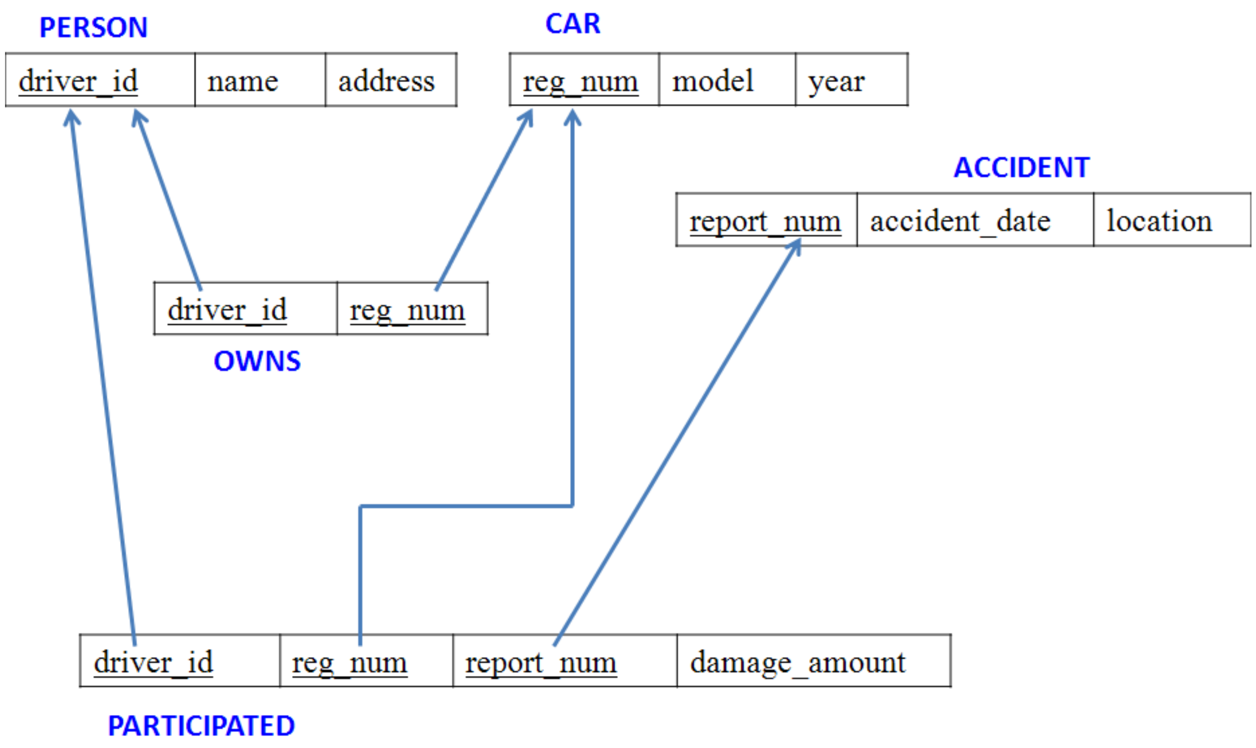
Medha Madhusudhan
1BM19CS230
CSE-4A
DBMS Record for Lab Test 1

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)

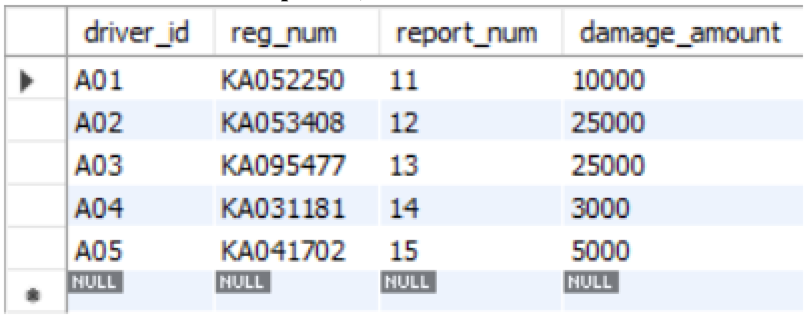
i) Create the above tables by properly specifying the primary keys and the foreign keys.
ii)Enter at least five tuples for each relation.
iii)Demonstrate how you
    a.Update the damage amount to 25000 for the car with a specific reg-num(example 'K A053408') for which the accident report number was 12.
b.Add a new accident to the database.
iv)Find the total number of people who owned cars that were involved in accidents in 2008.
v)Find the number of accidents in which cars belonging to a specific model (example)were involved.



```
create database Insurance;
use Insurance;

create table Person(
        driver_id varchar(20) not null,
        driver_name varchar(20) not null,
        address varchar(20) not null,
        primary key(driver_id)
);

create table Car(
        reg_num varchar(20) not null,
        model varchar(20) not null,
        year_purchase int not null,
        primary key(reg_num)
);

create table Accident(
        report_num int not null,
        accident_date date not null,
        location varchar(20) not null,
        primary key(report_num)
);

create table Owns(
        driver_id varchar(20) not null,
        reg_num varchar(20) not null,
        primary key(driver_id,reg_num),
        foreign key(driver_id) references Person(driver_id),
        foreign key(reg_num) references Car(reg_num)
);

create table Participated(
        driver_id varchar(20) not null,
        reg_num varchar(20) not null,
        report_num int not null,
        damage_amount int not null,
        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)
);

insert into Person(driver_id,driver_name,address)
        values ('A01','Richard','Srinivas Nagar'),
                ('A02','Pradeep','Rajajinagar'),
                ('A03','Smith','Ashok Nagar'),
                ('A04','Venu','NR Colony'),
                ('A05','Jhon','Hanumanth Nagar');

insert into Car(reg_num,model,year_purchase)
        values ('KA052250','Indica',1990),
                ('KA031181','Lancer',1957),
                ('KA095477','Toyota',1998),
                ('KA053408','Honda',2008),
                ('KA041702','Audi',2005);

insert into Owns(driver_id,reg_num)
        values ('A01','KA052250'),
                ('A02','KA053408'),
                ('A03','KA031181'),
                ('A04','KA095477'),
                ('A05','KA041702');

insert into Accident(report_num,accident_date,location)
        values (11,'2003-01-01','Mysore Road'),
                (12,'2004-02-02','South End Circle'),
                (13,'2003-01-21','Bull Temple Road'),
                (14,'2008-02-17','Mysore Road'),
                (15,'2005-03-04','Kanakpura Road');

insert into Participated(driver_id,reg_num,report_num,damage_amount)
        values ('A01','KA052250',11,10000),
                ('A02','KA053408',12,50000),
                ('A03','KA095477',13,25000),
                ('A04','KA031181',14,3000),
                ('A05','KA041702',15,5000);
```

-- demonstrate how you update damage amount to 25000 for reg number 'KA053408' and report num 12

```
update Participated
          set damage_amount = 25000 where reg_num = 'KA053408' and report_num = 12;
select * from Participated;
```

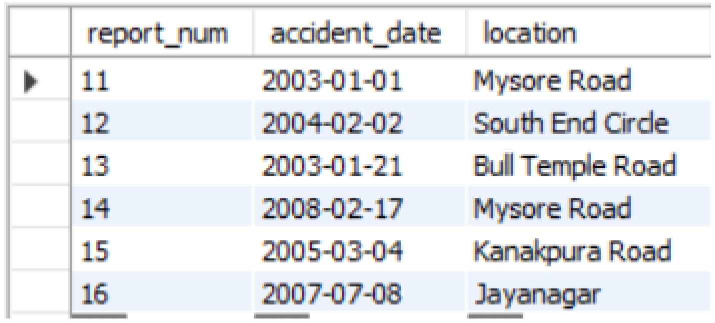


driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A02	KA053408	12	25000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000
NULL	NULL	NULL	NULL

-- add a new accident to the database

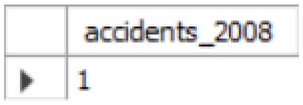
```
insert into Accident values (16,'2007-07-08','Jayanagar');
```

select * from Accident;



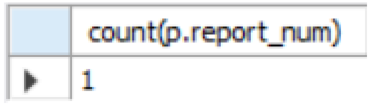
	report_num	accident_date	location
▶	11	2003-01-01	Mysore Road
	12	2004-02-02	South End Circle
	13	2003-01-21	Bull Temple Road
	14	2008-02-17	Mysore Road
	15	2005-03-04	Kanakpura Road
	16	2007-07-08	Jayanagar

-- find the total no. of people who owned cars involved in accident in 2008
select count(*) as accidents_2008 from Accident where accident_date between '2008-01-01' and '2008-12-31';



	accidents_2008
▶	1

-- find the number of accidents in which cars belonging to model 'Lancer' were involved
select count(p.report_num) from Participated p,Car c
where p.reg_num = c.reg_num and c.model = 'Lancer';



	count(p.report_num)
▶	1

Program 2 - Banking Database

Consider the following database for a banking enterprise.

