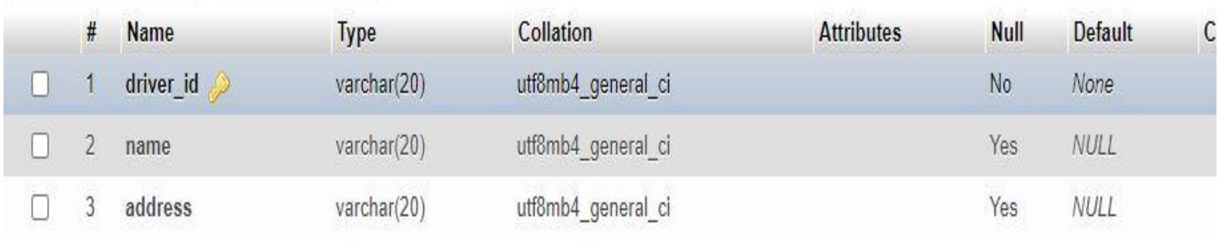
Consider the Insurance database given below. The primary keys are underlined and the data types are specified.

PERSON (driver-id #: String, name: String, address: String)
CAR (Regno: String, model: String, year: int)
ACCIDENT (report-number: int, date: date, location: String)
OWNS (driver-id #: String, Regno: String)
PARTICIPATED (driver-id: String, Regno: String, report-number: int, damage-amount: int)

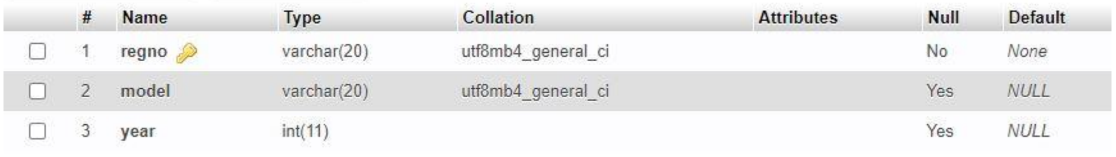
i. Create the above tables by properly specifying the primary keys and the foreign keys.

```
CREATE TABLE PERSON (
Driver_ID char(5) NOT    null,
Name varchar(30),
Address varchar(100),
primary key (Driver_ID));
```



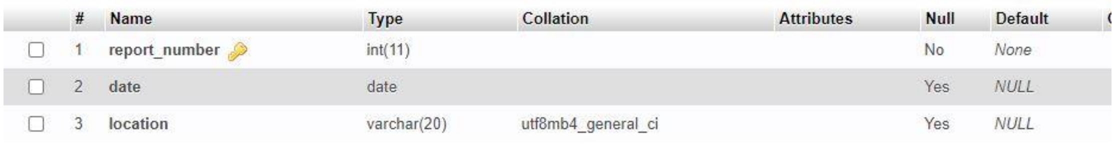
	#	Name	Type	Collation	Attributes	Null	Default	C
☐	1	driver_id	varchar(20)	utf8mb4_general_ci		No	*None*	
☐	2	name	varchar(20)	utf8mb4_general_ci		Yes	*NULL*	
☐	3	address	varchar(20)	utf8mb4_general_ci		Yes	*NULL*	

```
CREATE TABLE CAR (Reg_no char(4) NOT null, model varchar(10), Year int(4));
```



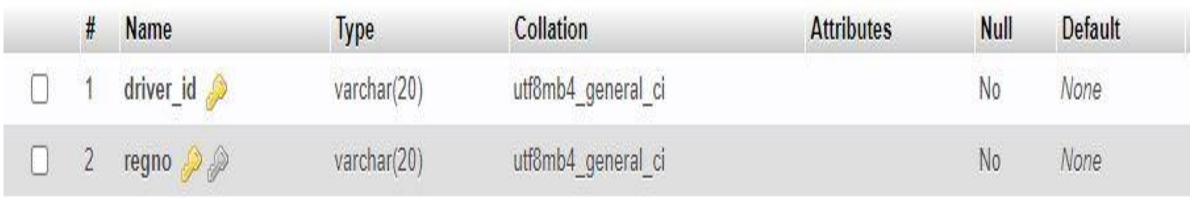
	#	Name	Type	Collation	Attributes	Null	Default
☐	1	regno	varchar(20)	utf8mb4_general_ci		No	*None*
☐	2	model	varchar(20)	utf8mb4_general_ci		Yes	*NULL*
☐	3	year	int(11)			Yes	*NULL*

```
CREATE table accident (Report_number int(5) NOT NULL,Date date,Location
  varchar(15));
```



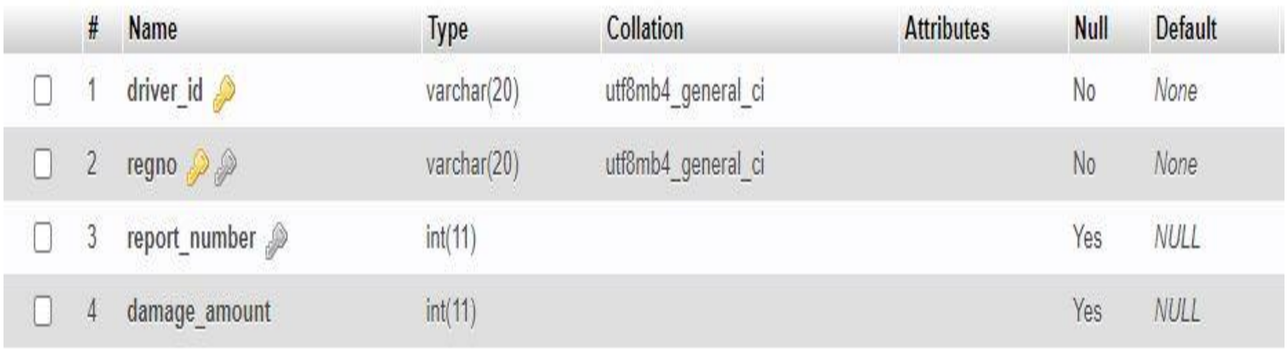
	#	Name	Type	Collation	Attributes	Null	Default
☐	1	report_number	int(11)			No	*None*
☐	2	date	date			Yes	*NULL*
☐	3	location	varchar(20)	utf8mb4_general_ci		Yes	*NULL*

CREATE table owns(Driver_ID char(5), Reg_no char(4),FOREIGN KEY (Driver_ID) REFERENCES person(Driver_ID));



	#	Name	Type	Collation	Attributes	Null	Default
☐	1	driver_id	varchar(20)	utf8mb4_general_ci		No	*None*
☐	2	regno	varchar(20)	utf8mb4_general_ci		No	*None*

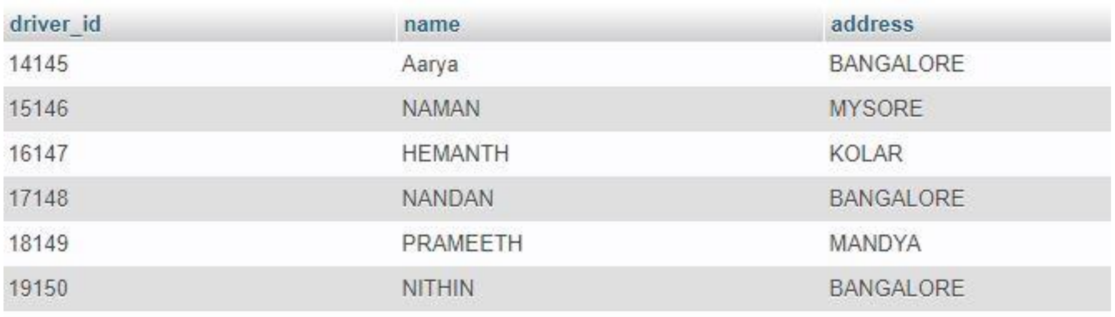
CREATE table participated(
Driver_ID char(5),
Reg_no char(4),
Report_number int(5),
Damage_amount int(7),
FOREIGN KEY (Driver_ID) REFERENCES person(Driver_ID));



	#	Name	Type	Collation	Attributes	Null	Default
☐	1	driver_id	varchar(20)	utf8mb4_general_ci		No	*None*
☐	2	regno	varchar(20)	utf8mb4_general_ci		No	*None*
☐	3	report_number	int(11)			Yes	*NULL*
☐	4	damage_amount	int(11)			Yes	*NULL*

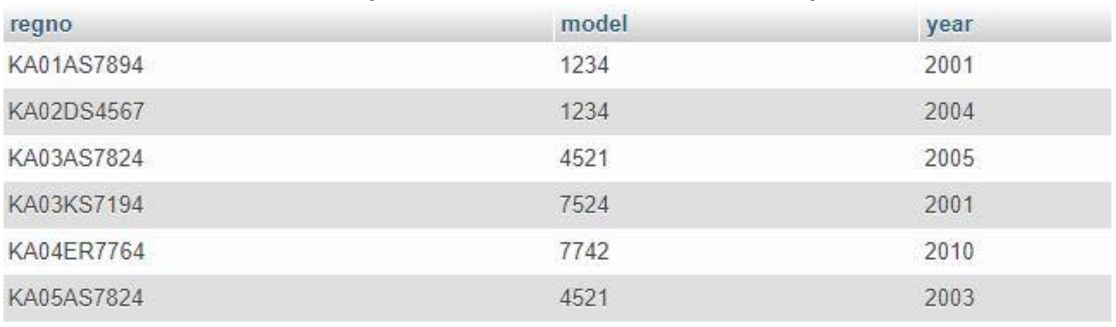
ii. Enter at least five tuples for each relation.

