DBMS LAB RECORD

NAME –SHATAKSHI

AGARWAL

USN – 1BM19CS149

SEMESTER – 4

SECTION – C

C

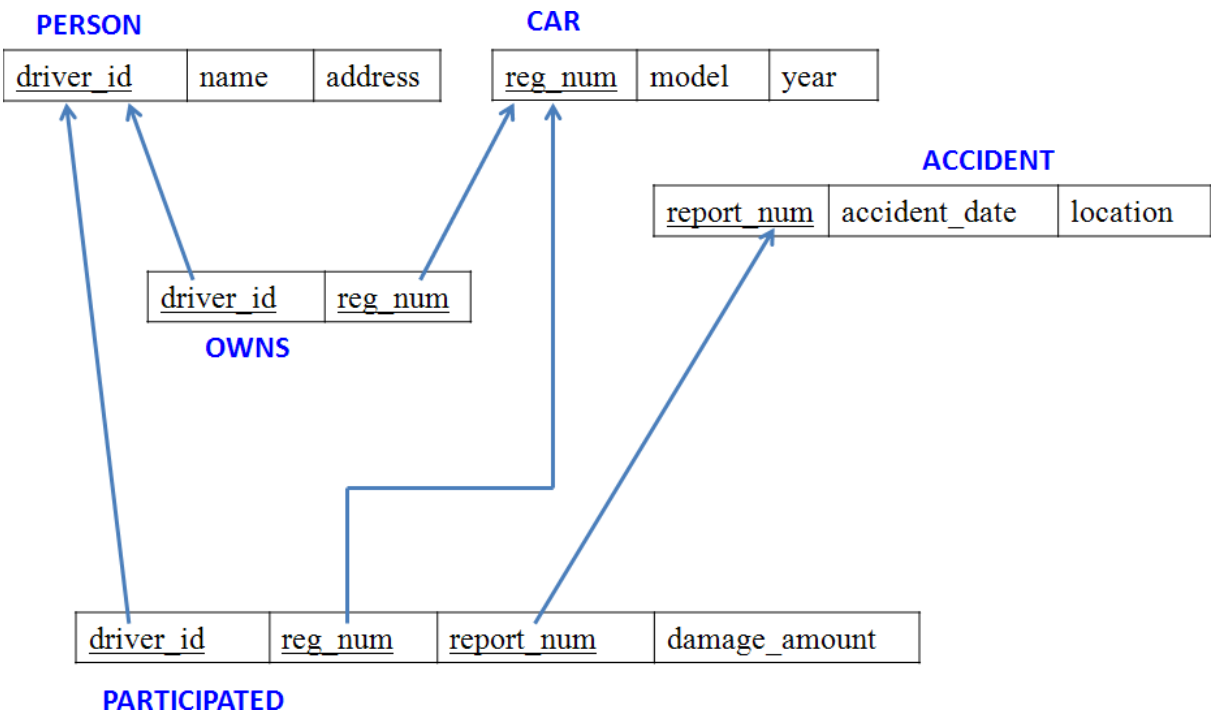
PROGRAM 1: INSURANCE DATABASE

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

PERSON (driver_id: String, name: String, address: String)
CAR (reg_num: String, model: String, year: int)
ACCIDENT (report_num: int, accident_date: date, location: String)
OWNS (driver_id: String, reg_num: String)
PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)

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

Schema diagram



Tables

PERSON

driver_id	name	address
A01	Richard	Srinivas nagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	N R Colony
A05	Jhon	Hanumanth nagar

CAR

reg_num	model	year
KA052250	Indica	1990
KA031181	Lancer	1957
KA095477	Toyota	1998
KA053408	Honda	2008
KA041702	Audi	2005

OWNS

driver_id	reg_num
A01	KA052250
A02	KA053408
A03	KA031181
A04	KA095477
A05	KA041702