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

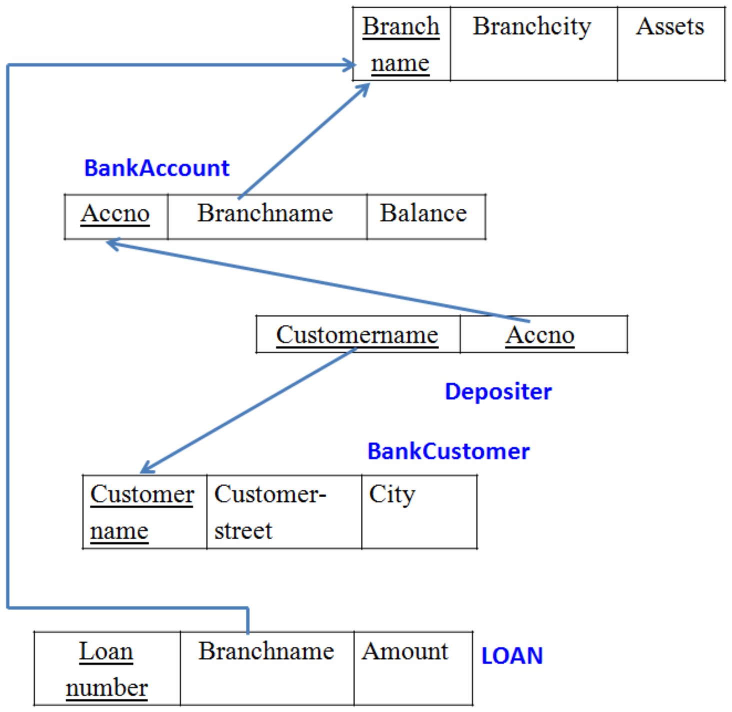
BankAccount(accno: int, branch-name: String, balance: real)

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

Depositor(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).



```
create database banking;
use banking;

create table Branch(
        branch_name varchar(20) not null,
        branch_city varchar(20) not null,
        assets real not null,
	primary key(branch_name)
);

create table BankAccount(
	accno int not null,
	branch_name varchar(20) not null,
	balance real not null,
	primary key(accno),
	foreign key(branch_name) references Branch(branch_name)
);

create table BankCustomer(
	customer_name varchar(20) not null,
	customer_street varchar(20) not null,
	customer_city varchar(20) not null,
	primary key(customer_name)
);

create table Depositor(
	customer_name varchar(20) not null,
	accno int not null,
	primary key(customer_name,accno),
	foreign key(customer_name) references BankCustomer(customer_name),
	foreign key(accno) references BankAccount(accno)
);

create table Loan(
	loanno int not null,
	branch_name varchar(20) not null,
	amount real not null,
	primary key(loanno),
	foreign key(branch_name) references Branch(branch_name)
);

insert into Branch(branch_name,branch_city,assets)
	values ('SBI_Chamrajpet','Bangalore',50000),
		('SBI_ResidencyRoad','Bangalore',10000),
		('SBI_ShivajiRoad','Bombay',20000),
		('SBI_ParliamentRoad','Delhi',10000),
		('SBI_Jantarmantar','Delhi',20000);
```

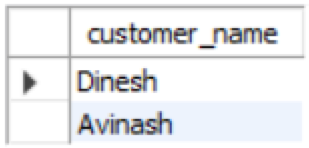
```
insert into BankAccount(accno,branch_name,balance)
	values (1,'SBI_Chamrajpet',2000),
		(2,'SBI_ResidencyRoad',5000),
		(3,'SBI_ShivajiRoad',6000),
		(4,'SBI_ParliamentRoad',9000),
		(5,'SBI_Jantarmantar',8000),
		(6,'SBI_ShivajiRoad',4000),
		(8,'SBI_ResidencyRoad',4000),
		(9,'SBI_ParliamentRoad',3000),
		(10,'SBI_ResidencyRoad',5000),
		(11,'SBI_Jantarmantar',2000);

insert into BankCustomer(customer_name,customer_street,customer_city)
	values ('Avinash','Bull_Temple_Road','Bangalore'),
		('Dinesh','Bannerghatta_Road','Bangalore'),
		('Mohan','NationalCollege_Road','Bangalore'),
		('Nikil','Akbar_Road','Delhi'),
		('Ravi','Prithviraj_Road','Delhi');

insert into Depositor(customer_name,accno)
	values ('Avinash',1),
		('Dinesh',2),
		('Nikil',4),
		('Ravi',5),
		('Avinash',8),
		('Nikil',9),
		('Dinesh',10),
		('Nikil',11);

insert into Loan(loanno,branch_name,amount)
	values (1,'SBI_Chamrajpet',1000),
		(2,'SBI_ResidencyRoad',2000),
		(3,'SBI_ShivajiRoad',3000),
		(4,'SBI_ParliamentRoad',4000),
		(5,'SBI_Jantarmantar',5000);
```

-- Branch,BankAccount,BankCustomer,Depositor,Loan

-- find all the customers who have atleast two accounts at the 'SBI_ResidencyRoad'

```
select distinct d.customer_name from Depositor d,BankAccount ba,Branch b
```

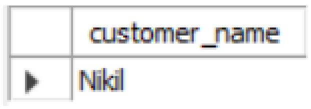
```
where d.accno = ba.accno and ba.branch_name = b.branch_name and ba.branch_name =
'SBI_ResidencyRoad'
and 2 <= (select count(accno) from Depositor where customer_name = d.customer_name);
```



	customer_name
▶	Dinesh
	Avinash

-- find all the customers who have an account at all branches in Delhi

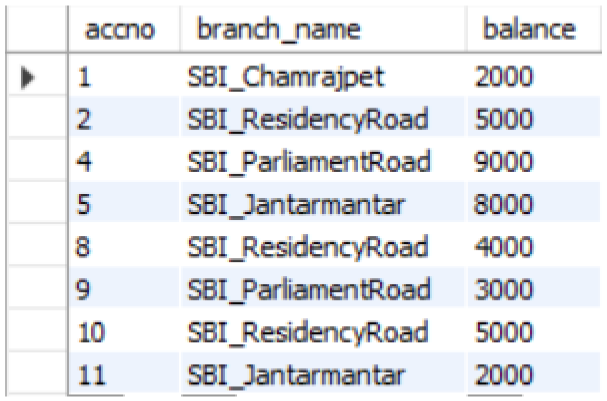
```
select d.customer_name from Depositor d,BankAccount ba,Branch b
where d.accno = ba.accno and b.branch_name = ba.branch_name and b.branch_city =
'Delhi'
group by d.customer_name
having count(d.accno) >= (select count(branch_name) from Branch where branch_city =
'Delhi');
```



	customer_name
▶	Nikil

-- demonstrate how you delete all account tuples at every branch located at Bombay

```
delete from BankAccount
where branch_name in (select branch_name from Branch where branch_city =
'Bombay');
select * from BankAccount;
```



	accno	branch_name	balance
▶	1	SBI_Chamrajpet	2000
	2	SBI_ResidencyRoad	5000
	4	SBI_ParliamentRoad	9000
	5	SBI_Jantarmantar	8000
	8	SBI_ResidencyRoad	4000
	9	SBI_ParliamentRoad	3000
	10	SBI_ResidencyRoad	5000
	11	SBI_Jantarmantar	2000

Program 3 - Suppliers Database

Consider the following schema:

SUPPLIERS(<u>sid: integer</u>, sname: string, address: string)

PARTS(<u>pid: integer</u>, pname: string, color: string)

CATALOG(<u>sid: integer, pid</u>: integer, cost: real)

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

Write the following queries in SQL:

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

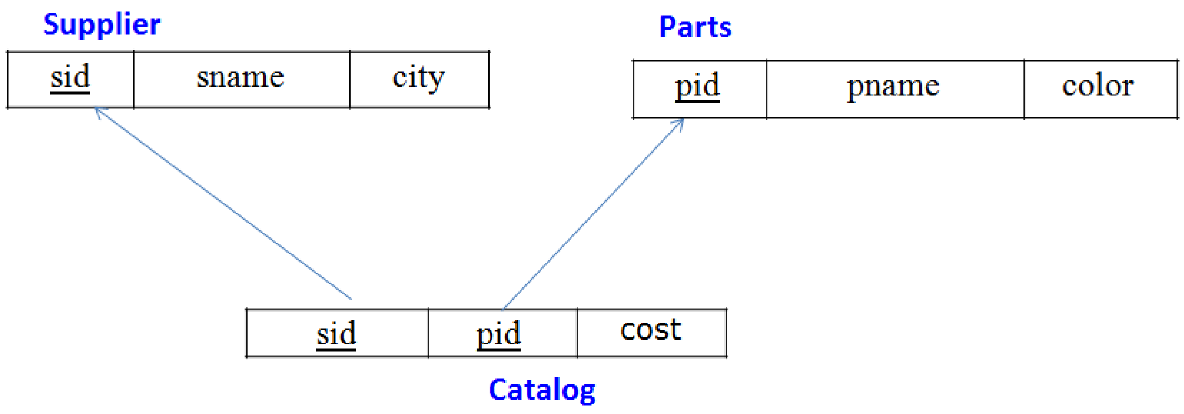
ii) Find the snames of suppliers who supply every part.

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

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

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).

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



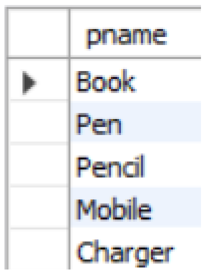
```
create database supplierdb;
use supplierdb;

create table Suppliers(
        sid int not null,
        sname varchar(20) not null,
        address varchar(20) not null,
        primary key(sid)
);
```

```
create table Parts(
        pid int not null,
        pname varchar(20) not null,
        color varchar(20) not null,
        primary key(pid)
);

create table Catalog(
        sid int not null,
        pid int not null,
        cost real not null,
        primary key(sid,pid),
        foreign key(sid) references Suppliers(sid),
        foreign key(pid) references Parts(pid)
);

insert into Suppliers(sid,sname,address)
        values (10001,'Acme Widget','Bangalore'),
                (10002,'Johns','Kolkata'),
                (10003,'Vimal','Mumbai'),
                (10004,'Reliance','Delhi');

insert into Parts(pid,pname,color)
        values (20001,'Book','Red'),
                (20002,'Pen','Red'),
                (20003,'Pencil','Green'),
                (20004,'Mobile','Green'),
                (20005,'Charger','Black');

insert into Catalog(sid,pid,cost)
```

```
          values (10001,20001,10),
                 (10001,20002,10),
                 (10001,20003,30),
                 (10001,20004,10),
                 (10001,20005,10),
                 (10002,20001,10),
                 (10002,20002,20),
                 (10003,20003,30),
                 (10004,20003,40);
```

-- find the pnames of parts for which there is some supplier

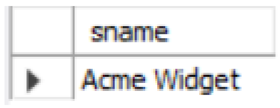
```
select distinct p.pname from Parts p,Catalog c
where p.pid = c.pid and exists (select 'X' from Catalog where pid = p.pid);
```



```
pname
Book
Pen
Pencil
Mobile
Charger
```

-- find the snames of suppliers who supply every part

```
select s.sname from Suppliers s,Catalog c
where s.sid = c.sid
group by s.sname
having count(c.pid) = (select count(*) from Parts);
```



```
sname
Acme Widget
```

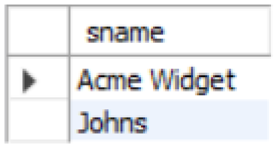
-- find the snames of suppliers who supply every red part

```
select s.sname from Suppliers s,Parts p,Catalog c
where s.sid = c.sid and p.pid = c.pid
```

and p.color = 'Red'

group by s.sname

having count(c.pid) = (select count(*) from Parts where color = 'Red');