PERSON:
INSERT into person VALUES("1412","Aarya","bangalore");



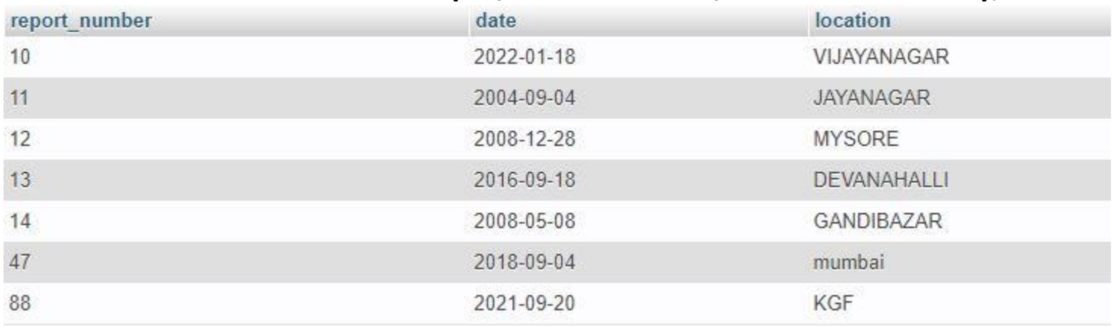
driver_id	name	address
14145	Aarya	BANGALORE
15146	NAMAN	MYSORE
16147	HEMANTH	KOLAR
17148	NANDAN	BANGALORE
18149	PRAMEETH	MANDYA
19150	NITHIN	BANGALORE

CAR:
insert INTO car VALUES("KA01AS7894","1234",2001);



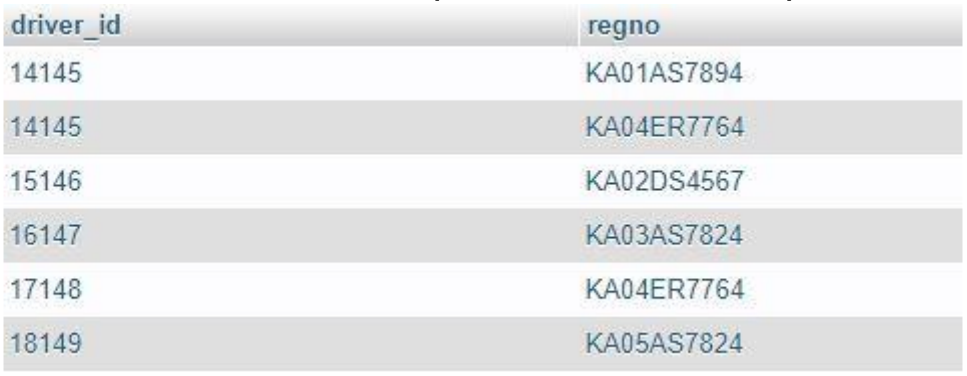
regno	model	year
KA01AS7894	1234	2001
KA02DS4567	1234	2004
KA03AS7824	4521	2005
KA03KS7194	7524	2001
KA04ER7764	7742	2010
KA05AS7824	4521	2003

Accident:
insert INTO accident VALUES(10,"2022-01-18","VIJAYANAGAR");



report_number	date	location
10	2022-01-18	VIJAYANAGAR
11	2004-09-04	JAYANAGAR
12	2008-12-28	MYSORE
13	2016-09-18	DEVANAHALLI
14	2008-05-08	GANDIBAZAR
47	2018-09-04	mumbai
88	2021-09-20	KGF

OWNS:
INSERT INTO owns VALUES ('14145', 'KA01AS7894');

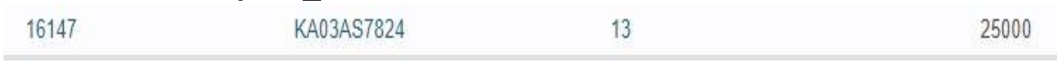


driver_id	regno
14145	KA01AS7894
14145	KA04ER7764
15146	KA02DS4567
16147	KA03AS7824
17148	KA04ER7764
18149	KA05AS7824

iii. Demonstrate how you

a.Update the damage amount for the car with a specific Regno in the accident with report number 12 to
25000.

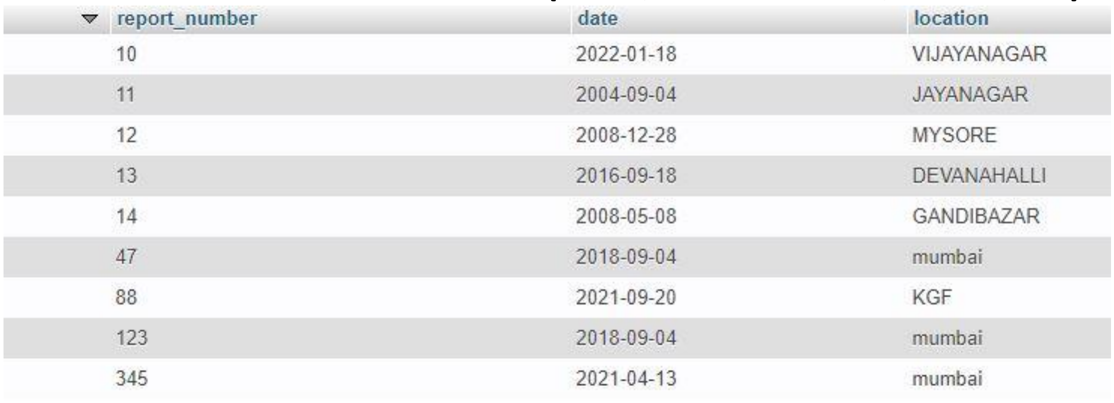
UPDATE participated SET damage_amount=25000 WHERE regno='KA03 AS7824' and report_number=13



16147	KA03AS7824	13	25000

b. Add a new accident to the database.

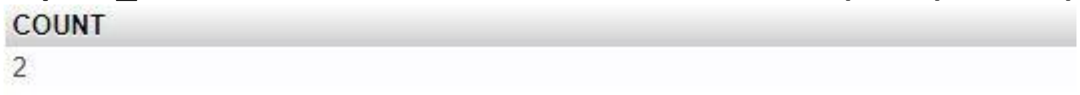
INSERT INTO accident VALUES ('345', '2021-04-13', 'mumbai');



report_number	date	location
10	2022-01-18	VIJAYANAGAR
11	2004-09-04	JAYANAGAR
12	2008-12-28	MYSORE
13	2016-09-18	DEVANAHALLI
14	2008-05-08	GANDIBAZAR
47	2018-09-04	mumbai
88	2021-09-20	KGF
123	2018-09-04	mumbai
345	2021-04-13	mumbai

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

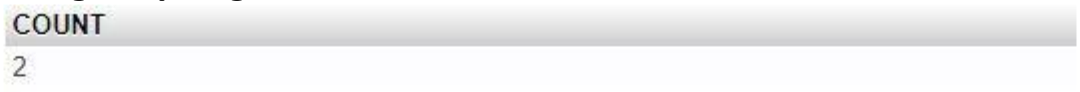
SELECT count(driver_id) AS COUNT
from PARTICIPATED WHERE report_number IN (SELECT report_number FROM ACCIDENT WHERE YEAR (date)=2008)



COUNT
2

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

select count(*) from car c,participated p where c.regno=p.regno and c.model='1234';



COUNT
2

Consider the following database for a banking enterprise.
BRANCH (branch-name: String, branch-city: String, assets: real)
ACCOUNTS (accno: int, branch-name: String, balance: real)
DEPOSITOR (customer-name: String, customer-street: String, customer-city: String)
LOAN (loan-number: int, branch-name: String, amount: real)
BORROWER (customer-name: String, loan-number: int)

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

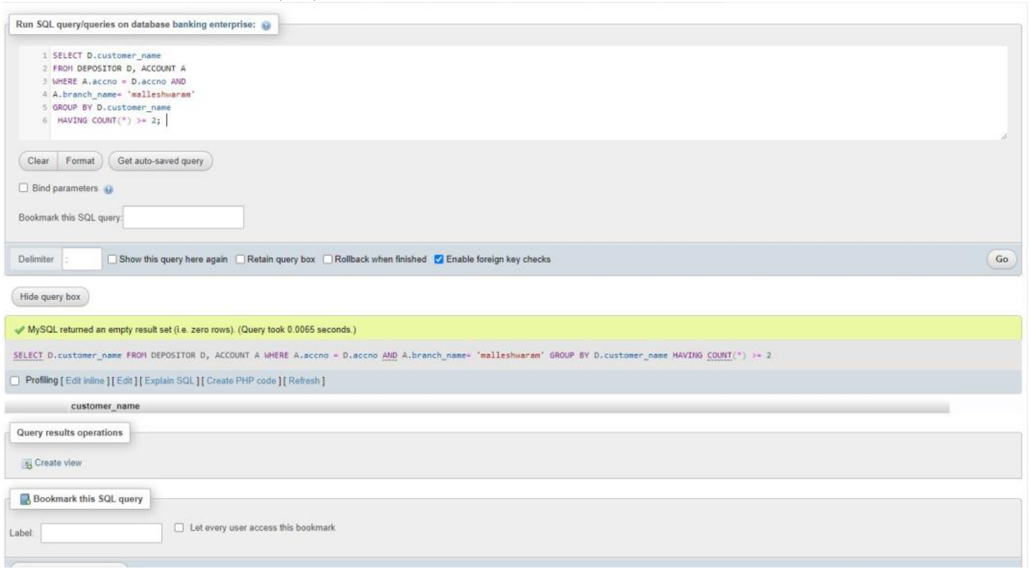
```
create table branch(branch_name varchar(30),branch_city
varchar(20),assets int,primary key(branch_name));
 create table account(accno int,branch_name varchar(20),balance
int,primary key(accno));


create table customer(customer_name
varchar(20),customer_street varchar(20),customer_city
varchar(20),primary key(customer_name));


 create table depositor(customer_name varchar(20),accno
int,primary key(customer_name,accno),foreign
key(customer_name) references customer(customer_name),foreign
key(accno) references account(accno) on delete cascade);


create table loan(loan_number int,branch_name
varchar(20),amount int,primary key(loan_number),foreign
key(branch_name) references branch(branch_name));



 create table borrower(customer_name varchar(20),loan_number
int,primary key(customer_name,loan_number),foreign
key(customer_name) references customer(customer_name),foreign
key(loan_number) references loan(loan_number));
```

ii.Enter at least five tuples for each relation.

```
INSERT INTO customer  VALUES ('aarya', 'rajajinagar',
'bangalore');
INSERT INTO account  VALUES ('123123', 'malleshwaram',
'30000');
INSERT INTO depositor VALUES ('aarya', '576124');
INSERT INTO loan  VALUES ('87', 'malleshwaram', '70000000');
INSERT INTO borrower VALUES ('aarya', '91');
```

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

```
SELECT D.customer_name
FROM DEPOSITOR D, ACCOUNT A
WHERE A.accno = D.accno AND
A.branch_name= 'malleshwaram'
GROUP BY D.customer_name
 HAVING COUNT(*) >= 2;
```



Run SQL query/queries on database banking enterprise:

```
SELECT D.customer_name
FROM DEPOSITOR D, ACCOUNT A
WHERE A.accno = D.accno AND
A.branch_name= 'malleshwaram'
GROUP BY D.customer_name
 HAVING COUNT(*) >= 2;
```

Delimiter ; — ☐ Show this query here again ☐ Retain query box ☐ Rollback when finished ☑ Enable foreign key checks