ACCIDENT

report_num	accident_date	location
11	01-JAN-03	Mysore Road
12	02-FEB-04	South end Circle
13	21-JAN-03	Bull temple Road
14	17-FEB-08	Mysore Road
15	04-MAR-05	Kanakpura Road

PARTICIPATED

driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A02	KA053408	12	50000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000

```
create database insurance;
use insurance;

create table person(
        driver_id varchar(10),
    name varchar(20),
        address varchar(30),
        primary key(driver_id)
);

desc person;

create table car(
        reg_num varchar(10),
        model varchar(10),
        year int,
        primary key(reg_num)
);

desc car;

create table accident(
        report_num int,
        accident_date date,
        location varchar(20),
        primary key(report_num)
);
desc accident;
create table owns(
        driver_id varchar(10),
        reg_num varchar(10),
```

```
        primary key(driver_id,reg_num),
        foreign key(driver_id) references person(driver_id),
        foreign key(reg_num) references car(reg_num)
);

desc owns;

create table participated(
        driver_id varchar(10),
        reg_num varchar(10),
        report_num int,
        damage_amount int,
        primary key(driver_id,reg_num,report_num),
        foreign key(driver_id) references person(driver_id),
        foreign key(reg_num) references car(reg_num),
        foreign key(report_num) references accident(report_num)
);

desc participated;

insert into person values('A01','Raghu','Electronic City');
insert into person values('A02','Rishab','Orange County');
insert into person values('A03','Rufus','NR Colony');
insert into person values('A04','Jamal','Lawrence Park');
insert into person values('A05','Kevin','Rosedale');

commit;

select * from person;

insert into car values('KA031111','Accord',2005);
insert into car values('KA041122','MX-5',2019);
insert into car values('KA051133','Indica',2010);
insert into car values('KA061144','Prius',2015);
insert into car values('KA071155','Camry',2020);
insert into car values('KA01010', 'Accord', 2002);
commit;

select * from car;

insert into accident values(111,'2020-01-01','NR Road');
insert into accident values(122,'2020-02-02','Dalhousie Road');
insert into accident values(133,'2020-03-03','Henry Road');
insert into accident values(144,'2020-04-04','Beehive Road');
insert into accident values(155,'2020-05-05','Orange Street');
insert into accident values(200, '2008-12-01', 'Pinto Road');
commit;
```

```
select * from accident;

insert into owns values ('A01','KA031111');
insert into owns values ('A02','KA041122');
insert into owns values ('A03','KA051133');
insert into owns values ('A04','KA061144');
insert into owns values ('A05','KA071155');
insert into owns values('A02', 'KA01010');

commit;

select * from owns;

insert into participated values ('A01','KA031111',111, 10000);
insert into participated values ('A02','KA041122',122, 20000);
insert into participated values ('A03','KA051133',133, 30000);
insert into participated values ('A04','KA061144',144, 40000);
insert into participated values ('A05','KA071155',155, 50000);

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

select * from participated;

-- Query 3a
update participated
set damage_amount = 2500
where reg_num='KA031111';

select * from participated;
-- Query 3b
insert into accident values(101,'2020-12-01','Xavier Road');
insert into participated values('A01','KA031111',101, 1001);
commit;
select  *  from  accident;
select * from participated;

-- Query 4
select count(*) from accident where year(accident_date)=2008;
-- Query 5
select count(*) from participated where reg_num in ( select reg_num
from car where model="Accord");
```

Field	Type	Null	Key	Default	Extra
report_num	int	NO	PRI	NULL	
accident_date	date	YES		NULL	
location	varchar(20)	YES		NULL	

Result 3

report_num	accident_da...	location
111	2020-01-01	NR Road
122	2020-02-02	Dalhousie Road
133	2020-03-03	Henry Road
144	2020-04-04	Beehive Road
155	2020-05-05	Orange Street
200	2008-12-01	Pinto Road
NULL	NULL	NULL

accident 12

Field	Type	Null	Key	Default	Extra
reg_num	varchar(10)	NO	PRI	NULL	
model	varchar(10)	YES		NULL	
year	int	YES		NULL	