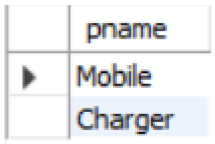
sname
Acme Widget
Johns

-- find the pname of parts supplied by acme widget and no one else

select p.pname from Parts p,Catalog c

where p.pid = c.pid

and p.pid not in (select pid from Catalog where sid in (select sid from Suppliers where sname <> 'Acme Widget'));

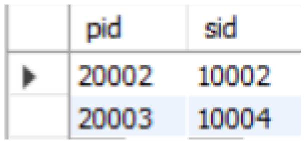


pname
Mobile
Charger

-- find the sids of suppliers who charge more for some part than average cost of that part

select distinct c.pid,c.sid from Catalog c

where c.cost > (select avg(c1.cost) from Catalog c1 where c1.pid = c.pid);



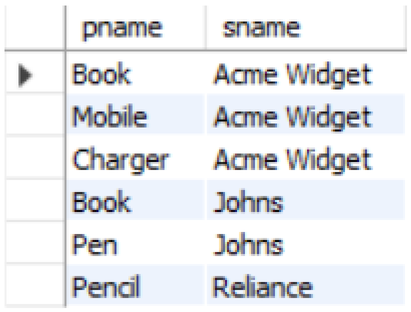
pid	sid
20002	10002
20003	10004

-- for each part, find the sname of supplier who charges most for that part

select p.pname,s.sname from Parts p,Suppliers s,Catalog c

where p.pid = c.pid and s.sid = c.sid

and c.cost = (select max(cost) from Catalog where pid = p.pid);



pname	sname
Book	Acme Widget
Mobile	Acme Widget
Charger	Acme Widget
Book	Johns
Pen	Johns
Pencil	Reliance

Program 4 - Student Database

Consider the following database for student enrollment for course :

STUDENT(<u>snum</u>: integer, sname: string, major: string, lvl: string, age: integer)

CLASS(<u>cname</u>: string, meets at: time, room: string, fid: integer)

ENROLLED(<u>snum</u>: integer, <u>cname</u>: string)

FACULTY(<u>fid</u>: 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.

i. Find the names of all Juniors (level = JR) who are enrolled in a class taught by

ii. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.

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

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

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

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

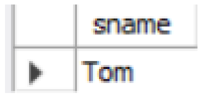
vii. For each age value that appears in Students, find the level value that appears most often.

```
create database studentfaculty;

use studentfaculty;


create table Student(

        snum int not null,

        sname varchar(10) not null,
	major varchar(2) not null,
	lvl varchar(2) not null,
	age int not null,
	primary key(snum)
);

create table Faculty(
	fid int not null,
	fname varchar(10) not null,
	deptid int not null,
	primary key(fid)
);

create table Class(
	cname varchar(10) not null,
	meetsat time not null,
	room varchar(10) not null,
	fid int not null,
	primary key(cname),
	foreign key(fid) references Faculty(fid)
);

create table Enrolled(
	snum int not null,
	cname varchar(10) not null,
	primary key(snum,cname),
	foreign key(snum) references Student(snum),
        foreign key(cname) references Class(cname)
);

insert into Student(snum,sname,major,lvl,age)
        values (1,'Jhon','CS','Sr',19),
                (2,'Smith','CS','Jr',20),
                (3,'Jacob','CV','Sr',20),
                (4,'Tom','CS','Jr',20),
                (5,'Rahul','CS','Jr',20),
                (6,'Rita','CS','Sr',21);

insert into Faculty(fid,fname,deptid)
        values (11,'Harish',1000),
                (12,'MV',1000),
                (13,'Mira',1001),
                (14,'Shiva',1002),
                (15,'Nupur',1000);

insert into Class(cname,meetsat,room,fid)
        values ('Class1','10:15:16','R1',14),
                ('Class10','10:15:16','R128',14),
                ('Class2','10:15:20','R2',12),
                ('Class3','10:15:25','R3',11),
                ('Class4','20:15:20','R4',14),
                ('Class5','20:15:20','R3',15),
                ('Class6','13:20:20','R2',14),
                ('Class7','10:10:10','R3',14);
```

```
insert into Enrolled(snum,cname)
        values (1,'Class1'),
                (2,'Class1'),
                (3,'Class3'),
                (4,'Class3'),
                (5,'Class4');

insert into Enrolled(snum,cname)
        values (1,'Class5'),
                (2,'Class5'),
                (3,'Class5'),
                (4,'Class5'),
                (5,'Class5');
```

-- find names of all juniors who are enrolled in a class taught by 'Harish'

```
select s.sname from Student s,Enrolled e
where s.snum = e.snum
and s.lvl = 'Jr'
and e.cname in (select cname from Class where fid = (select fid from Faculty where fname = 'Harish'));
```



```
sname
Tom
```

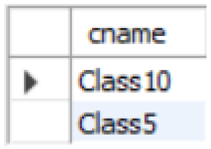
-- find the name of all classes that either meet in room128 or have >=5 students

```
select c.cname from Class c
where c.room = 'R128'
```

union

select distinct e.cname from Enrolled e

where 5 <= (select count(snum) from Enrolled where cname = e.cname);



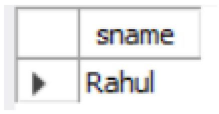
	cname
▶	Class10
	Class5

-- 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,Class c,Enrolled e

where s.snum = e.snum and c.cname = e.cname

and exists (select 'X' from Class c1,Enrolled e1 where c1.cname = e1.cname and c1.meetsat = c.meetsat and e1.snum = e.snum and c1.cname != e.cname);



	sname
▶	Rahul

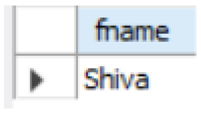
-- find the names of faculty who teach in every room in which class is taught

select f.fname from Faculty f,Class c

where f.fid = c.fid

group by f.fid

having count(f.fid) = (select count(distinct room) from Class);



	fname
▶	Shiva

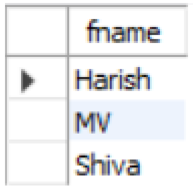
-- find the names of faculty members for whom combined enrollment of courses that they teach is less than five

select distinct f.fname

from Class c,Faculty f

where c.fid = f.fid

and 5 > (select count(snum) from enrolled where cname in (select cname from Class where Class.fid = c.fid));

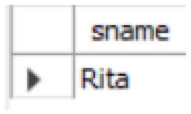


fname
Harish
MV
Shiva

-- find the names of students who are not enrolled in any class

select s.sname from Student s

where not exists (select 'X' from Enrolled where snum = s.snum);



sname
Rita

-- for each age value that appears in Students, find the level that appears the most

select s.age,s.lvl from Student s

where s.lvl = 'Jr'

group by s.age

having count(s.lvl) > (select count(lvl) from Student where lvl ='Sr' and age = s.age)

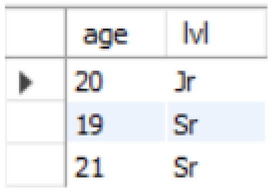
union

select s.age,s.lvl from Student s

where s.lvl = 'Sr'

group by s.age

having count(s.lvl) > (select count(lvl) from Student where lvl ='Jr' and age = s.age);



age	lvl
20	Jr
19	Sr
21	Sr

Program 5 - Airlines Database

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

FLIGHTS(<u>flno</u>: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer)

AIRCRAFT(<u>aid</u>: integer, aname: string, cruisingrange: integer)

CERTIFIED(eid: integer, aid: integer)

EMPLOYEES(<u>eid</u>: 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.

i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

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

iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

iv. 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.

v. Find the names of pilots certified for some Boeing aircraft.

vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

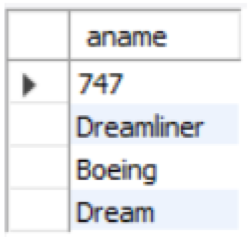
vii. A customer wants to travel from Madison to New York with no more than two changes of flight. List the choice of departure times from Madison if the customer wants to arrive in New York by 6 p.m.

```
create database airlinesdb;
use airlinesdb;

create table Flights(
        flno int not null,
        from_loc varchar(20) not null,
        to_loc varchar(20) not null,
        distance int not null,
        departs time not null,
	arrives time not null,
	price int not null,
	primary key(flno)
);

create table Aircraft(
	aid int not null,
	aname varchar(20) not null,
	cruisingrange int not null,
	primary key(aid)
);

create table Employees(
	eid int not null,
	ename varchar(20) not null,
	salary int not null,
	primary key(eid)
);

create table Certified(
	eid int not null,
	aid int not null,
	primary key(eid,aid),
	foreign key(eid) references Employees(eid),
	foreign key(aid) references Aircraft(aid)
);

insert into Flights(flno,from_loc,to_loc,distance,departs,arrives,price)
	values (101,"Bangalore","Delhi",2500,"07:15:31","12:15:31",5000),
		(102,"Bangalore","Lucknow",3000,"07:15:31","11:15:31",6000),
		(103,"Lucknow","Delhi",500,"12:15:31","17:15:31",3000),
		(107,"Bangalore","Frankfurt",8000,"07:15:31","22:15:31",60000),
		(104,"Bangalore","Frankfurt",8500,"07:15:31","23:15:31",75000),
		(105,"Kolkata","Delhi",3400,"07:15:31","09:15:31",7000);

insert into Flights(flno,from_loc,to_loc,distance,departs,arrives,price)
	values (106,"Delhi","Kolkata",3400,"12:15:35","14:20:00",7000);

insert into Aircraft(aid,aname,cruisingrange)
	values (101,"747",3000),
	(102,"Boeing",900),
	(103,"647",800),
	(104,"Dreamliner",10000),
	(105,"Boeing",3500),
	(106,"707",1500),
	(107,"Dream",12000);
```

```
insert into Employees(eid,ename,salary)
        values (701,'A',50000),
        (702,'B',100000),
         (703,'C',150000),
         (704,'D',90000),
         (705,'E',40000),
         (706,'F',60000),
         (707,'G',90000);

insert into Certified(eid,aid)
        values (701,101),
        (701,102),
         (701,106),
         (701,105),
         (702,104),
         (703,104),
         (704,104),
         (702,107),
         (703,107),
         (704,107),
         (702,101),
         (703,105),
         (704,105),
         (705,103);
```

-- find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

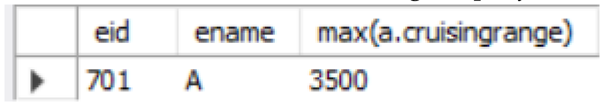
```
select distinct a.aname from Aircraft a,Employees e, Certified c
        where a.aid = c.aid and e.eid = c.eid and e.salary > 80000;
```



aname
747
Dreamliner
Boeing
Dream

-- 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.