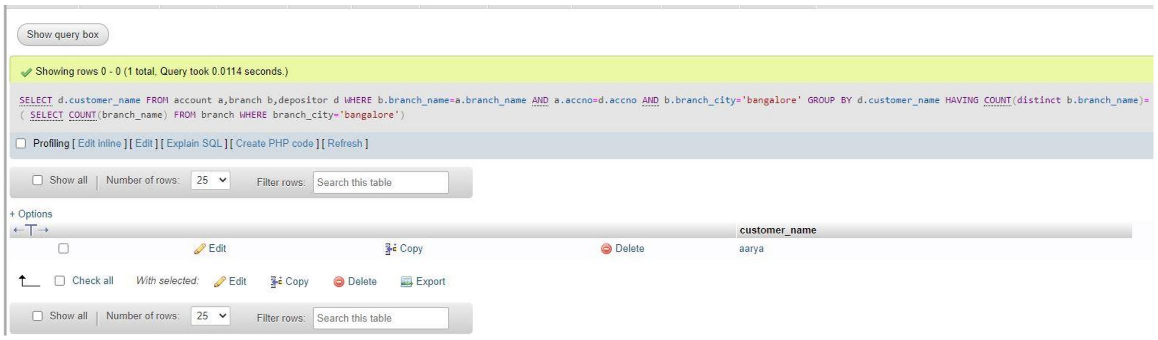
✓ MySQL returned an empty result set (i.e. zero rows). (Query took 0.0065 seconds.)

```
SELECT D.customer_name FROM DEPOSITOR D, ACCOUNT A WHERE A.accno = D.accno AND A.branch_name= 'malleshwaram' GROUP BY D.customer_name HAVING COUNT(*) >= 2
```

customer_name

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

```
SELECT d.customer_name
FROM account a,branch b,depositor d
WHERE b.branch_name=a.branch_name AND
a.accno=d.accno AND
b.branch_city='bangalore'
GROUP BY d.customer_name
HAVING COUNT(distinct b.branch_name)=(
SELECT COUNT(branch_name)
FROM branch
WHERE branch_city='bangalore');
```



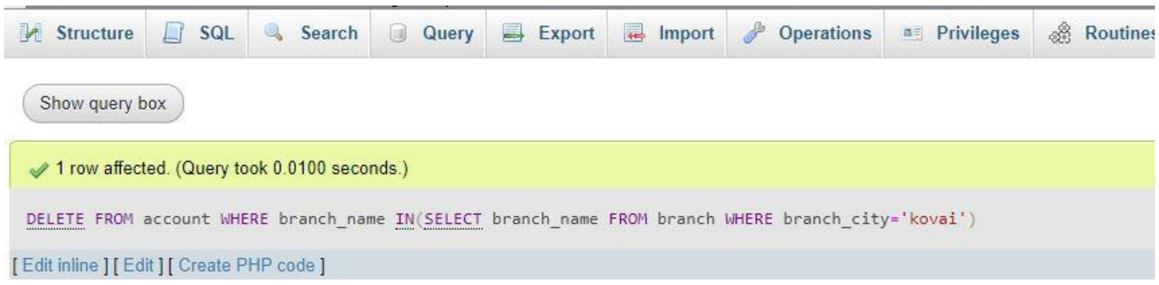
```
Showing rows 0 - 0 (1 total, Query took 0.0114 seconds.)

SELECT d.customer_name FROM account a,branch b,depositor d WHERE b.branch_name=a.branch_name AND a.accno=d.accno AND b.branch_city='bangalore' GROUP BY d.customer_name HAVING COUNT(distinct b.branch_name)=
( SELECT COUNT(branch_name) FROM branch WHERE branch_city='bangalore')
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

				customer_name
☐	Edit	Copy	Delete	aarya

Check all With selected: Edit Copy Delete Export
Show all | Number of rows: 25 Filter rows: Search this table

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


Structure SQL Search Query Export Import Operations Privileges Routines
Show query box

```
1 row affected. (Query took 0.0100 seconds.)

DELETE FROM account WHERE branch_name IN(SELECT branch_name FROM branch WHERE branch_city='kovai')
```

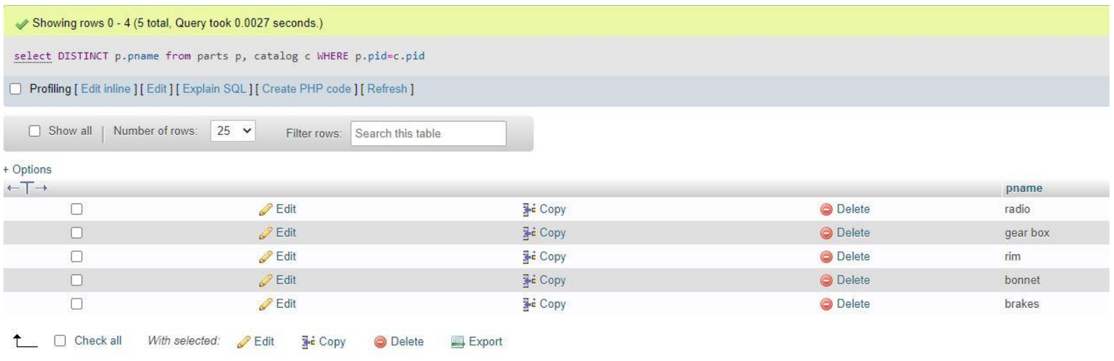
[Edit inline] [Edit] [Create PHP code]

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:

```
create TABLE suppliers(sid int(4),sname varchar(20),address
varchar(100),CONSTRAINT ID PRIMARY key(sid))

CREATE TABLE parts(pid int,pname varchar(20), color
varchar(10),CONSTRAINT PID PRIMARY KEY(pid))

CREATE table catalog(sid int,pid int,cosr real,CONSTRAINT F_sid
FOREIGN KEY(sid) REFERENCES suppliers(sid), CONSTRAINT F_pid
FOREIGN KEY(pid) REFERENCES parts(pid));


insert into suppliers values(123,'aarya','rajajinagar')
INSERT INTO parts VALUES ('1', 'radio', 'grey');
INSERT INTO catalog VALUES ('123', '1', '10000');
```

i. 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
```



Showing rows 0 - 4 (5 total, Query took 0.0027 seconds.)

```
select DISTINCT p.pname from parts p, catalog c WHERE p.pid=c.pid
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

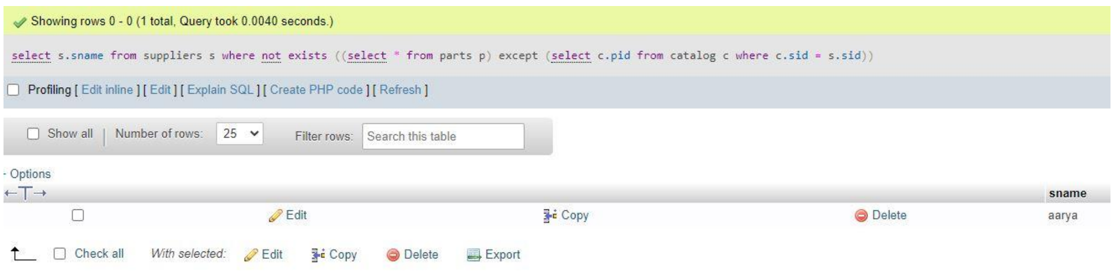
+ Options

				pname
☐	Edit	Copy	Delete	radio
☐	Edit	Copy	Delete	gear box
☐	Edit	Copy	Delete	rim
☐	Edit	Copy	Delete	bonnet
☐	Edit	Copy	Delete	brakes

Check all With selected: Edit Copy Delete Export

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

```
select s.sname from suppliers s where not exists ((select * from
parts p) except (select c.pid from catalog c where c.sid = s.sid))
```



Showing rows 0 - 0 (1 total, Query took 0.0040 seconds.)

```
select s.sname from suppliers s where not exists ((select * from parts p) except (select c.pid from catalog c where c.sid = s.sid))
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

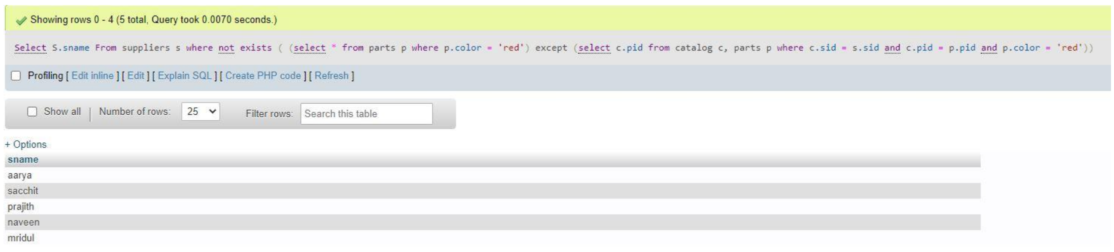
- Options

				sname
☐	Edit	Copy	Delete	aarya

Check all With selected: Edit Copy Delete Export

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

Select S.sname From suppliers s where not exists ((select * from parts p where p.color = 'red') except (select c.pid from catalog c, parts p where c.sid = s.sid and c.pid = p.pid and p.color = 'red'))



Showing rows 0 - 4 (5 total, Query took 0.0070 seconds.)

```
Select S.sname From suppliers s where not exists ( (select * from parts p where p.color = 'red') except (select c.pid from catalog c, parts p where c.sid = s.sid and c.pid = p.pid and p.color = 'red'))
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

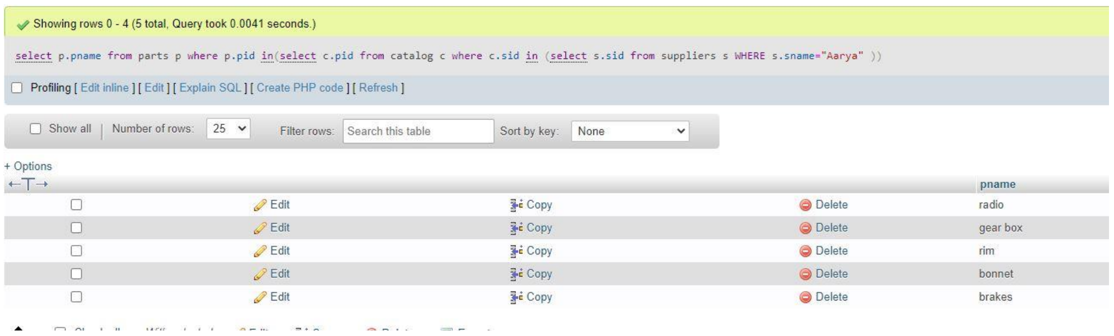
Show all | Number of rows: 25 Filter rows: Search this table

+ Options

sname
aarya
sacchit
prajith
naveen
mridul

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

select p.pname from parts p where p.pid in(select c.pid from c atalog c where c.sid in (select s.sid from suppliers s WHERE s .sname="AcmeWidget"))



Showing rows 0 - 4 (5 total, Query took 0.0041 seconds.)