Result 2

reg_num	model	year
KA01010	Accord	2002
KA031111	Accord	2005
KA041122	MX-5	2019
KA051133	Indica	2010
KA061144	Prius	2015
KA071155	Camry	2020
NULL	NULL	NULL

car 11

Field	Type	Null	Key	Default	Extra
driver_id	varchar(10)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	

Result 4

driver_id	reg_num
A02	KA01010
A01	KA031111
A02	KA041122
A03	KA051133
A04	KA061144
A05	KA071155
NULL	NULL

owns 13

Field	Type	Null	Key	Default	Extra
driver_id	varchar(10)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

Result 5

driver_id	reg_num	report_num	damage_amount
A01	KA031111	111	10000
A02	KA01010	200	500
A02	KA041122	122	20000
A03	KA051133	133	30000
A04	KA061144	144	40000
A05	KA071155	155	50000
NULL	NULL	NULL	NULL

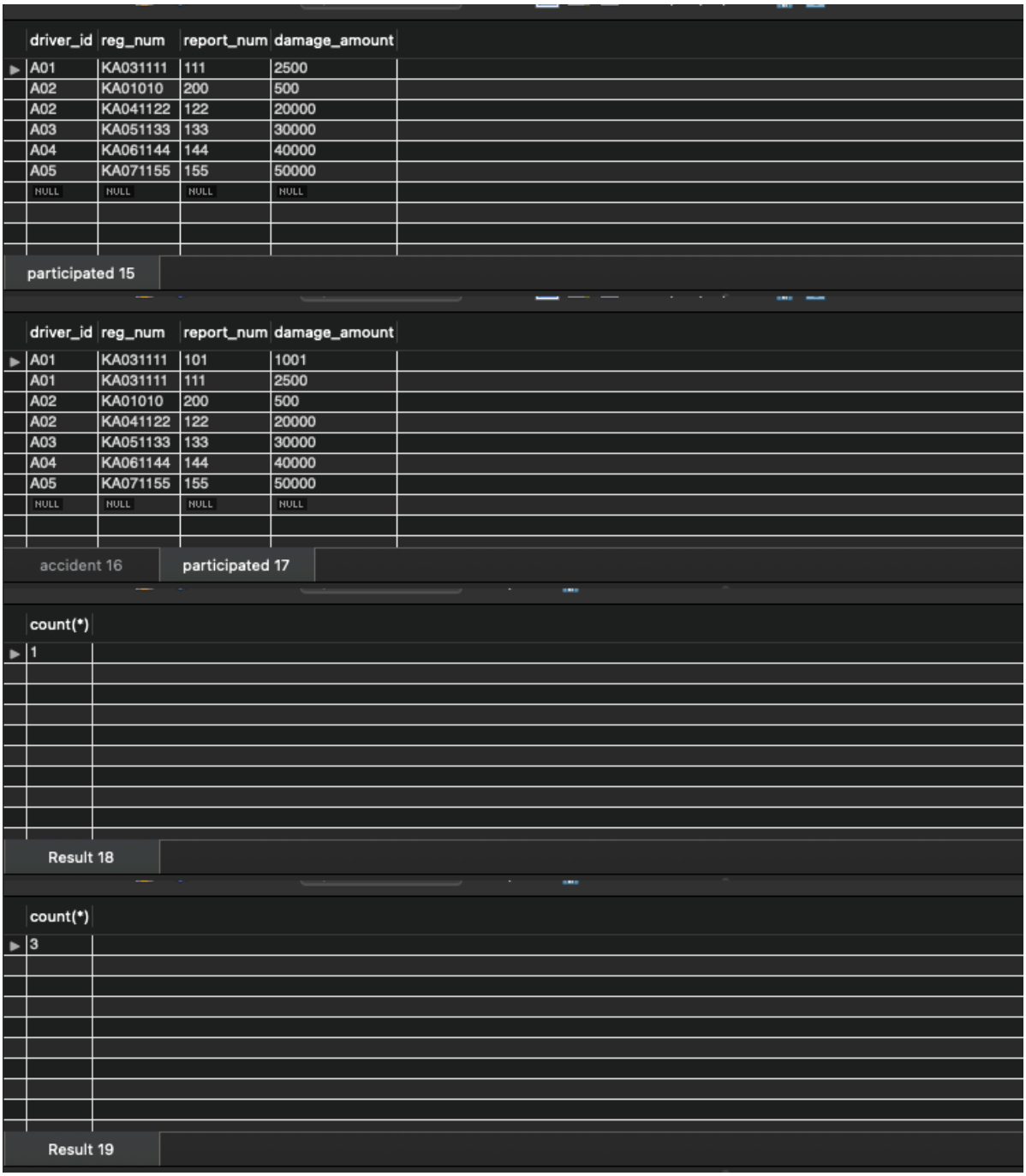
participated 14

Field	Type	Null	Key	Default	Extra
driver_id	varchar(10)	NO	PRI	NULL	
name	varchar(20)	YES		NULL	
address	varchar(30)	YES		NULL	

Result 1