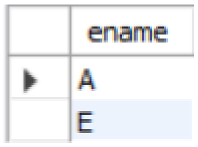
```
select e.eid,e.ename,max(a.cruisingrange) from Employees e,Certified c,Aircraft a where
e.eid = c.eid and a.aid = c.aid group by e.ename having count(c.aid) > 3;
```



eid	ename	max(a.cruisingrange)
701	A	3500

-- find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

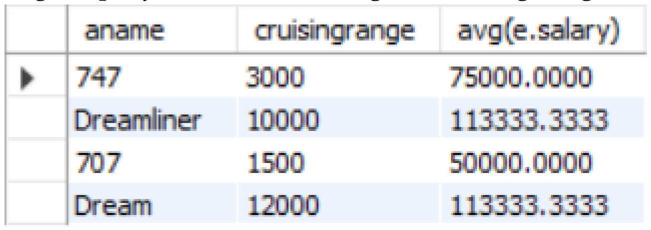
```
select e.ename from Employees e where salary < (select min(price) from Flights where
from_loc = "Bangalore" and to_loc = "Frankfurt");
```



ename
A
E

-- 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.

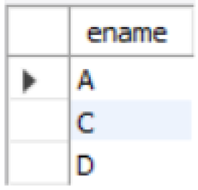
```
select a.aname,a.cruisingrange,avg(e.salary)
        from Aircraft a,Employees e,Certified c
        where c.eid = e.eid and c.aid = a.aid
   group by a.aname having a.cruisingrange > 1000 ;
```



aname	cruisingrange	avg(e.salary)
747	3000	75000.0000
Dreamliner	10000	113333.3333
707	1500	50000.0000
Dream	12000	113333.3333

-- find the names of pilots certified for some Boeing aircraft.

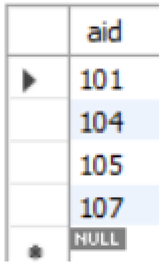
```
select distinct e.ename from Employees e,Certified c,Aircraft a
        where e.eid = c.eid and a.aid = c.aid and aname like "Boeing";
```



ename
A
C
D

-- find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

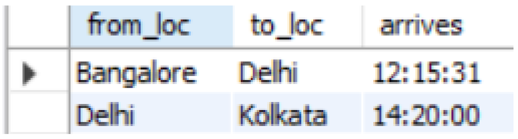
```
select a.aid from aircraft a where a.cruisingrange >= (select distance from Flights where
from_loc = "Bangalore" and to_loc = "Delhi");
```



aid
101
104
105
107
NULL

-- a customer wants to travel from bangalore to kolkata with no more than two changes of flight list the choice of departure times customer wants to arrive by 6 p.m.

```
select f.from_loc,f.to_loc,f.arrives from Flights f
where (f.from_loc = "Bangalore" and f.to_loc = (select from_loc from Flights where
to_loc = "Kolkata")) or f.to_loc = "Kolkata";
```



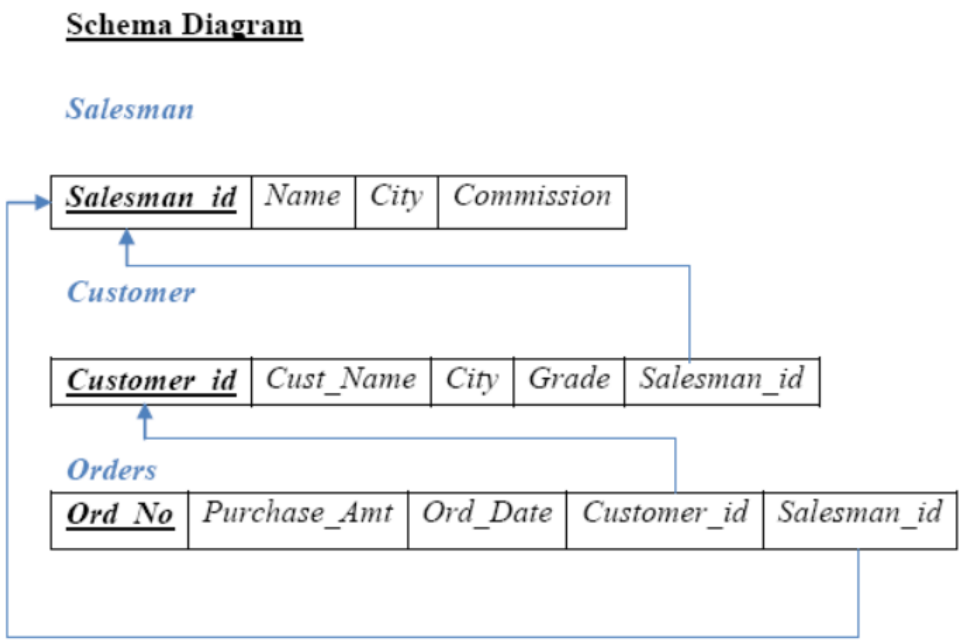
from_loc	to_loc	arrives
Bangalore	Delhi	12:15:31
Delhi	Kolkata	14:20:00

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

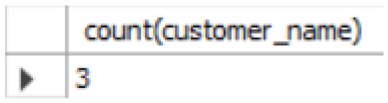
1. Count the customers with grades above Bangalore's average.
2. Find the name and numbers of all salesmen who had more than one customer.
3. List all salesmen and indicate those who have and don't have customers in their cities (Use UNION operation.)
4. Create a view that finds the salesman who has 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 Orderdb;
use Orderdb;

create table Salesman(
        salesman_id int not null,
        salesman_name varchar(20) not null,
        city varchar(20) not null,
        commission int not null,
        primary key(salesman_id)
);

create table Customer(
        customer_id int not null,
        customer_name varchar(20) not null,
        city varchar(20) not null,
        grade int not null,
        salesman_id int,
        primary key(customer_id),
        foreign key(salesman_id) references Salesman(salesman_id) on delete set null
);

create table Orders(
        order_id int not null,
        purchase_amt int not null,
        order_date date not null,
        customer_id int not null,
        salesman_id int,
        primary key(order_id),
        foreign key(customer_id) references Customer(customer_id),
        foreign key(salesman_id) references Salesman(salesman_id) on delete set null
);

insert into Salesman(salesman_id,salesman_name,city,commission)
        values (1000,'John','Bangalore',25),
                (2000,'Ravi','Bangalore',20),
                (3000,'Kumar','Mysore',15),
                (4000,'Smith','Delhi',30),
                (5000,'Harsha','Hyderabad',15);

insert into Customer(customer_id,customer_name,city,grade,salesman_id)
        values (10,'Preethi','Bangalore',100,1000),
                (11,'Vivek','Mangalore',300,1000),
                (12,'Bhaskar','Chennai',400,2000),
                (13,'Chethan','Bangalore',200,2000),
                (14,'Mamatha','Bangalore',400,3000);

insert into Orders(order_id,purchase_amt,order_date,customer_id,salesman_id)
        values (50,5000,'2017-05-04',10,1000),
                (51,450,'2017-01-20',10,2000),
                (52,1000,'2017-02-24',13,2000),
                (53,3500,'2017-04-13',14,3000),
                (54,550,'2017-03-09',12,2000);
```

-- count the customers with grades above Bangalore's average

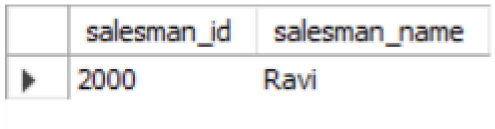
```
select count(customer_name) from Customer where grade > (select avg(grade) from
Customer where city = 'Bangalore');
```



	count(customer_name)
▶	3

-- Find the name and numbers of all salesmen who had more than one customer

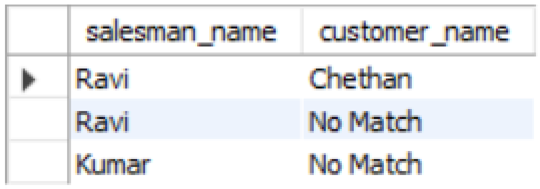
```
select distinct c.salesman_id,s.salesman_name from Customer c,Salesman s
where c.salesman_id = s.salesman_id
and 1 < (select count(customer_id) from Customer where salesman_id = c.salesman_id);
```



	salesman_id	salesman_name
▶	2000	Ravi

-- List all salesmen and indicate those who have and dont have customers in their city

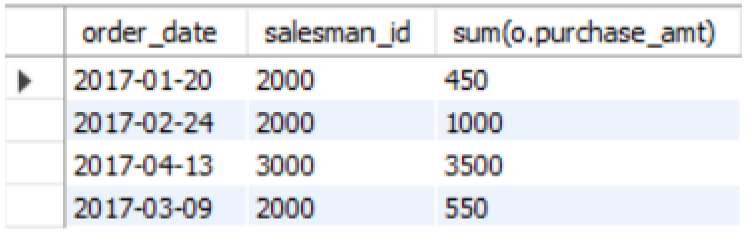
```
select s.salesman_name,c.customer_name from Salesman s,Customer c
where s.salesman_id = c.salesman_id and c.city = s.city
union
select s.salesman_name,'No Match' from Salesman s,Customer c
where s.salesman_id = c.salesman_id and c.city != s.city;
```



	salesman_name	customer_name
▶	Ravi	Chethan
	Ravi	No Match
	Kumar	No Match

-- create a view that finds the salesman who has the customer with the highest order of the day

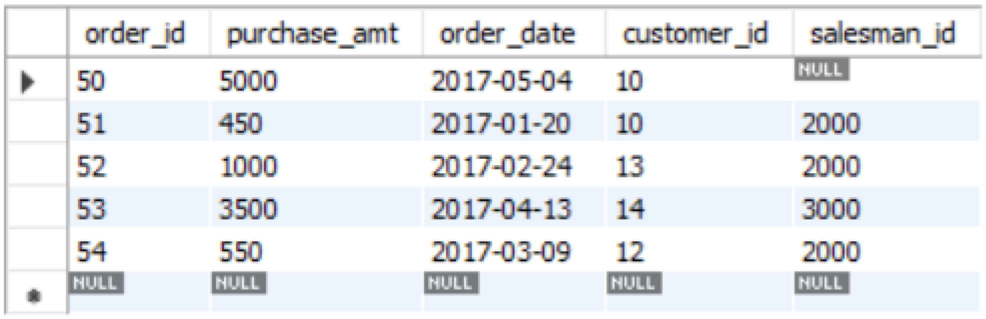
```
create view salesman_view as
select o.order_date,salesman_id,sum(o.purchase_amt) from Orders o group by order_date
having sum(purchase_amt) = (select max(sum(purchase_amt)) from Customer where
order_date = o.order_date and salesman_id = o.salesman_id);
select * from salesman_view;
```



	order_date	salesman_id	sum(o.purchase_amt)
▶	2017-01-20	2000	450
	2017-02-24	2000	1000
	2017-04-13	3000	3500
	2017-03-09	2000	550

-- delete salesman with id 1000