```
select p.pname from parts p where p.pid in(select c.pid from catalog c where c.sid in (select s.sid from suppliers s WHERE s.sname="Aarya" ))
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

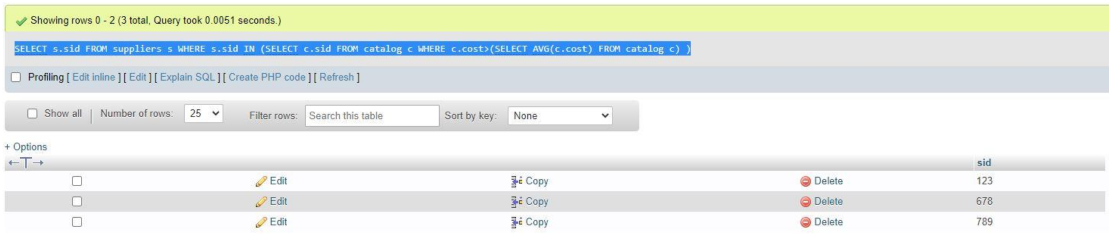
+ Options

				pname
☐	Edit	Copy	Delete	radio
☐	Edit	Copy	Delete	gear box
☐	Edit	Copy	Delete	rim
☐	Edit	Copy	Delete	bonnet
☐	Edit	Copy	Delete	brakes

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 s.sid FROM suppliers s WHERE s.sid IN

```
(SELECT c.sid FROM catalog c WHERE c.cost>(SELECT AVG(c.cost)
FROM catalog c) )
```


Showing rows 0 - 2 (3 total, Query took 0.0051 seconds.)

```
SELECT s.sid FROM suppliers s WHERE s.sid IN (SELECT c.sid FROM catalog c WHERE c.cost>(SELECT AVG(c.cost) FROM catalog c) )
```

sid
123
678
789

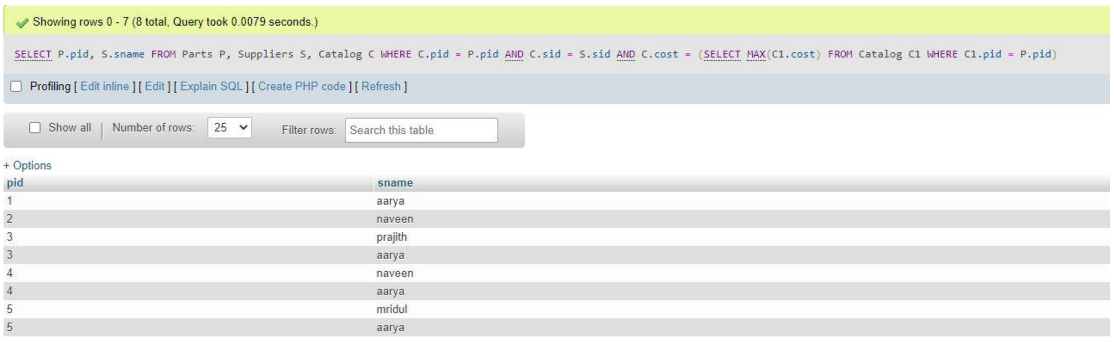
Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
Edit Copy Delete

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

```
select s.sname, p.pname  from suppliers s,parts p where s.sid
in(SELECT c.sid from catalog c where c.cost)
```

```
SELECT P.pid, S.sname FROM Parts P, Suppliers S, Catalog C WHERE
C.pid = P.pid AND C.sid = S.sid AND C.cost = (SELECT MAX (C1.cost)
FROM Catalog C1 WHERE C1.pid = P.pid)
```

```
SELECT P.pid, S.sname FROM Parts P, Suppliers S,catalog c WHERE
c.pid=p.pid AND c.sid=s.sid AND s.sid IN (SELECT c2.sid from
catalog c2 where c2.pid=p.pid AND c2.pid in (select max(c1.pid)
FROM catalog c1 group by c1.pid ))
```


Showing rows 0 - 7 (8 total, Query took 0.0079 seconds.)

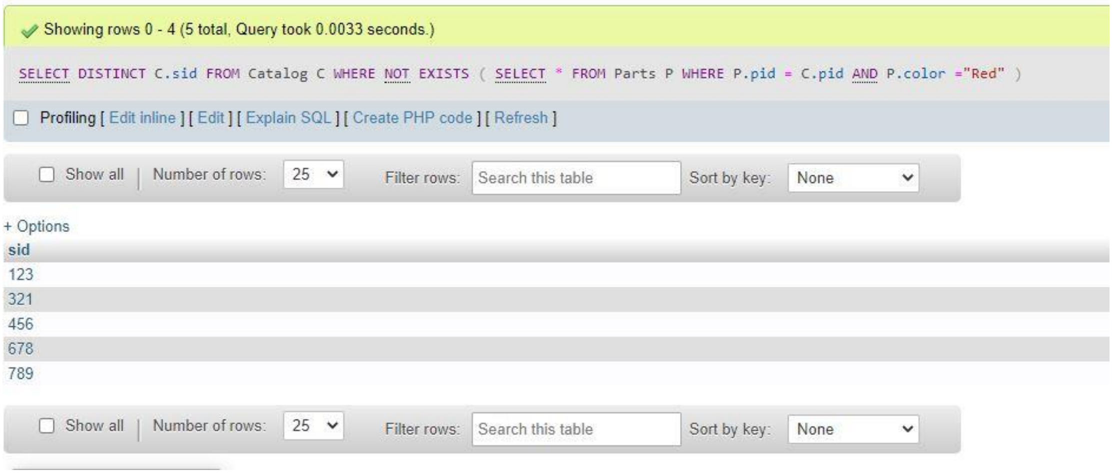
```
SELECT P.pid, S.sname FROM Parts P, Suppliers S, Catalog C WHERE C.pid = P.pid AND C.sid = S.sid AND C.cost = (SELECT MAX(C1.cost) FROM Catalog C1 WHERE C1.pid = P.pid)
```

pid	sname
1	aarya
2	naveen
3	prajith
3	aarya
4	naveen
4	aarya
5	mridul
5	aarya

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table
+ Options

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

```
SELECT DISTINCT C.sid FROM Catalog C WHERE NOT EXISTS ( SELECT *
FROM Parts P WHERE P.pid = C.pid AND P.color ="Red" )
```


Showing rows 0 - 4 (5 total, Query took 0.0033 seconds.)

```
SELECT DISTINCT C.sid FROM Catalog C WHERE NOT EXISTS ( SELECT * FROM Parts P WHERE P.pid = C.pid AND P.color ="Red" )
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

sid
123
321
456
678
789

Show all | Number of rows: 25
Filter rows: Search this table
Sort by key: None

LAB Program-4 (Student Faculty Database):
Upload the document(Queries with output screenshot)/github link here

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)

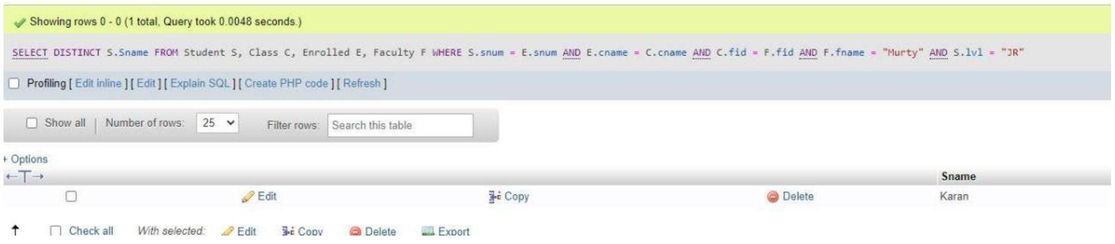
FACULTY(fid: integer, fname: string, deptid: integer)

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.

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

```
 SELECT DISTINCT S.Sname
FROM Student S, Class C, Enrolled E, Faculty F
WHERE S.snum = E.snum AND E.cname = C.cname AND C.fid = F.fid AND
F. fname = ‘Murty’ AND S.lvl = ‘JR’
```



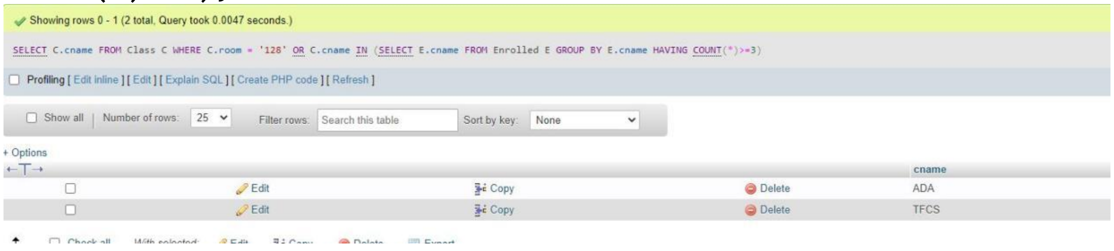
```
Showing rows 0 - 0 (1 total, Query took 0.0048 seconds.)

SELECT DISTINCT S.Sname FROM Student S, Class C, Enrolled E, Faculty F WHERE S.snum = E.snum AND E.cname = C.cname AND C.fid = F.fid AND F.fname = "Murty" AND S.lvl = "JR"
```

Sname
Karan

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
+ Options
Edit
Copy
Delete
Check all
With selected:
Edit
Copy
Delete
Export

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

```
SELECT C.cname FROM Class C WHERE C.room = '128' OR C.cname IN
(SELECT E.cname FROM Enrolled E GROUP BY E.cname HAVING
COUNT(*)>=3);
```



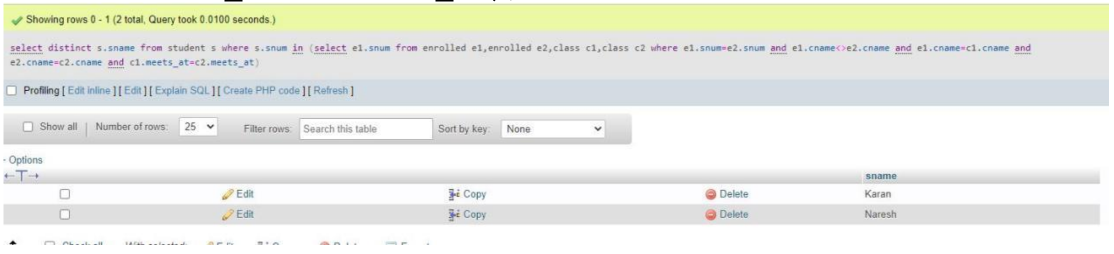
```
Showing rows 0 - 1 (2 total, Query took 0.0047 seconds.)