driver_id	name	address
A01	Raghu	Electronic City
A02	Rishab	Orange County
A03	Rufus	NR Colony
A04	Jamal	Lawrence Park
A05	Kevin	Rosedale
NULL	NULL	NULL

person 10



driver_id	reg_num	report_num	damage_amount
A01	KA031111	111	2500
A02	KA01010	200	500
A02	KA041122	122	20000
A03	KA051133	133	30000
A04	KA061144	144	40000
A05	KA071155	155	50000
NULL	NULL	NULL	NULL

participated 15

driver_id	reg_num	report_num	damage_amount
A01	KA031111	101	1001
A01	KA031111	111	2500
A02	KA01010	200	500
A02	KA041122	122	20000
A03	KA051133	133	30000
A04	KA061144	144	40000
A05	KA071155	155	50000
NULL	NULL	NULL	NULL

accident 16 participated 17

count(*)
1

Result 18

count(*)
3

Result 19

PROGRAM 2: BANKING ENTERPRISE DATABASE

Consider the following database for a banking enterprise.

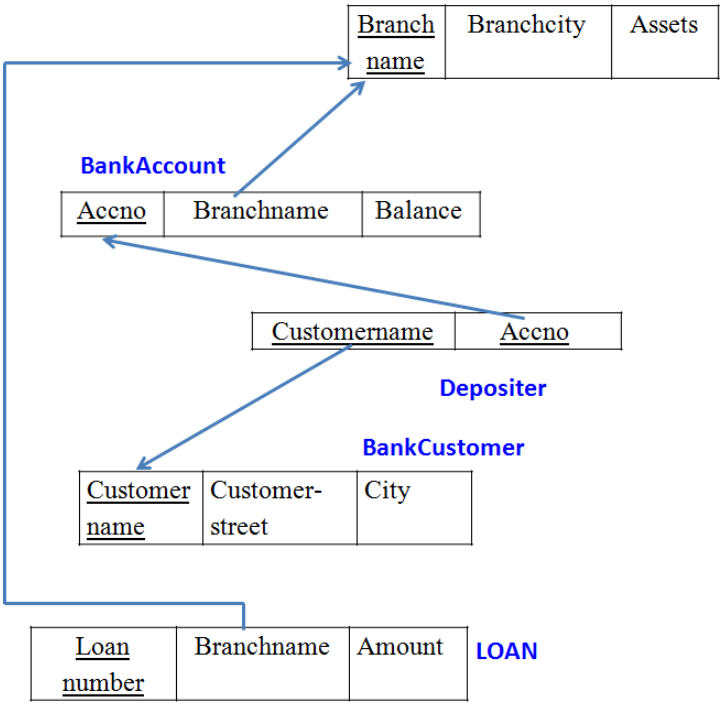
Branch (branch-name: String, branch-city: String, assets: real)
BankAccount(accno: int, branch-name: String, balance: real)