```
delete from Salesman where salesman_id = 1000;
select * from Salesman;
select * from Orders;
```

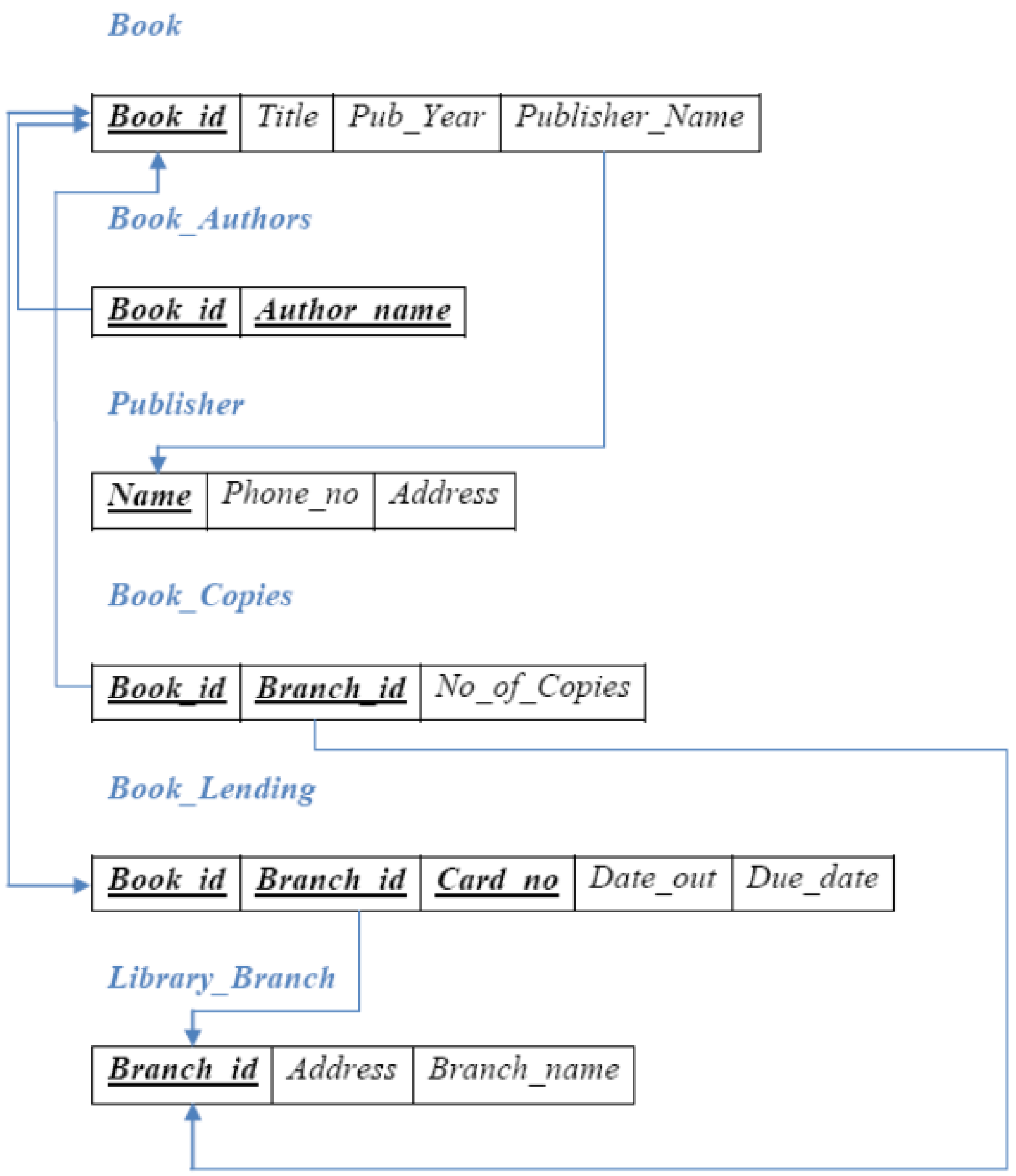


order_id	purchase_amt	order_date	customer_id	salesman_id
50	5000	2017-05-04	10	NULL
51	450	2017-01-20	10	2000
52	1000	2017-02-24	13	2000
53	3500	2017-04-13	14	3000
54	550	2017-03-09	12	2000
NULL	NULL	NULL	NULL	NULL

Program 7 - Book Database

1. Retrieve details of all books in the library – id, title, name of publisher, authors, number of copies in each branch, etc.
2. Get the particulars of borrowers who have borrowed more than 3 books, but from Jan 2017 to Jun 2017
3. Delete a book in BOOK table. Update the contents of other tables to reflect this data manipulation operation.
4. Partition the BOOK table based on year of publication. Demonstrate its working with a simple query.
5. Create a view of all books and its number of copies that are currently available in the Library.

Schema Diagram



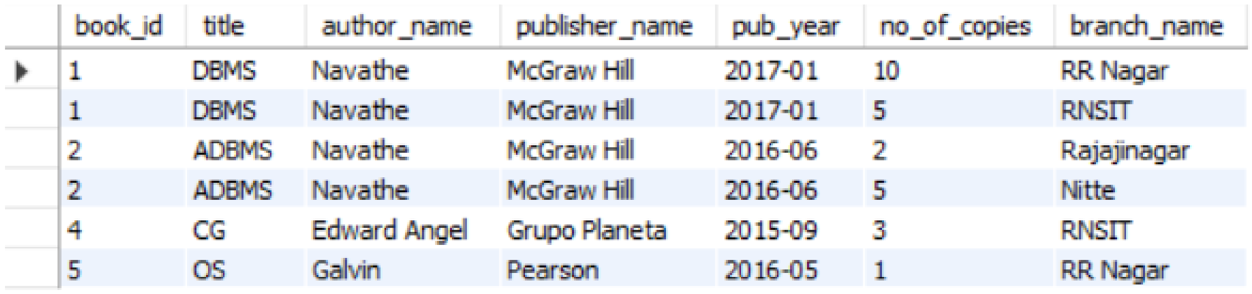
create database librarydb;

```
use librarydb;

create table Book(
	book_id int not null,
	title varchar(10) not null,
	pub_year varchar(20) not null,
   publisher_name varchar(20) not null,
	primary key(book_id)
);

create table BookAuthors(
	book_id int not null,
	author_name varchar(20) not null,
	primary key(book_id,author_name),
	foreign key(book_id) references Book(book_id) on delete cascade
);

create table Publisher(
	p_name varchar(20) not null,
	phone_no varchar(10) not null,
	address varchar(20) not null,
	primary key(p_name)
);

create table LibraryBranch(
	branch_id int not null,
	address varchar(20) not null,
	branch_name varchar(20) not null,
	primary key(branch_id)
);

create table BookCopies(
	book_id int not null,
	branch_id int not null,
	no_of_copies int not null,
	primary key(book_id,branch_id),
	foreign key(book_id) references Book(book_id)on delete cascade,
	foreign key(branch_id) references LibraryBranch(branch_id) on delete cascade
);

create table Card(
	card_no int not null,
   primary key(card_no)
);

create table BookLending(
	book_id int not null,
        branch_id int not null,
        card_no int not null,
        date_out date not null,
        due_date date not null,
        primary key(book_id,branch_id,card_no),
        foreign key(book_id) references Book(book_id) on delete cascade,
        foreign key(branch_id) references LibraryBranch(branch_id) on delete cascade,
        foreign key(card_no) references Card(card_no) on delete cascade
);

insert into Publisher(p_name,phone_no,address)
        values ('McGraw Hill','9989076587','Bangalore'),
                ('Pearson','9889076565','New Delhi'),
                ('Random House','7455679345','Hyderabad'),
                ('Hachette Livre','8970862340','Chennai'),
                ('Grupo Planeta','7756120238','Bangalore');

insert into Book(book_id,title,pub_year,publisher_name)
        values (1,'DBMS','2017-01','McGraw Hill'),
                (2,'ADBMS','2016-06','McGraw Hill'),
                (3,'CN','2016-09','Pearson'),
                (4,'CG','2015-09','Grupo Planeta'),
                (5,'OS','2016-05','Pearson');

insert into BookAuthors(author_name,book_id)
        values ('Navathe',1),
                ('Navathe',2),
                ('Tanenbaum',3),
                ('Edward Angel',4),
                ('Galvin',5);

insert into LibraryBranch(branch_id,branch_name,address)
        values (10,'RR Nagar','Bangalore'),
                (11,'RNSIT','Bangalore'),
                (12,'Rajajinagar','Bangalore'),
                (13,'Nitte','Mangalore'),
                (14,'Manipal','Udupi');

insert into BookCopies(book_id,branch_id,no_of_copies)
        values (1,10,10),
                (1,11,5),
                (2,12,2),
                (2,13,5),
                (3,14,7),
                (5,10,1),
                (4,11,3);
```