SELECT C.cname FROM Class C WHERE C.room = '128' OR C.cname IN (SELECT E.cname FROM Enrolled E GROUP BY E.cname HAVING COUNT(*)>=3)
```

				cname
☐	Edit	Copy	Delete	ADA
☐	Edit	Copy	Delete	TFCS

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

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

```
select distinct s.sname     from student s
     where s.snum in (select e1.snum
     from enrolled e1,enrolled e2,class c1,class c2
     where e1.snum=e2.snum and e1.cname<>e2.cname and
     e1.cname=c1.cname and e2.cname=c2.cname and
     c1.meets_at=c2.meets_at);
```



```
Showing rows 0 - 1 (2 total, Query took 0.0100 seconds.)

select distinct s.sname from student s where s.snum in (select e1.snum from enrolled e1,enrolled e2,class c1,class c2 where e1.snum=e2.snum and e1.cname<>e2.cname and e1.cname=c1.cname and
e2.cname=c2.cname and c1.meets_at=c2.meets_at)
```

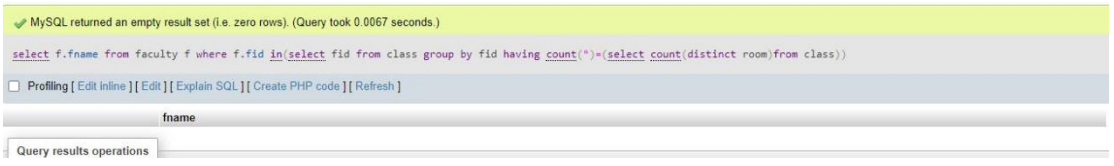
				sname
☐	Edit	Copy	Delete	Karan
☐	Edit	Copy	Delete	Naresh

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all
Number of rows: 25
Filter rows: Search this table
Sort by key: None
Options

iv.
Find

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

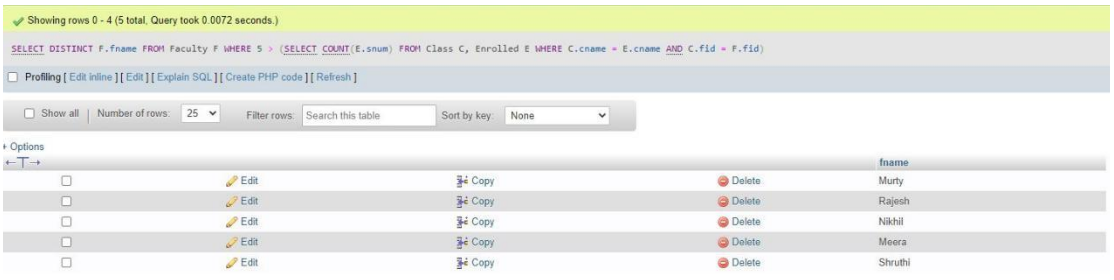
```
select f.fname
    from faculty f
     where f.fid in(select fid from class
     group by fid having count(*)=(select count(distinct room)from
class));
```



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0067 seconds.)

select f.fname from faculty f where f.fid in(select fid from class group by fid having count(*)=(select count(distinct room)from class))

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

fname

Query results operations
```

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

```
SELECT DISTINCT F.fname FROM Faculty F WHERE 5 > (SELECT COUNT
(E.snum) FROM Class C, Enrolled E WHERE C.cname = E.cname AND
C.fid = F.fid)
```

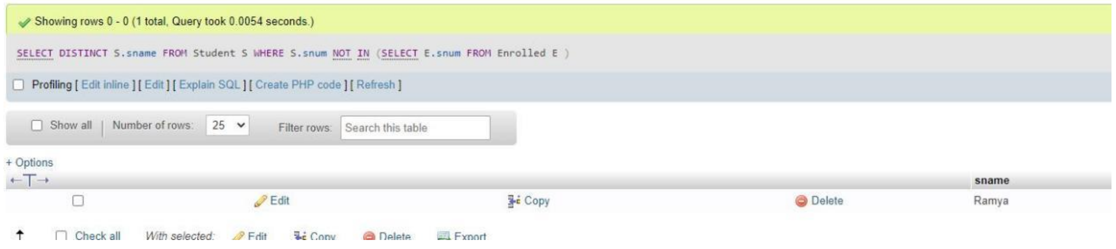


```
Showing rows 0 - 4 (5 total, Query took 0.0072 seconds.)

SELECT DISTINCT F.fname FROM Faculty F WHERE 5 > (SELECT COUNT(E.snum) FROM Class C, Enrolled E WHERE C.cname = E.cname AND C.fid = F.fid)

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25   Filter rows: Search this table   Sort by key: None

+ Options
```

				fname
☐	Edit	Copy	Delete	Murty
☐	Edit	Copy	Delete	Rajesh
☐	Edit	Copy	Delete	Nikhil
☐	Edit	Copy	Delete	Meera
☐	Edit	Copy	Delete	Shruthi

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

```
 SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum NOT IN (SELECT E.snum
FROM Enrolled E );
```

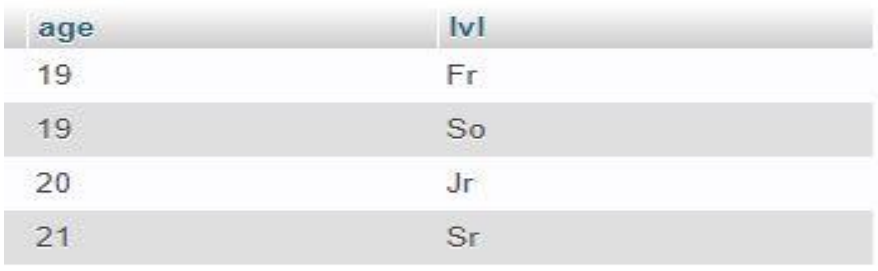


```
Showing rows 0 - 0 (1 total, Query took 0.0054 seconds.)

SELECT DISTINCT S.sname FROM Student S WHERE S.snum NOT IN (SELECT E.snum FROM Enrolled E )

Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

Show all | Number of rows: 25   Filter rows: Search this table

+ Options
```

				sname
☐	Edit	Copy	Delete	Ramya

vii.
For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

```
SELECT S.age, S.level FROM Student S GROUP BY S.age, S.level, HAVING
S.level IN (SELECT S1.level FROM Student S1 WHERE S1.age = S.age
GROUP BY S1.level, S1.age HAVING COUNT (*) >= ALL (SELECT COUNT
(*) FROM Student S2 WHERE s1.age = S2.age GROUP BY S2.level,
S2.age))
```



age	lvl
19	Fr
19	So
20	Jr
21	Sr

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)

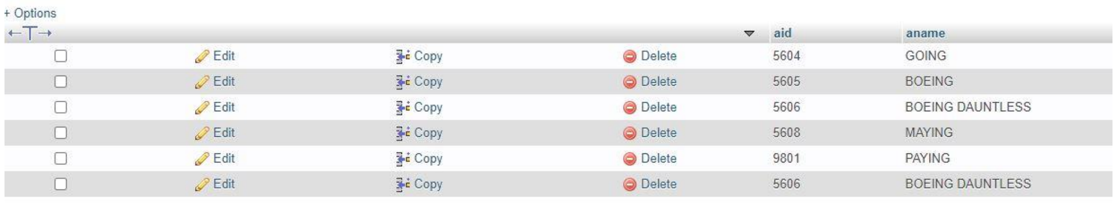
Note that the Employees relation
describes pilots and other kinds of employees as well; Every pilot is certified
for some aircraft, and only pilots are certified to fly.

Write each of the following queries
in SQL.

```
create table flight( fno int,ffrom varchar(20),fto
varchar(20),distance int(11),departs time,arrives
time,price float , primary key(fno));

create table aircraft(aid int,aname varchar(20),crange
int,CONSTRAINT aid_pk primary key(aid));

create table employees(eid int, ename varchar(20),salary
int,constraint eid_pk primary key(eid));

create table certified(eid int,aid int,primary key(eid,aid) ,
CONSTRAINT eid_fk foreign key(eid) references employees(eid),
CONSTRAINT aid_fk foreign key(aid) references aircraft(aid));
```

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

```
SELECT a.aid,a.aname from aircraft a,employees e,certified c where
a.aid=c.aid and e.eid=c.eid and e.salary>80000
```

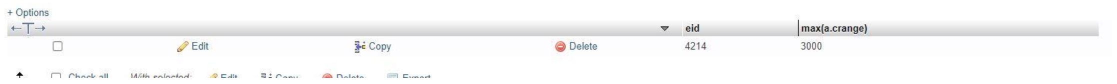


+ Options

				aid	aname
☐	Edit	Copy	Delete	5604	GOING
☐	Edit	Copy	Delete	5605	BOEING
☐	Edit	Copy	Delete	5606	BOEING DAUNTLESS
☐	Edit	Copy	Delete	5608	MAYING
☐	Edit	Copy	Delete	9801	PAYING
☐	Edit	Copy	Delete	5606	BOEING DAUNTLESS

ii.
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,max(a.crange) from employees e,aircraft a ,certified
c  where a.aid=c.aid and e.eid=c.eid GROUP by c.eid HAVING
COUNT(*)>3
```



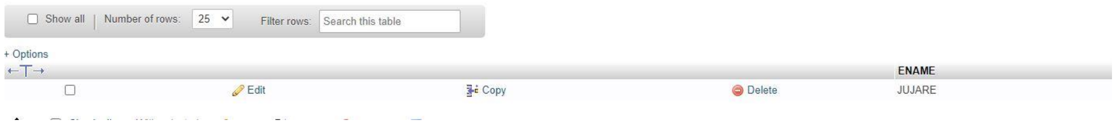
+ Options

				eid	max(a.crange)
☐	Edit	Copy	Delete	4214	3000

☐ Check all With selected: Edit Copy Delete Export

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

```
select distinct e.ename from employees e,certified c where
e.eid=c.eid and e.salary<(select min(price) from flight f where
f.ffrom='bangalore' and f.fto='frankfurt')
```



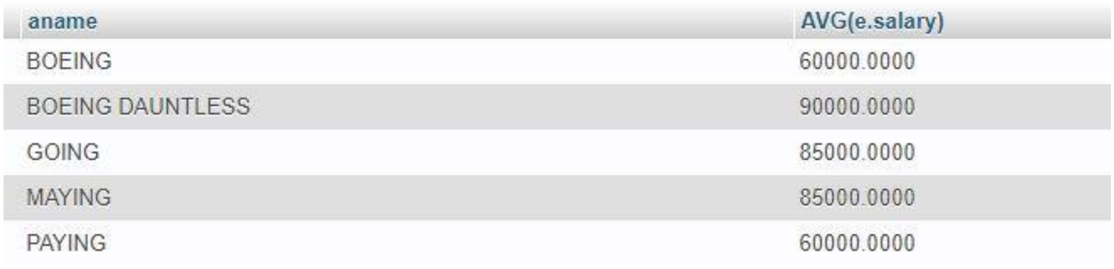
☐ Show all | Number of rows: 25 Filter rows: Search this table

+ Options

				ENAME
☐	Edit	Copy	Delete	JUJARE

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.

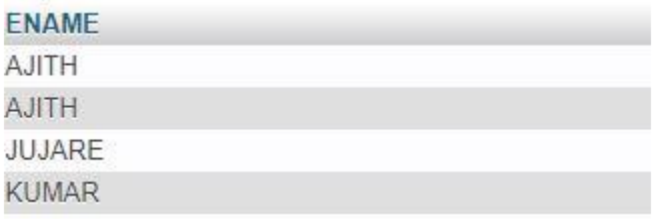
```
select a.aname,AVG(e.salary) from aircraft a,employees
e,certified c where a.aid=c.aid and e.eid=c.eid and a.crange>1000
group by a.aname
```



aname	AVG(e.salary)
BOEING	60000.0000
BOEING DAUNTLESS	90000.0000
GOING	85000.0000
MAYING	85000.0000
PAYING	60000.0000

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

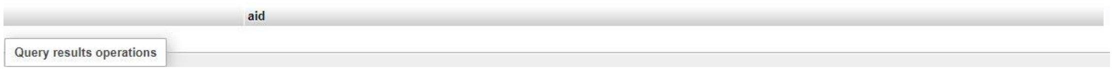
```
select ename from aircraft a,certified c,employees e where
a. aid=c.aid and c.eid=e.eid and a.aname like '%boeing%
```



ENAME
AJITH
AJITH
JUJARE
KUMAR

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

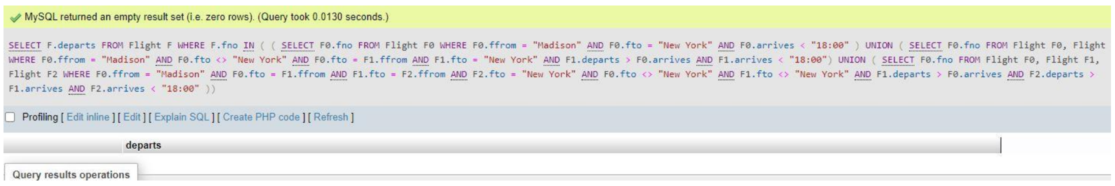
```
select a.aid from aircraft a where a.crange>=(select
min(f.distance) from flight f where f.ffrom='bangalore' and
f.fto='new delhi')
```



aid

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

