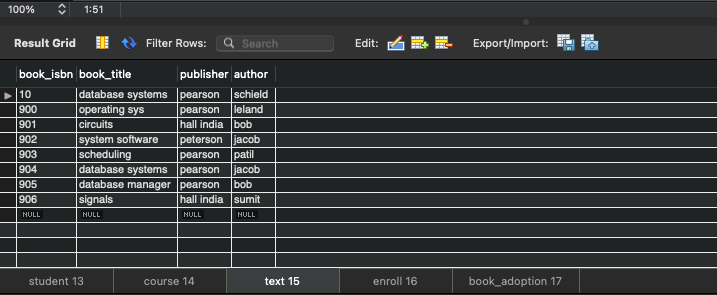
**1. List all the student details studying in fourth semester ‘C’ section.**

****

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

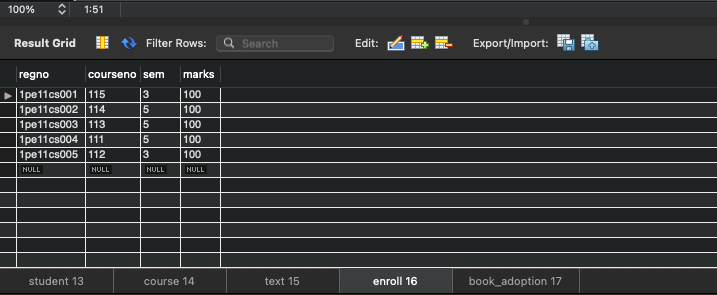
****

**3. Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.**

****

**4. Categorize students based on the following criterion:**

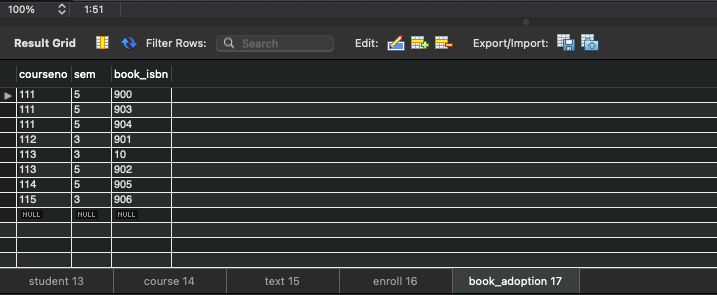
**If FinalIA = 17 to 20 then CAT = ‘Outstanding’**

****

**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.**

****