```
insert into Card(card_no)
        values (100),
               (101),
               (102),
               (103),
               (104);

insert into BookLending(date_out,due_date,book_id,branch_id,card_no)
        values ('2017-01-01','2017-06-01',1,10,101),
               ('2017-01-11','2017-03-11',3,14,101),
               ('2017-02-21','2017-04-21',2,13,101),
               ('2017-03-15','2017-07-15',4,11,101),
               ('2017-04-12','2017-05-12',1,11,104);
```

-- Queries
-- Retrieve the details of all books in library - id,title,publisher

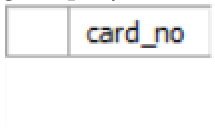
```
name,author,no_of_copies in each branch
select
b.book_id,b.title,ba.author_name,b.publisher_name,b.pub_year,bc.no_of_copies,lb.branc
h_name
from Book b,BookCopies bc,LibraryBranch lb,BookAuthors ba
where b.book_id = bc.book_id and lb.branch_id = bc.branch_id and b.book_id =
ba.book_id;
```



book_id	title	author_name	publisher_name	pub_year	no_of_copies	branch_name
1	DBMS	Navathe	McGraw Hill	2017-01	10	RR Nagar
1	DBMS	Navathe	McGraw Hill	2017-01	5	RNSIT
2	ADBMS	Navathe	McGraw Hill	2016-06	2	Rajajinagar
2	ADBMS	Navathe	McGraw Hill	2016-06	5	Nitte
4	CG	Edward Angel	Grupo Planeta	2015-09	3	RNSIT
5	OS	Galvin	Pearson	2016-05	1	RR Nagar

-- Get the particulars of borrowers who have borrowed more than 3 books between jan to jun 2017

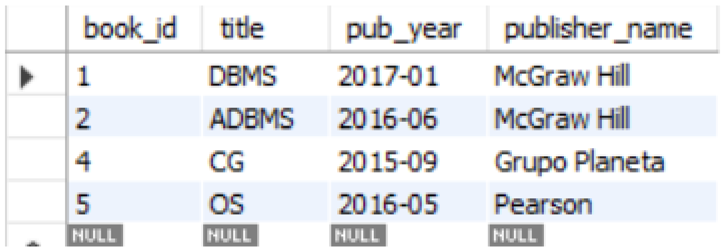
```
select card_no from BookLending
where date_out between '2017-01-01' and '2017-06-30'
group by card_no having count(book_id) > 3;
```



card_no

-- delete a book from the table

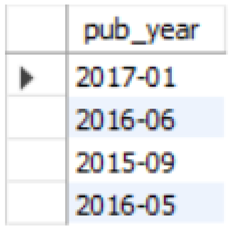
```
delete from Book where title = 'CN';
select * from Book;
```



book_id	title	pub_year	publisher_name
1	DBMS	2017-01	McGraw Hill
2	ADBMS	2016-06	McGraw Hill
4	CG	2015-09	Grupo Planeta
5	OS	2016-05	Pearson
NULL	NULL	NULL	NULL

-- partition the book table based on the year of publication
create view book_dates as
select pub_year from Book;

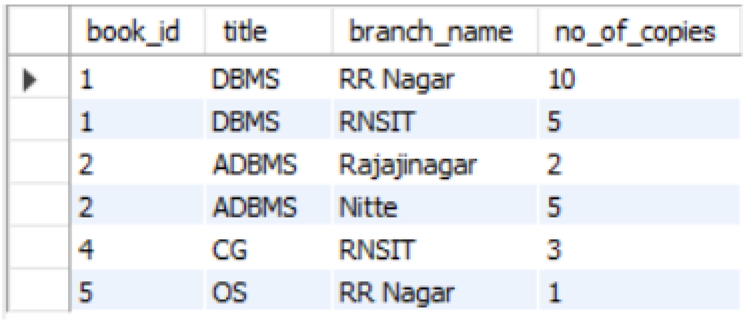
select * from book_dates;



pub_year
2017-01
2016-06
2015-09
2016-05

-- create a view of all books and its number of copies currently available in library
create view book_view as
select b.book_id,b.title,lb.branch_name,bc.no_of_copies
from Book b,BookCopies bc,Librarybranch lb
where b.book_id = bc.book_id and lb.branch_id = bc.branch_id;

select * from book_view;



book_id	title	branch_name	no_of_copies
1	DBMS	RR Nagar	10
1	DBMS	RNSIT	5
2	ADBMS	Rajajinagar	2
2	ADBMS	Nitte	5
4	CG	RNSIT	3
5	OS	RR Nagar	1

Program 8 - Student Enrollment Database

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)

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.

```
create database student2;
use student2;

create table Student(
        regno varchar(10) not null,
        sname varchar(20) not null,
        major varchar(20) not null,
        bdate date not null,
        primary key(regno)
);

create table Course(
        courseno int not null,
        cname varchar(20) not null,
        dept varchar(20) not null,
        primary key(courseno)
);

create table Enroll(
        regno varchar(10) not null,
        courseno int not null,
        sem int not null,
        marks int not null,
        primary key(regno,courseno),
        foreign key(regno) references Student(regno) on delete cascade,
        foreign key(courseno) references Course(courseno) on delete cascade
);
```

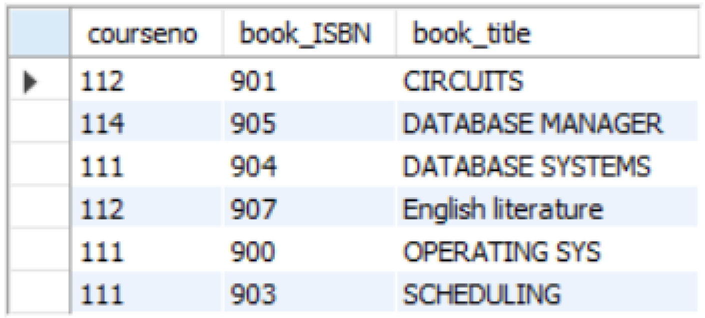
```
create table Text(
        book_ISBN int not null,
        book_title varchar(20) not null,
        publisher varchar(20) not null,
        author varchar(20) not null,
        primary key(book_ISBN)
);

create table BookAdoption(
        courseno int not null,
        sem int not null,
        book_ISBN int not null,
        primary key(courseno,book_ISBN),
        foreign key(courseno) references Course(courseno) on delete cascade,
        foreign key(book_ISBN) references Text(book_ISBN) on delete cascade
);

insert into Student(regno,sname,major,bdate)
        values
   ('1pe11cs001','a','jr','1993-10-25'),
        ('1pe11cs002','b','sr','1993-09-24'),
        ('1pe11cs003','c','sr','1993-11-27'),
        ('1pe11cs004','d','sr','1993-04-13'),
        ('1pe11cs005','e','jr','1994-08-24');

insert into Course(courseno,cname,dept)
        values
        (111,'OS','CSE'),
        (112,'EC','CSE'),
        (113,'SS','ISE'),
        (114,'DBMS','CSE'),
        (115,'SIGNALS','ECE');

insert into Enroll(regno,courseno,sem,marks)
        values
        ('1pe11cs001',115,3,100),
        ('1pe11cs002',114,5,100),
        ('1pe11cs003',113,5,100),
        ('1pe11cs004',111,5,100),
        ('1pe11cs005',112,3,100);

insert into Text(book_ISBN,book_title,publisher,author)
        values
        (10,'DATABASE SYSTEMS','PEARSON','SCHIELD'),
        (900,'OPERATING SYS','PEARSON','LELAND'),
        (901,'CIRCUITS','HALL INDIA','BOB'),
        (902,'SYSTEM SOFTWARE','PETERSON','JACOB'),
        (903,'SCHEDULING','PEARSON','PATIL'),
        (904,'DATABASE SYSTEMS','PEARSON','JACOB'),
        (905,'DATABASE MANAGER','PEARSON','BOB'),
        (906,'SIGNALS','HALL INDIA','SUMIT');

insert into BookAdoption(courseno,sem,book_ISBN)
        values
        (111,5,900),
        (111,5,903),
        (111,5,904),
        (112,3,901),
        (113,3,10),
        (114,5,905),
        (113,5,902),
        (115,3,906);

update Text set publisher = 'PEARSON' where book_ISBN = 907;
```

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

```
insert into Text values (907,'English literature','Random House','ABCDE');
insert into BookAdoption values (112,5,907);
```

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

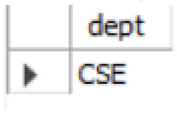
```
select ba.courseno,ba.book_ISBN,t.book_title
from BookAdoption ba,Text t,Course c
where ba.book_ISBN = t.book_ISBN and ba.courseno = c.courseno
and c.dept = 'CSE'
and 2 < (select count(ba1.courseno) from BookAdoption ba1,Course c1 where
ba1.courseno = c1.courseno and ba1.sem = ba.sem and c.dept = 'CSE')
order by t.book_title;
```



courseno	book_ISBN	book_title
112	901	CIRCUITS
114	905	DATABASE MANAGER
111	904	DATABASE SYSTEMS
112	907	English literature
111	900	OPERATING SYS
111	903	SCHEDULING

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

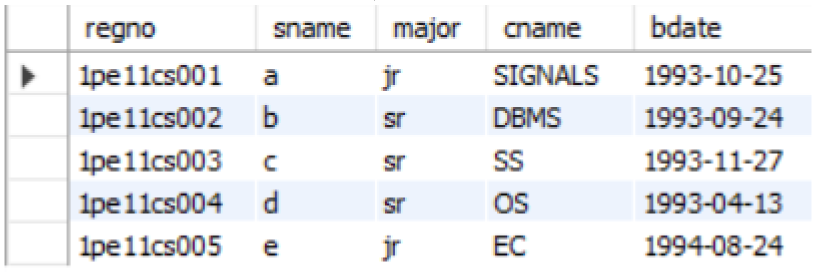
```
select c.dept from Course c,BookAdoption ba
where c.courseno = ba.courseno
group by c.dept
having count(ba.book_ISBN) = (select count(book_ISBN) from Text where publisher =
'Pearson');
```



dept
CSE

-- Generate suitable reports.