```
SELECT F.departs FROM Flight F WHERE F.fno IN ( ( SELECT F0.fn
o FROM Flight F0 WHERE F0.ffrom = "Madison" AND F0.fto = "New
York" AND F0.arrives < "18:00" ) UNION ( SELECT F0.fno FROM Fl
ight F0, Flight F1 WHERE F0.ffrom = "Madison" AND F0.fto <> "N
ew York" AND F0.fto = F1.ffrom AND F1.fto = "New
York" AND F1.departs > F0.arrives AND F1.arrives < "18:00") UN
ION ( SELECT F0.fno FROM Flight F0, Flight F1, Flight F2 WHERE
 F0.ffrom = "Madison" AND F0.fto = F1.ffrom AND F1.fto = F2.ff
rom AND F2.fto = "New York" AND F0.fto <> "New
York" AND F1.fto <> "New
York" AND F1.departs > F0.arrives AND F2.departs > F1.arrives
AND F2.arrives < "18:00" ))
```



```
MySQL returned an empty result set (i.e. zero rows). (Query took 0.0130 seconds.)

SELECT F.departs FROM Flight F WHERE F.fno IN ( ( SELECT F0.fno FROM Flight F0 WHERE F0.ffrom = "Madison" AND F0.fto = "New York" AND F0.arrives < "18:00" ) UNION ( SELECT F0.fno FROM Flight F0, Flight
WHERE F0.ffrom = "Madison" AND F0.fto <> "New York" AND F0.fto = F1.ffrom AND F1.fto = "New York" AND F1.departs > F0.arrives AND F1.arrives < "18:00") UNION ( SELECT F0.fno FROM Flight F0, Flight F1,
Flight F2 WHERE F0.ffrom = "Madison" AND F0.fto = F1.ffrom AND F1.fto = F2.ffrom AND F2.fto = "New York" AND F0.fto <> "New York" AND F1.fto <> "New York" AND F1.departs > F0.arrives AND F2.departs >
F1.arrives AND F2.arrives < "18:00" ))
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
departs
Query results operations

Program 6 : Order Database

Consider the
following schema for Order Database:

SALESMAN (*Salesman_id,*
***Name, City, Commission*)**

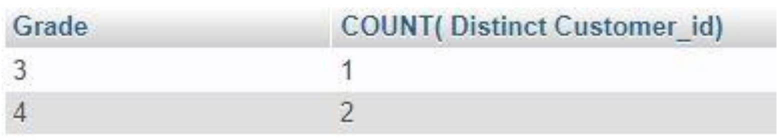
CUSTOMER (*Customer_id,*
***Cust_Name, City, Grade, Salesman_id*)**

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.

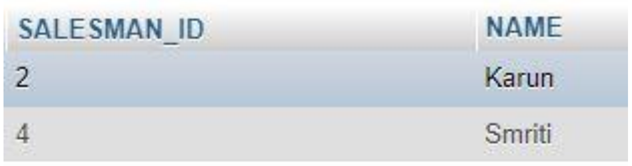
```
SELECT Grade, COUNT( Distinct Customer_id) FROM
 customer GROUP BY Grade HAVING Grade > (SELECT
 AVG(Grade) FROM customer WHERE City='BANGALORE
')
```



Grade	COUNT(Distinct Customer_id)
3	1
4	2

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

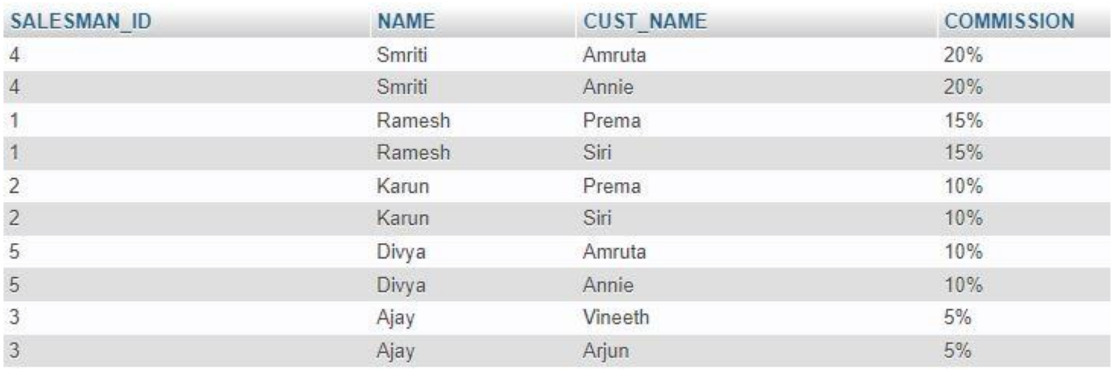
```
SELECT SALESMAN_ID, NAME FROM SALESMAN A WHERE 1
< (SELECT COUNT(*) FROM CUSTOMER WHERE
SALESMAN_ID=A.SALESMAN_ID);
```



SALESMAN_ID	NAME
2	Karun
4	Smriti

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

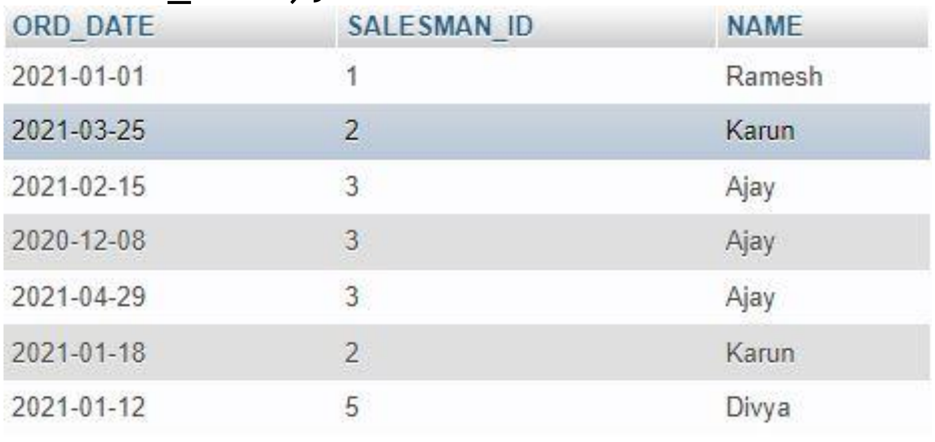
```
SELECT SALESMAN.SALESMAN_ID, NAME, CUST_NAME,
COMMISSION FROM SALESMAN, CUSTOMER WHERE
SALESMAN.CITY = CUSTOMER.CITY UNION SELECT
SALESMAN_ID, NAME, 'NO MATCH', COMMISSION FROM
SALESMAN WHERE NOT CITY = ANY (SELECT CITY FROM
CUSTOMER) ORDER BY 2 DESC;
```



SALESMAN_ID	NAME	CUST_NAME	COMMISSION
4	Smriti	Amruta	20%
4	Smriti	Annie	20%
1	Ramesh	Prema	15%
1	Ramesh	Siri	15%
2	Karun	Prema	10%
2	Karun	Siri	10%
5	Divya	Amruta	10%
5	Divya	Annie	10%
3	Ajay	Vineeth	5%
3	Ajay	Arjun	5%

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

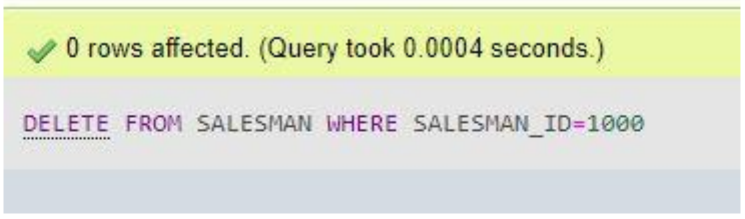
```
CREATE VIEW ELITSALESMAN AS SELECT B.ORD_DATE,
A.SALESMAN_ID, A.NAME FROM
SALESMAN A, ORDERS B WHERE A.SALESMAN_ID =
B.SALESMAN_ID AND B.PURCHASE_AMT=(SELECT
MAX(PURCHASE_AMT) FROM ORDERS C WHERE C.ORD_DATE
= B.ORD_DATE);
```



ORD_DATE	SALESMAN_ID	NAME
2021-01-01	1	Ramesh
2021-03-25	2	Karun
2021-02-15	3	Ajay
2020-12-08	3	Ajay
2021-04-29	3	Ajay
2021-01-18	2	Karun
2021-01-12	5	Divya

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

```
DELETE FROM SALESMAN WHERE SALESMAN_ID=1000
```



```
0 rows affected. (Query took 0.0004 seconds.)

DELETE FROM SALESMAN WHERE SALESMAN_ID=1000
```

PROGRAM 7. BOOK DEALER DATABASE
The following tables are maintained by a book dealer:

AUTHOR(author-id: int, name: String, city: String, country: String)

PUBLISHER(publisher-id: int, name: String, city: String, country: String)

CATALOG (book-id: int, title: String, author-id: int, publisher-id: int, category-id: int, year: int, price: int)

CATEGORY(category-id: int, description: String)

ORDER-DETAILS(order-no: int, book-id: int, quantity: int)