BankCustomer (customer-name: String, customer-street: String, customer-city: String)
Depositer(customer-name: String, accno: int)
Loan (loan-number: int, branch-name: String, amount: real)

i. Create the above tables by properly specifying the primary keys and the foreign keys.
ii. Enter at least five tuples for each relation.
iii. Find all the customers who have at least two accounts at the *Main* branch (ex. SBI_ResidencyRoad).
iv. Find all the customers who have an account at *all* the branches located in a specific city (Ex. Delhi).
v. Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).

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

Schema Diagram



Branch

BRANCHNAME	BRANCHCITY	ASSESTS
SBI_Chamrajpet	Bangalore	50000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
SBI_ParlimentRoad	Delhi	10000
SBI_Jantarmantar	Delhi	20000

BankAccount

ACCNO	BRANCHNAME	BALANCE
1	SBI_Chamrajpet	2000
2	SBI_ResidencyRoad	5000
3	SBI_ShivajiRoad	6000
4	SBI_ParlimentRoad	9000
5	SBI_Jantarmantar	8000
6	SBI_ShivajiRoad	4000
8	SBI_ResidencyRoad	4000
9	SBI_ParlimentRoad	3000
10	SBI_ResidencyRoad	5000
11	SBI_Jantarmantar	2000

BankCustomer

CUSTOMERNAME	CUSTOMERSTREET	CUSTOMERCITY
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi

Depositer

CUSTOMERNAME	ACCNO
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11

Loan

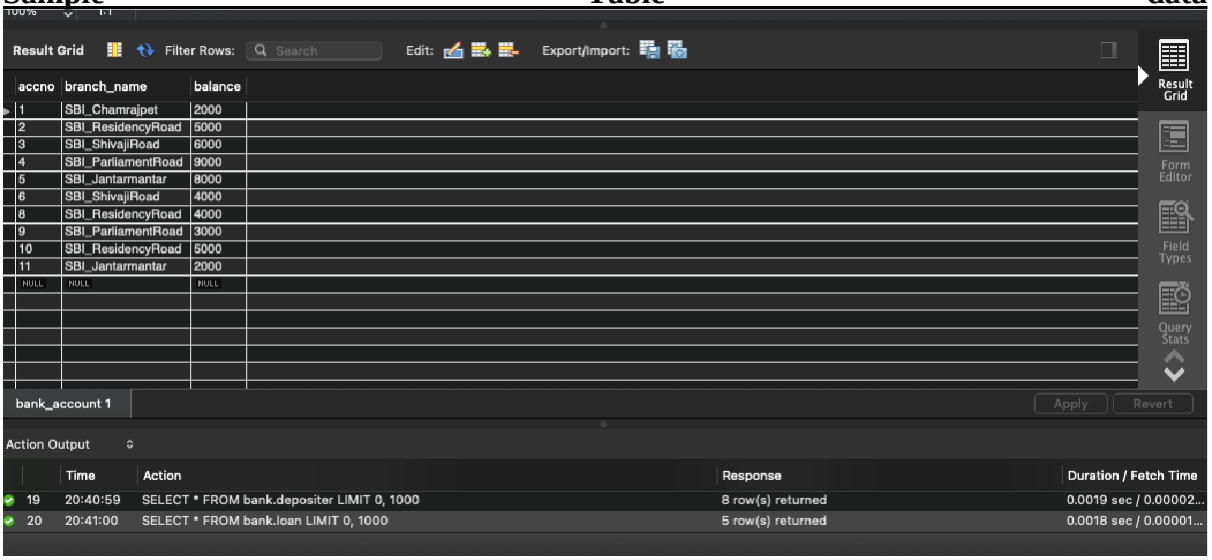
LOANNUMBER	BRANCHNAME	AMOUNT
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_ParlimentRoad	4000
5	SBI_Jantarmantar	5000

```
create database bank;
use bank;

create table branch (
        branch_name varchar(25),
   branch_city varchar(15),
   assets int,
   primary key (branch_name)
);

create table bank_account (
        accno int,
   branch_name varchar(25),
   balance int,
   primary key (accno),
   foreign key (branch_name) references branch(branch_name)
);

create table bank_customer (
        customer_name varchar(10),
   customer_street varchar(25),
   customer_city varchar(15),
   primary key (customer_name)
```

```
);

create table depositer (
        customer_name varchar(10),
        accno int,
   primary key(customer_name, accno),
   foreign key (customer_name) references bank_customer(customer_name),
   foreign key (accno) references bank_account(accno)
);

create table loan (
        loan_number int,
   branch_name varchar(25),
   amount int,
   primary key (loan_number),
   foreign key (branch_name) references branch(branch_name)
);

insert into branch values('SBI_Chamrajpet', 'Bangalore', 50000);
insert into branch values('SBI_ResidencyRoad', 'Bangalore', 10000);
insert into branch values('SBI_ShivajiRoad', 'Bombay', 20000);
insert into branch values('SBI_ParliamentRoad', 'Delhi', 10000);
insert into branch values('SBI_Jantarmantar', 'Delhi', 20000);
commit;

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

insert into bank_customer values ('Avinash', 'Bull_Temple_Road', 'Bangalore');
insert into bank_customer values ('Dinesh', 'Bannergatta_Road', 'Bangalore');
insert into bank_customer values ('Mohan', 'National_College_Road', 'Bangalore');
insert into bank_customer values ('Nikhil', 'Akbar_Road', 'Delhi');
insert into bank_customer values ('Ravi', 'Prithviraj_Road', 'Delhi');
commit;

insert into depositer values('Avinash', 1);
insert into depositer values('Dinesh', 2);
insert into depositer values('Nikhil', 4);
insert into depositer values('Ravi', 5);
insert into depositer values('Avinash', 8);
insert into depositer values('Nikhil', 9);
```

```
insert into depositer values('Dinesh', 10);
insert into depositer values('Nikhil', 11);
commit;

insert into loan values(1, 'SBI_Chamrajpet', 1000);
insert into loan values(2, 'SBI_ResidencyRoad', 2000);
insert into loan values(3, 'SBI_ShivajiRoad', 3000);
insert into loan values(4, 'SBI_ParliamentRoad', 4000);
insert into loan values(5, 'SBI_Jantarmantar', 5000);
commit;


select * from branch;
select * from bank_account;
select * from bank_customer;
select * from depositer;
select * from loan;

select distinct c.customer_name from bank_customer c,bank_account b where exists(select d.customer_name,count(d.customer_name) from depositer d,bank_account ba  where ba.accno = d.accno and
c.customer_name = d.customer_name and ba.branch_name = 'SBI_ResidencyRoad' group by d.customer_name having count(d.customer_name)>=2);


select d.customer_name from depositer d,branch b,bank_account a
where b.branch_name=a.branch_name
AND a.accno=d.accno
and branch_city='Delhi'
group by d.customer_name
 HAVING COUNT(distinct b.branch_name)=(
            SELECT COUNT(branch_name)
            FROM branch
            WHERE branch_city='Delhi');



delete from bank_account where branch_name in (select branch_name from branch where branch_city = 'Bombay');
```