```
create view student_view as
select s.regno,s.sname,s.major,c.cname,s.bdate from Student s,Enroll e,course c where
s.regno = e.regno and c.courseno = e.courseno;

select * from student_view;
```



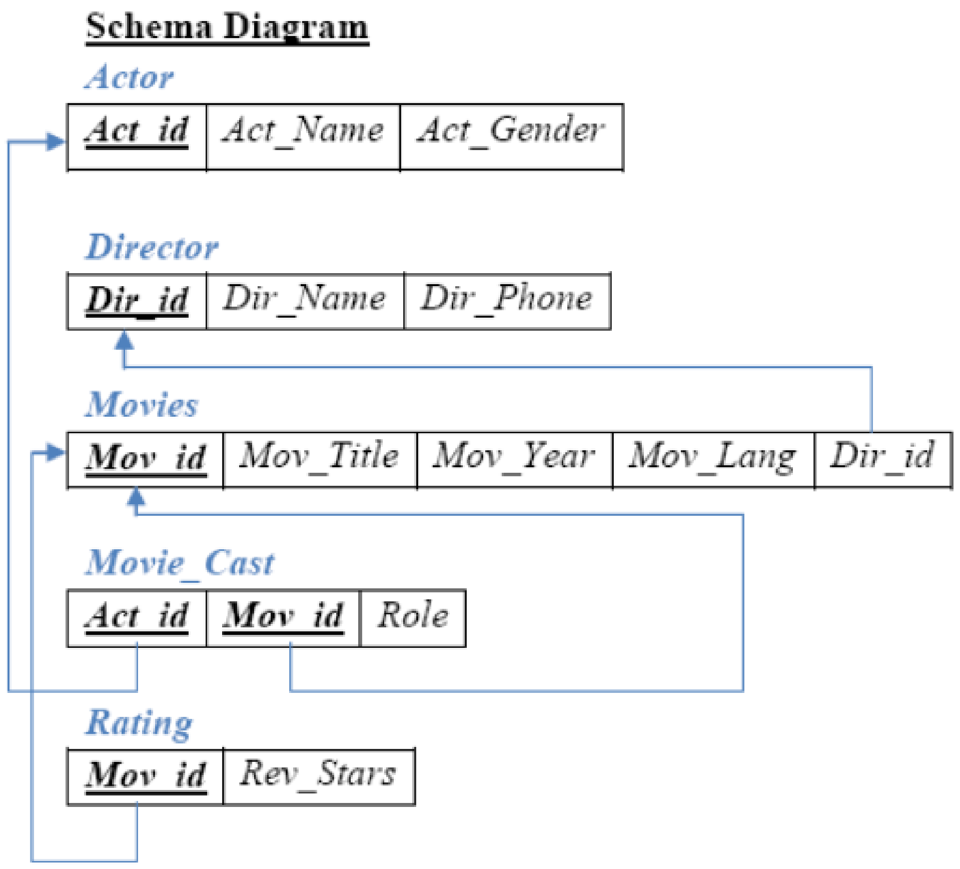
regno	sname	major	cname	bdate
1pe11cs001	a	jr	SIGNALS	1993-10-25
1pe11cs002	b	sr	DBMS	1993-09-24
1pe11cs003	c	sr	SS	1993-11-27
1pe11cs004	d	sr	OS	1993-04-13
1pe11cs005	e	jr	EC	1994-08-24

Program 9 - Movies 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.



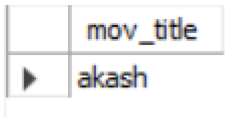
create database moviesDB;
use moviesDB;

```
create table Actor(
        act_id int not null,
        act_name varchar(20) not null,
        act_gender varchar(1) not null,
        primary key(act_id)
);

create table Director(
        dir_id int not null,
        dir_name varchar(20) not null,
        dir_phone varchar(10) not null,
        primary key(dir_id)
);

create table Movies(
        mov_id int not null,
        mov_title varchar(20) not null,
        mov_year year not null,
        mov_lang varchar(20) not null,
        dir_id int,
        primary key(mov_id),
        foreign key(dir_id) references Director(dir_id) on delete cascade
);

create table MovieCast(
        act_id int not null,
        mov_id int not null,
        role varchar(20) not null,
        primary key(act_id,mov_id),
        foreign key(act_id) references Actor(act_id) on delete cascade,
        foreign key(mov_id) references Movies(mov_id) on delete cascade
);

create table Rating(
        mov_id int not null,
        rev_stars int not null,
        primary key(mov_id),
        foreign key(mov_id) references Movies(mov_id) on delete cascade
);


insert into Actor(act_id,act_name,act_gender)
        values(301,'Anushka','F'),
                (302,'Prabhas','M'),
                (303,'Punith','M'),
                (304,'Jermy','M');
```

```
insert into Director(dir_id,dir_name,dir_phone)
        values(60,'Rajamouli','8751611001'),
                (61,'Hitchcock','7766138911'),
                (62,'Faran',9986776531),
                (63,'Steven Spielberg',8989776530);

insert into Movies(mov_id,mov_title,mov_year,mov_lang,dir_id)
        values(1001,'baahubali 2','2017','Telugu',60),
                (1002,'baahubali 1','2015','Telugu',60),
                (1003,'akash','2008','Kannada',61),
                (1004,'war horse','2011','English',63);

insert into MovieCast(act_id,mov_id,role)
        values(301,1002,'Heroine'),
                (301,1001,'Heroine'),
                (303,1003,'Hero'),
                (303,1002,'Guest'),
                (304,1004,'Hero');

insert into Rating(mov_id,rev_stars)
        values(1001,4),
                (1002,2),
                (1003,5),
                (1004,4);
```

-- List the titles of all movies directed by 'Hitchcock'.

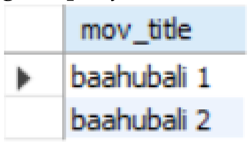
```
select mov_title from Movies,Director
where Movies.dir_id = Director.dir_id and dir_name like 'Hitchcock';
```



	mov_title
▶	akash

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

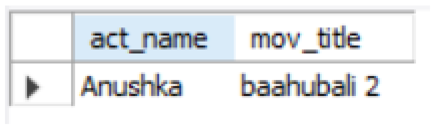
```
select distinct m.mov_title from Movies m,MovieCast mc
where m.mov_id = mc.mov_id
group by mc.act_id having count(mc.mov_id) >= 2;
```



	mov_title
▶	baahubali 1
	baahubali 2

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

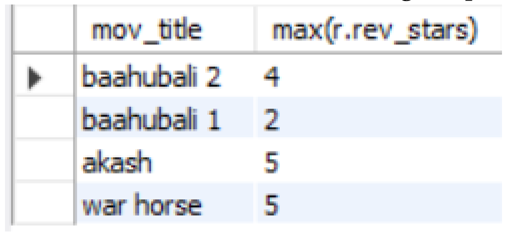
```
select a.act_name,m.mov_title from Actor a,Movies m,MovieCast mc
where m.mov_id = mc.mov_id and a.act_id = mc.act_id
and m.mov_year not between '2000' and '2015';
```



act_name	mov_title
Anushka	baahubali 2

-- 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.

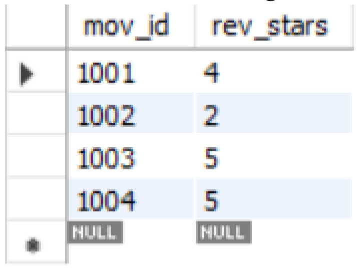
select m.mov_title,max(r.rev_stars) from Movies m,Rating r
where m.mov_id = r.mov_id group by r.mov_id;



mov_title	max(r.rev_stars)
baahubali 2	4
baahubali 1	2
akash	5
war horse	5

-- Update rating of all movies directed by 'Steven Spielberg' to 5.
update Rating
set rev_stars = 5
where mov_id = (select mov_id from Movies where dir_id = (select dir_id from Director where dir_name = 'Steven Spielberg'));

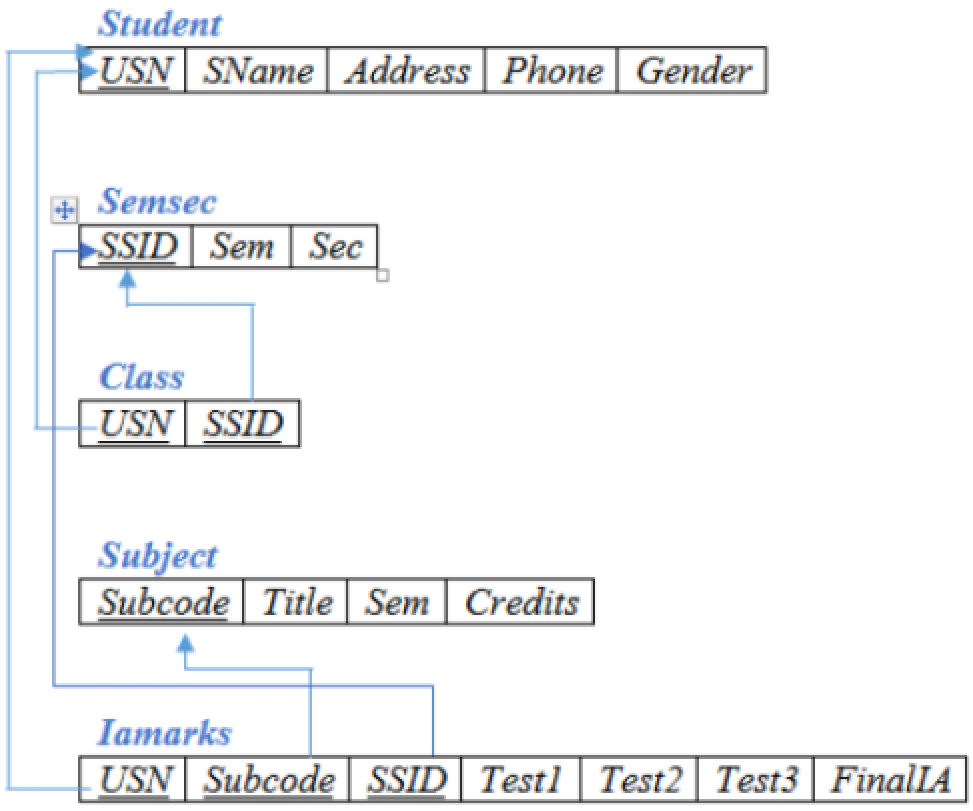
select * from Rating;



mov_id	rev_stars
1001	4
1002	2
1003	5
1004	5
NULL	NULL

Program 10 - College Database
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.

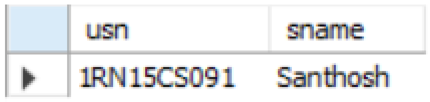


create database collegeDB;
use collegeDB;

create table Student(

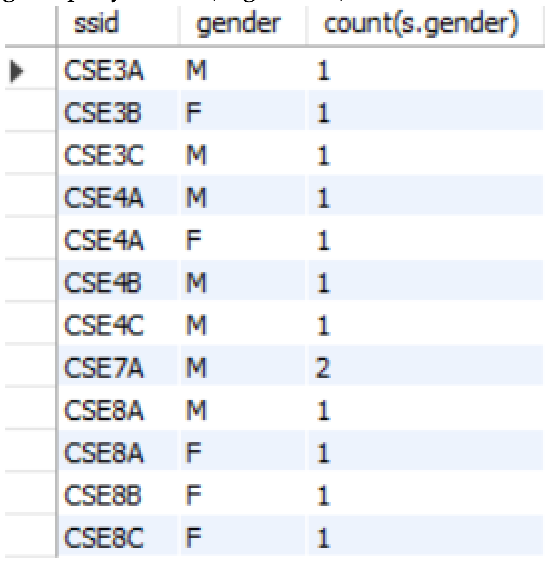
```
	usn varchar(10) not null,
	sname varchar(20) not null,
	address varchar(20) not null,
	phone varchar(10) not null,
	gender varchar(1) not null,
	primary key(usn)
);