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

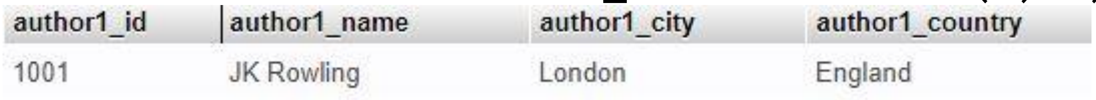
```
CREATE TABLE orderdetails1(
      order_id INT,
      book_id INT,
      quantity INT,
      PRIMARY KEY(order_id),
      FOREIGN      KEY(book_id)      REFERENCES
catalogue1(book_id));

CREATE TABLE publisher1 (
      publisher1_id INT,
      publisher1_name VARCHAR(20),
      publisher1_city VARCHAR(20),
      publisher1_country VARCHAR(20),
      PRIMARY KEY(publisher1_id));

CREATE TABLE category1 (
      category_id INT,
      description VARCHAR(30),
      PRIMARY KEY(category_id) );

CREATE TABLE catalogue1(
      book_id INT,
      book_title VARCHAR(30),
      author1_id INT,
      publisher1_id INT,
      category_id INT,
      year INT,
      price INT,
      PRIMARY KEY(book_id),
      FOREIGN    KEY(author1_id)     REFERENCES
author1(author1_id),
      FOREIGN   KEY(publisher1_id)   REFERENCES
publisher1(publisher1_id),
      FOREIGN    KEY(category_id)    REFERENCES
category1(category_id) );

 CREATE TABLE orderdetails1(
      order_id INT,
      book_id INT,
      quantity INT,
      PRIMARY KEY(order_id),
      FOREIGN      KEY(book_id)      REFERENCES
catalogue1(book_id));
```

ii. Enter at least five tuples for each relation.

```
INSERT INTO author1
(author1_id,author1_name,author1_city,author1_c
ountry) VALUES (1001,'JK
Rowling','London','England')

INSERT INTO publisher1
(publisher1_id,publisher1_name,publisher1_city,
publisher1_country) VALUES
(2001,'Bloomsbury','London','England')

INSERT INTO category1 (category_id,description)
VALUES
 (3001,'Fiction')

INSERT INTO catalogue1
(book_id,book_title,author1_id,publisher1_id,ca
tegory_id,year,price) VALUES (4001,'HP and Goblet
Of Fire',1001,2001,3001,2002,600)

INSERT INTO orderdetails1
(order_id,book_id,quantity) VALUES (5001,4001,5)
```

iii. Give the details of the authors who have 2 or more books in the catalog and the price of the books in the catalog and the year of publication is after 2000.

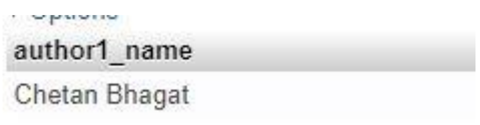
```
SELECT * FROM author1
          WHERE author1_id IN
          (SELECT  author1_id  FROM  catalogue1
WHERE
          year>2000 AND price>
          (SELECT AVG(price) FROM catalogue1)
          GROUP BY author1_id HAVING COUNT(*)>1)
```



author1_id	author1_name	author1_city	author1_country
1001	JK Rowling	London	England

iv. Find the author of the book which has maximum sales.

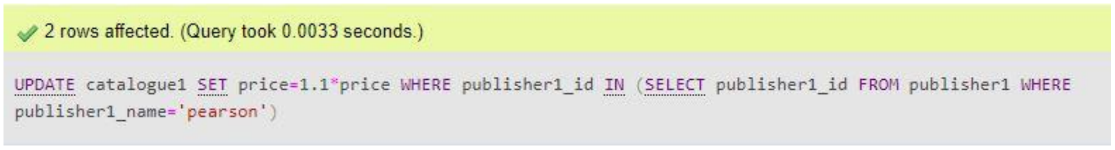
```
SELECT author1_name FROM author1 a,catalogue1 c
WHERE a.author1_id=c.author1_id AND book_id IN
(SELECT book_id FROM orderdetails1 WHERE
quantity=(SELECT MAX(quantity) FROM
orderdetails1))
```



author1_name
Chetan Bhagat

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

```
UPDATE catalogue1 SET price=1.1*price
WHERE publisher1_id IN (SELECT publisher1_id FROM
publisher1 WHERE publisher1_name='pearson')
```



```
2 rows affected. (Query took 0.0033 seconds.)

UPDATE catalogue1 SET price=1.1*price WHERE publisher1_id IN (SELECT publisher1_id FROM publisher1 WHERE
publisher1_name='pearson')
```

PROGRAM 8. STUDENT ENROLLMENT DATABASE
Consider the following database of student enrollment in courses and books adopted for each course.

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

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

ENROLL (regno: String, cname: String, 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.

```
CREATE TABLE student(
     regno VARCHAR(15),
     name VARCHAR(20),
     major VARCHAR(20),
     bdate DATE,
     PRIMARY KEY (regno) )

CREATE TABLE course(
     courseno INT,
     cname VARCHAR(20),
     dept VARCHAR(20),
     PRIMARY KEY (courseno) )

CREATE TABLE enroll(
     regno VARCHAR(15),
     courseno INT,
     sem INT(3),
     marks INT(4),
     PRIMARY KEY (regno,courseno),
     FOREIGN KEY (regno) REFERENCES student
(regno),
     FOREIGN KEY (courseno) REFERENCES course
(courseno) )

CREATE TABLE text(
     book_isbn INT(5),
     book_title VARCHAR(20),
     publisher VARCHAR(20),
     author VARCHAR(20),
     PRIMARY KEY (book_isbn) )

CREATE TABLE book_adoption(
     courseno INT,
     sem INT(3),
     book_isbn INT(5),
     PRIMARY KEY (courseno,book_isbn),
     FOREIGN KEY (courseno) REFERENCES course
(courseno),
     FOREIGN KEY (book_isbn) REFERENCES
text(book_isbn) )
```

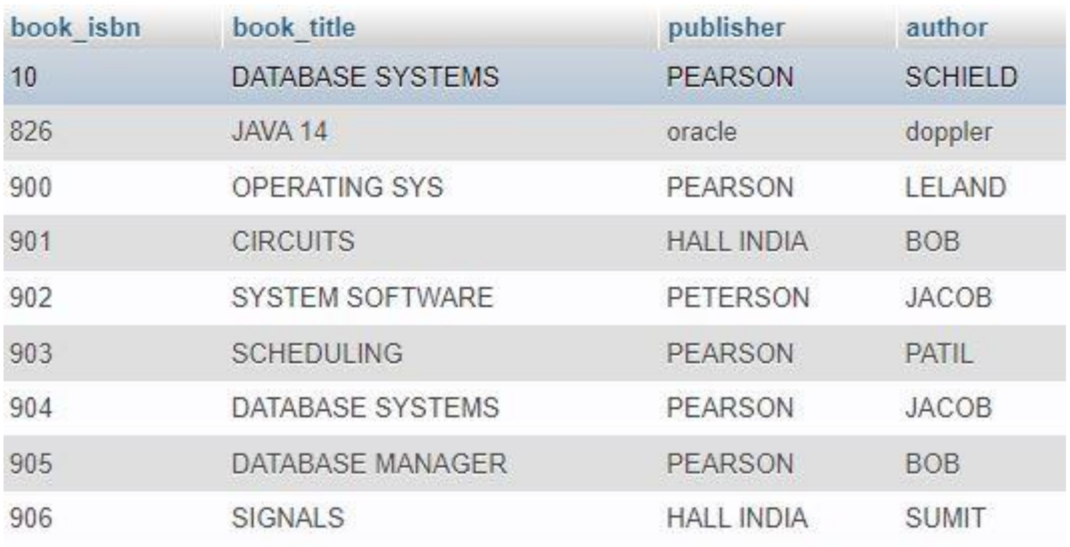
ii. Enter at least five tuples for each relation.

```
INSERT INTO student (regno,name,major,bdate)
VALUES ('1pe11cs002','b','sr','19930924')

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


INSERT INTO book_adoption (courseno,sem,book_isbn)
VALUES (111,5,900)
```

```
INSERT INTO enroll (regno,courseno,sem,marks)
VALUES ('1pe11cs002',114,5,100)

INSERT INTO text
(book_isbn,book_title,publisher,author) VALUES
 (10,'DATABASE SYSTEMS','PEARSON','SCHIELD')
```

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

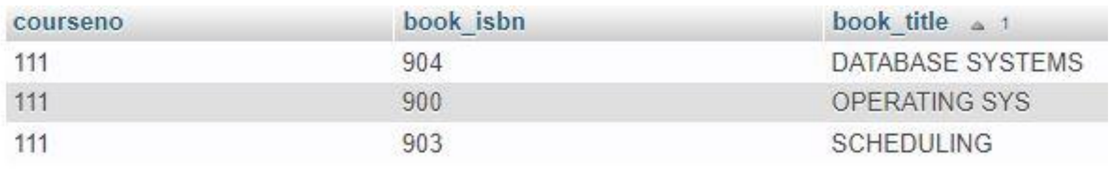
```
INSERT INTO `book_adoption` (`courseno`, `sem`,
`book_isbn`) VALUES ('114', '6', '10')

INSERT INTO `text` (`book_isbn`, `book_title`,
`publisher`, `author`) VALUES ('10’, 'DATABASE
SYSTEMS', 'PEARSON', 'SCHIELD')
```



book_isbn	book_title	publisher	author
10	DATABASE SYSTEMS	PEARSON	SCHIELD
826	JAVA 14	oracle	doppler
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

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.

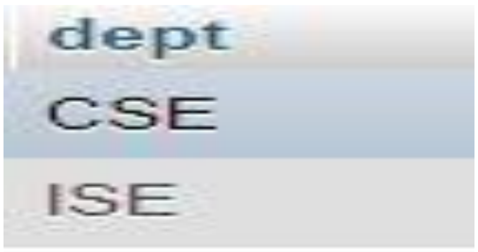
```
SELECT c.courseno,t.book_isbn,t.book_title
     FROM course c,book_adoption ba,text t
     WHERE c.courseno=ba.courseno
     AND ba.book_isbn=t.book_isbn
     AND c.dept='CSE'
     AND 2<(
     SELECT COUNT(book_isbn)
     FROM book_adoption b
     WHERE c.courseno=b.courseno)
     ORDER BY t.book_title
```



courseno	book_isbn	book_title ▲ 1
111	904	DATABASE SYSTEMS
111	900	OPERATING SYS
111	903	SCHEDULING

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

```
SELECT DISTINCT c.dept
     FROM course c
     WHERE c.dept IN
     ( SELECT c.dept
     FROM course c,book_adoption b,text t
     WHERE c.courseno=b.courseno
     AND t.book_isbn=b.book_isbn
     AND t.publisher='PEARSON')
```



dept
CSE
ISE

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)

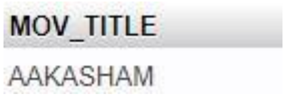
MOVIE_CAST(Act_id, Mov_id, Role)

RATING(Mov_id, Rev_Stars)

Write SQL queries to

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

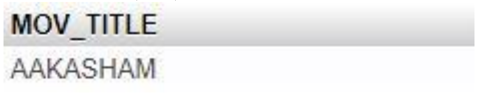
```
SELECT MOV_TITLE FROM MOVIES WHERE DIR_ID =
(SELECT DIR_ID FROM DIRECTOR WHERE
DIR_NAME='HITCHCOCK')
```



MOV_TITLE
AAKASHAM

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

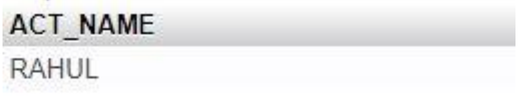
```
SELECT MOV_TITLE FROM MOVIES M,MOVIE_CAST MC WHERE
M.MOV_ID=MC.MOV_ID AND ACT_ID IN (SELECT ACT_ID
FROM MOVIE_CAST GROUP BY ACT_ID
HAVING COUNT(ACT_ID)>1) GROUP BY MOV_TITLE HAVING
COUNT(*)>1
```



MOV_TITLE
AAKASHAM

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

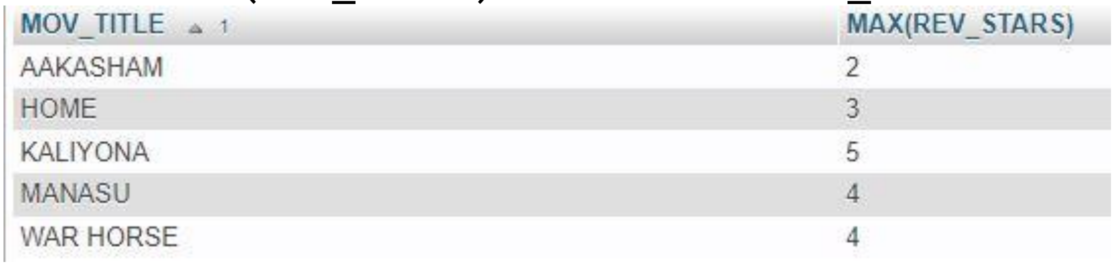
```
SELECT ACT_NAME FROM ACTOR A JOIN MOVIE_CAST C ON
A.ACT_ID=C.ACT_ID JOIN MOVIES M ON
C.MOV_ID=M.MOV_ID WHERE M.MOV_YEAR NOT BETWEEN
2000 AND 2015
```



ACT_NAME
RAHUL

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

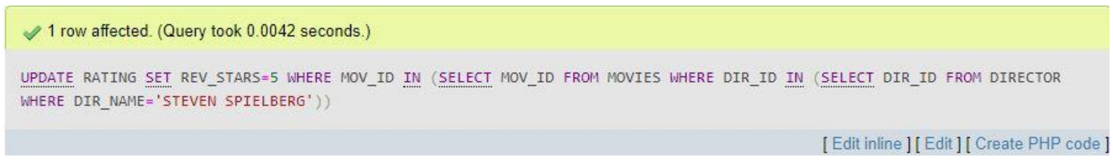
```
SELECT MOV_TITLE,MAX(REV_STARS) FROM MOVIES INNER
JOIN RATING USING (MOV_ID) GROUP BY MOV_TITLE
HAVING MAX(REV_STARS)>0 ORDER BY MOV_TITLE
```



MOV_TITLE	MAX(REV_STARS)
AAKASHAM	2
HOME	3
KALIYONA	5
MANASU	4
WAR HORSE	4

v. Update rating of all movies directed by 'Steven Spielberg' to 5.

```
UPDATE RATING SET REV_STARS=5 WHERE MOV_ID IN
(SELECT MOV_ID FROM MOVIES WHERE DIR_ID IN (SELECT
DIR_ID FROM DIRECTOR WHERE DIR_NAME='STEVEN
SPIELBERG'))
```



```
✓ 1 row affected. (Query took 0.0042 seconds.)

UPDATE RATING SET REV_STARS=5 WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES WHERE DIR_ID IN (SELECT DIR_ID FROM DIRECTOR
WHERE DIR_NAME='STEVEN SPIELBERG'))
```

PROGRAM 10:COLLEGE DATABASE

Consider the schema for College Database:

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

SEMSEC(SSID, Sem, Sec)

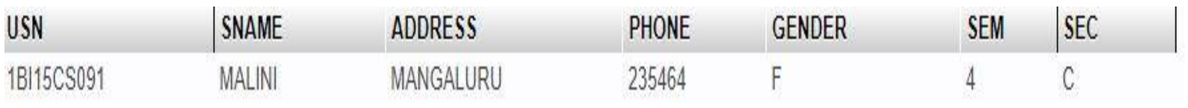
CLASS(USN, SSID)

SUBJECT(Subcode, Title, Sem, Credits)

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

Write SQL queries to
i. List all the student details studying in fourth semester 'C' section.

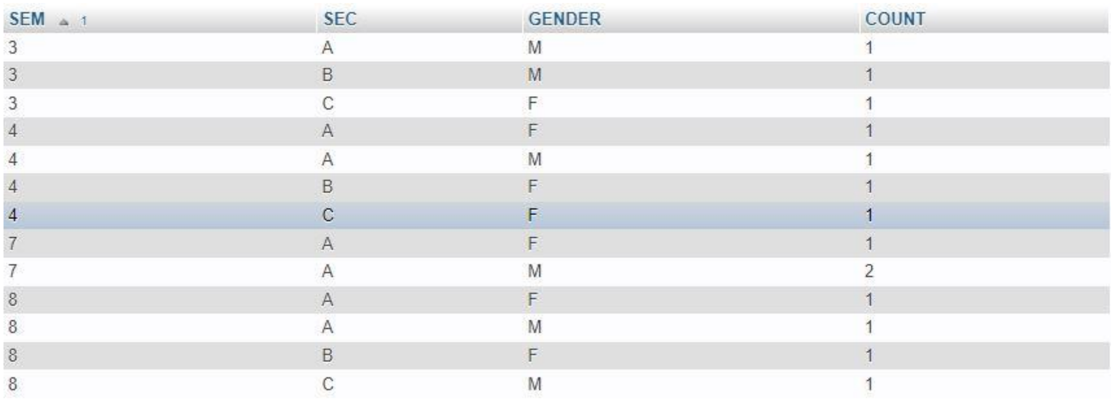
```
SELECT S.*, SS.SEM, SS.SEC FROM STUDENT S, SEMS
EC SS, CLASS C WHERE S.USN = C.USN AND SS.SSID
= C.SSID AND SS.SEM = 4 AND SS.SEC='C'
```



USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1BI15CS091	MALINI	MANGALURU	235464	F	4	C

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

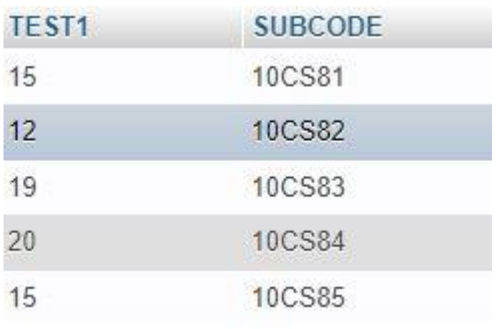
```
SELECT SS.SEM, SS.SEC, S.GENDER, COUNT(S.GENDER
) AS COUNT FROM STUDENT S, SEMSEC SS, CLASS C W
HERE S.USN = C.USN AND SS.SSID = C.SSID GROUP B
Y SS.SEM, SS.SEC, S.GENDER ORDER BY SEM
```



SEM	SEC	GENDER	COUNT
3	A	M	1
3	B	M	1
3	C	F	1
4	A	F	1
4	A	M	1
4	B	F	1
4	C	F	1
7	A	F	1
7	A	M	2
8	A	F	1
8	A	M	1
8	B	F	1
8	C	M	1

iii. Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.

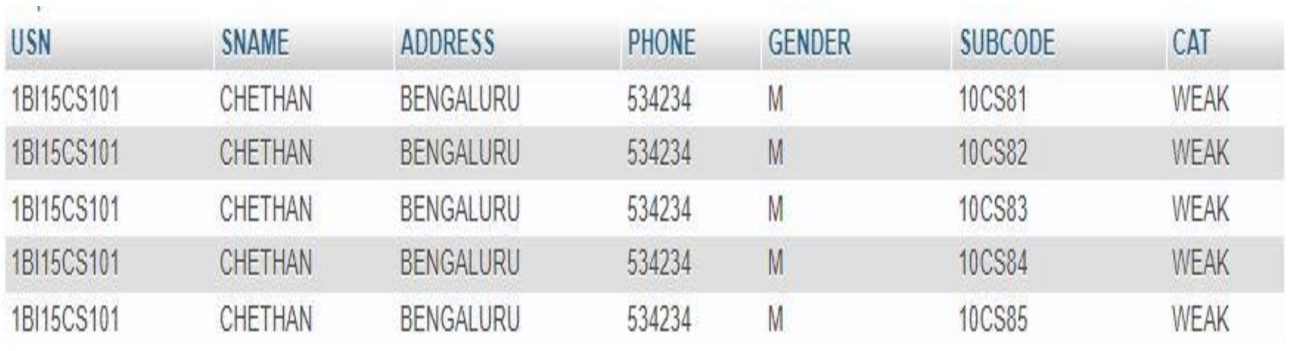
```
CREATE VIEW STUDENT_TEST1_MARKS_V AS SELECT TEST1,
SUBCODE FROM IAMARKS
WHERE USN = '1BI15CS101';

SELECT * FROM STUDENT_TEST1_MARKS_V;
```



TEST1	SUBCODE
15	10CS81
12	10CS82
19	10CS83
20	10CS84
15	10CS85

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

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER
, IA.SUBCODE, (CASE WHEN IA.FINALIA BETWEEN 17
AND 20 THEN 'OUTSTANDING' WHEN IA.FINALIA BETWE
EN 12 AND 16 THEN 'AVERAGE' ELSE 'WEAK' END) AS
 CAT FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUB
JECT SUB WHERE S.USN = IA.USN AND SS.SSID = IA.
SSID AND SUB.SUBCODE = IA.SUBCODE AND SUB.SEM =
 8
```



USN	SNAME	ADDRESS	PHONE	GENDER	SUBCODE	CAT
1BI15CS101	CHETHAN	BENGALURU	534234	M	10CS81	WEAK
1BI15CS101	CHETHAN	BENGALURU	534234	M	10CS82	WEAK
1BI15CS101	CHETHAN	BENGALURU	534234	M	10CS83	WEAK
1BI15CS101	CHETHAN	BENGALURU	534234	M	10CS84	WEAK
1BI15CS101	CHETHAN	BENGALURU	534234	M	10CS85	WEAK