create table SemSec(
	ssid varchar(5) not null,
	sem int not null,
	section varchar(1) not null,
	primary key(ssid)
);

create table Class(
	usn varchar(10) not null,
	ssid varchar(5) not null,
	primary key(usn,ssid),
	foreign key(usn) references Student(usn) on delete cascade,
	foreign key(ssid) references SemSec(ssid) on delete cascade
);

create table Subject(
	subcode varchar(6) not null,
	title varchar(10) not null,
	sem int not null,
	credits int not null,
	primary key(subcode)
);

create table IAMarks(
	usn varchar(10) not null,
	subcode varchar(6) not null,
	ssid varchar(5) not null,
	test1 int not null,
	test2 int not null,
	test3 int not null,
	final double not null default ((test1 + test2 + test3)/3),
	primary key(usn,subcode,ssid),
	foreign key(usn) references Student(usn) on delete cascade,
	foreign key(subcode) references Subject(subcode) on delete cascade,
	foreign key(ssid) references SemSec(ssid) on delete cascade
);

insert into Student
	values ('1RN13CS020','Akshay','Belagavi','8877881122','M'),
                    ('1RN13CS052','Sandhya','Bengaluru','7722829912','F'),
                    ('1RN13CS091','Teesha','Bengaluru','7712312312','F'),
                    ('1RN13CS066','Supriya','Mangaluru','8877881122','F'),
                    ('1RN14CS010','Abhay','Bengaluru','9900211201','M'),
                    ('1RN14CS032','Bhaskar','Bengaluru','9923211099','M'),
                    ('1RN15CS011','Ajay','Tumkur','9845091341','M'),
                    ('1RN15CS029','Chitra','Davangere','7696772121','F'),
                    ('1RN15CS045','Jeeva','Bellary','9944850121','M'),
                    ('1RN15CS091','Santhosh','Mangaluru','8812332201','M'),
                    ('1RN16CS045','Ismail','Kalburgi','9900232201','M'),
                    ('1RN16CS088','Sameera','Shimoga','9905542212','F'),
                    ('1RN16CS122','Vinayaka','Chikmagalur','8800880011','M'),
                    ('1RN14CS025','Asmi','Bengaluru','7894737377','F');

insert into SemSec values
          ('CSE8A',8,'A'),
          ('CSE8B',8,'B'),
          ('CSE8C',8,'C'),
          ('CSE7A',7,'A'),
          ('CSE7B',7,'B'),
          ('CSE7C',7,'C'),
          ('CSE6A',6,'A'),
          ('CSE6B',6,'B'),
          ('CSE6C',6,'C'),
          ('CSE5A',5,'A'),
          ('CSE5B',5,'B'),
          ('CSE5C',5,'C'),
          ('CSE4A',4,'A'),
          ('CSE4B',4,'C'),
          ('CSE4C',4,'C'),
          ('CSE3A',3,'A'),
          ('CSE3B',3,'B'),
          ('CSE3C',3,'C'),
          ('CSE2A',2,'A'),
          ('CSE2B',2,'B'),
          ('CSE2C',2,'C'),
          ('CSE1A',1,'A'),
          ('CSE1B',1,'B'),
          ('CSE1C',1,'C');

insert into Class values
          ('1RN13CS020','CSE8A'),
          ('1RN13CS052','CSE8A'),
          ('1RN13CS066','CSE8B'),
          ('1RN13CS091','CSE8C'),
          ('1RN14CS010','CSE7A'),
          ('1RN14CS032','CSE7A'),
          ('1RN15CS011','CSE4A'),
          ('1RN15CS029','CSE4A'),
          ('1RN15CS045','CSE4B'),
          ('1RN15CS091','CSE4C'),
          ('1RN16CS045','CSE3A'),
          ('1RN16CS088','CSE3B'),
          ('1RN16CS122','CSE3C');

insert into Subject values
          ('10CS81','ACA',8,4),
          ('10CS82','SSM',8,4),
          ('10CS83','NM',8,4),
          ('10CS84','CC',8,4),
          ('10CS85','PW',8,4),
          ('10CS71','OOAD',7,4),
          ('10CS72','ECS',7,4),
          ('10CS73','PTW',7,4),
          ('10CS74','DWDM',7,4),
          ('10CS75','JAVA',7,4),
          ('10CS76','SAN',7,4),
          ('15CS51','ME',5,4),
          ('15CS52','CN',5,4),
          ('15CS53','DBMS',5,4),
          ('15CS54','ATS',5,4),
          ('15CS55','JAVA',5,3),
          ('15CS56','AI',5,3),
          ('15CS41','M4',4,4),
          ('15CS42','SE',4,4),
          ('15CS43','DAA',4,4),
          ('15CS44','MPMC',4,4),
          ('15CS45','OOC',4,3),
          ('15CS46','DC',4,3),
          ('15CS31','M3',3,4),
          ('15CS32','ADE',3,4),
          ('15CS33','DSA',3,4),
          ('15CS34','CO',3,4),
          ('15CS35','USP',3,3),
          ('15CS36','DMS',3,3);

insert into IAMarks(usn,subcode,ssid,test1,test2,test3) values
          ('1RN13CS091','10CS81','CSE8C',15,16,18),
          ('1RN13CS091','10CS82','CSE8C',12,19,14),
          ('1RN13CS091','10CS83','CSE8C',19,15,20),
          ('1RN13CS091','10CS84','CSE8C',20,16,19),
          ('1RN13CS091','10CS85','CSE8C',15,15,12);

select * from IAMarks;
```

-- List all the student details studying in fourth semester 'C' section.
select s.usn,s.sname from Student s,Class c
where s.usn = c.usn and ssid = 'CSE4C';



usn	sname
1RN15CS091	Santhosh

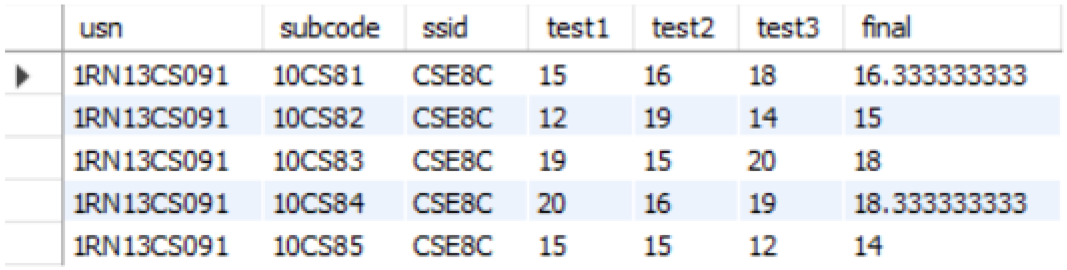
-- Compute the total number of male and female students in each semester and in each section.
select c.ssid,s.gender,count(s.gender) from Student s,Class c
where s.usn = c.usn
group by c.ssid,s.gender;



ssid	gender	count(s.gender)
CSE3A	M	1
CSE3B	F	1
CSE3C	M	1
CSE4A	M	1
CSE4A	F	1
CSE4B	M	1
CSE4C	M	1
CSE7A	M	2
CSE8A	M	1
CSE8A	F	1
CSE8B	F	1
CSE8C	F	1

-- Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.
create view student_details_view as
select * from IAMarks where usn = '1RN13CS091';

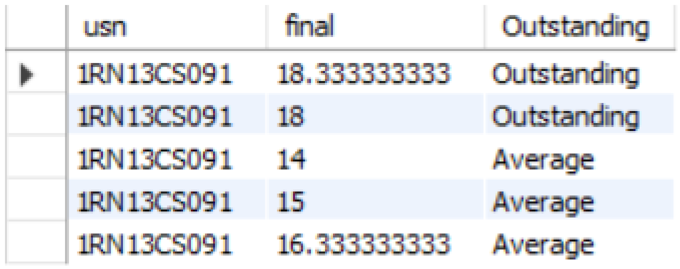
select * from student_details_view;



usn	subcode	ssid	test1	test2	test3	final
1RN13CS091	10CS81	CSE8C	15	16	18	16.333333333
1RN13CS091	10CS82	CSE8C	12	19	14	15
1RN13CS091	10CS83	CSE8C	19	15	20	18
1RN13CS091	10CS84	CSE8C	20	16	19	18.333333333
1RN13CS091	10CS85	CSE8C	15	15	12	14

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

```
select iam.usn,iam.final,'Outstanding'
from IAMarks iam,Student s,Class c
where iam.usn = s.usn and c.usn = s.usn
and final between 17 and 20 and c.ssid like 'CSE8_'
union
select iam.usn,iam.final,'Average'
from IAMarks iam,Student s,Class c
where iam.usn = s.usn and c.usn = s.usn
and final between 12 and 17 and c.ssid like 'CSE8_'
union
select iam.usn,iam.final,'Average'
from IAMarks iam,Student s,Class c
where iam.usn = s.usn and c.usn = s.usn
and final < 12 and c.ssid like 'CSE8_';
```



usn	final	Outstanding
1RN13CS091	18.333333333	Outstanding
1RN13CS091	18	Outstanding
1RN13CS091	14	Average
1RN13CS091	15	Average
1RN13CS091	16.333333333	Average