Sample Table data



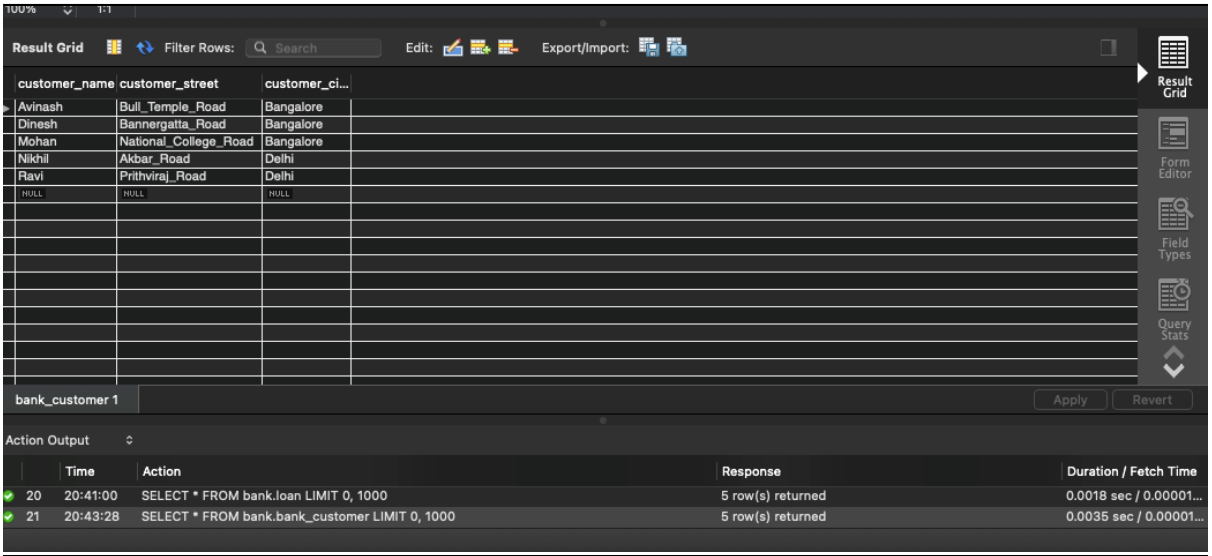
Result Grid | Filter Rows: Search | Edit: | Export/Import:

accno	branch_name	balance
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
NULL	NULL	NULL

bank_account 1 | Apply | Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
19	20:40:59	SELECT * FROM bank.depositer LIMIT 0, 1000	8 row(s) returned	0.0019 sec / 0.00002...
20	20:41:00	SELECT * FROM bank.loan LIMIT 0, 1000	5 row(s) returned	0.0018 sec / 0.00001...



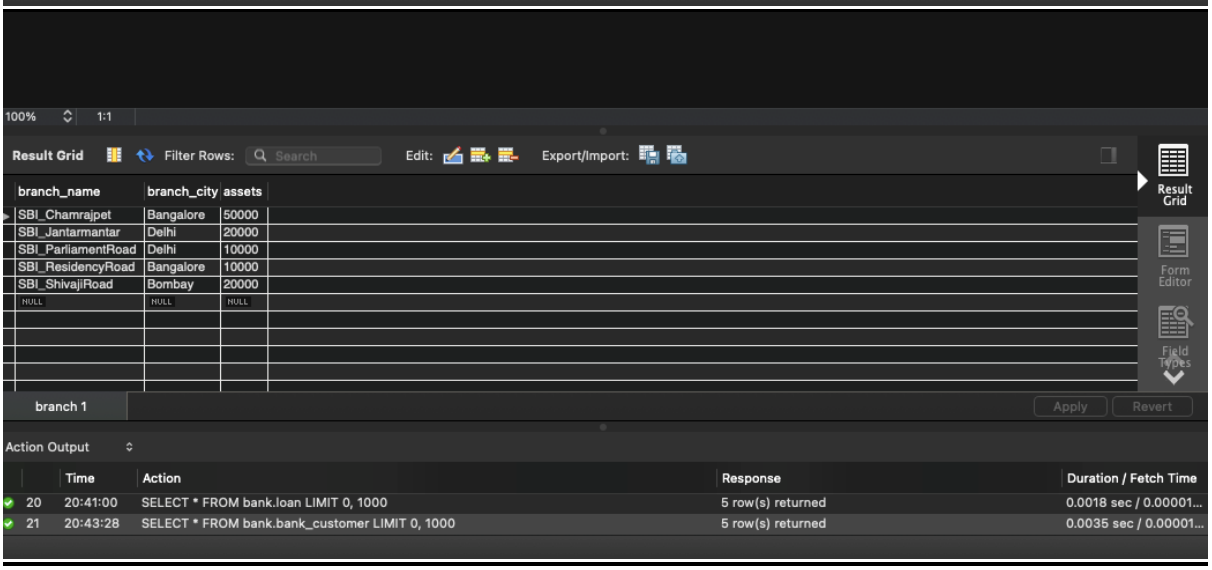
Result Grid | Filter Rows: Search | Edit: | Export/Import:

customer_name	customer_street	customer_ci...
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	National_College_Road	Bangalore
Nikhil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

bank_customer 1 | Apply | Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
20	20:41:00	SELECT * FROM bank.loan LIMIT 0, 1000	5 row(s) returned	0.0018 sec / 0.00001...
21	20:43:28	SELECT * FROM bank.bank_customer LIMIT 0, 1000	5 row(s) returned	0.0035 sec / 0.00001...



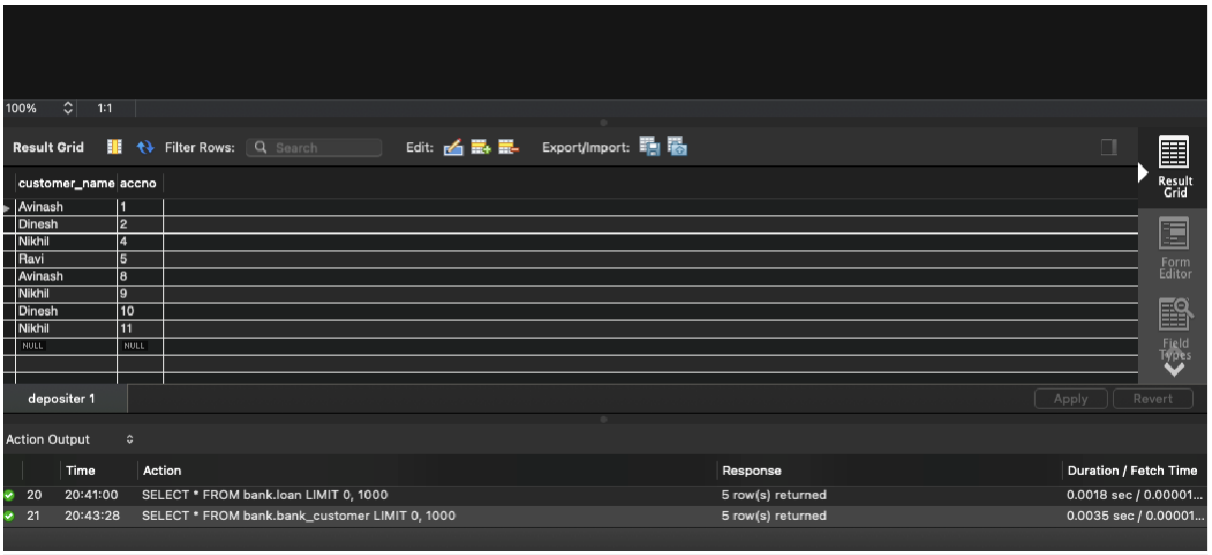
Result Grid | Filter Rows: Search | Edit: | Export/Import:

branch_name	branch_city	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParliamentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

branch 1 | Apply | Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
20	20:41:00	SELECT * FROM bank.loan LIMIT 0, 1000	5 row(s) returned	0.0018 sec / 0.00001...
21	20:43:28	SELECT * FROM bank.bank_customer LIMIT 0, 1000	5 row(s) returned	0.0035 sec / 0.00001...

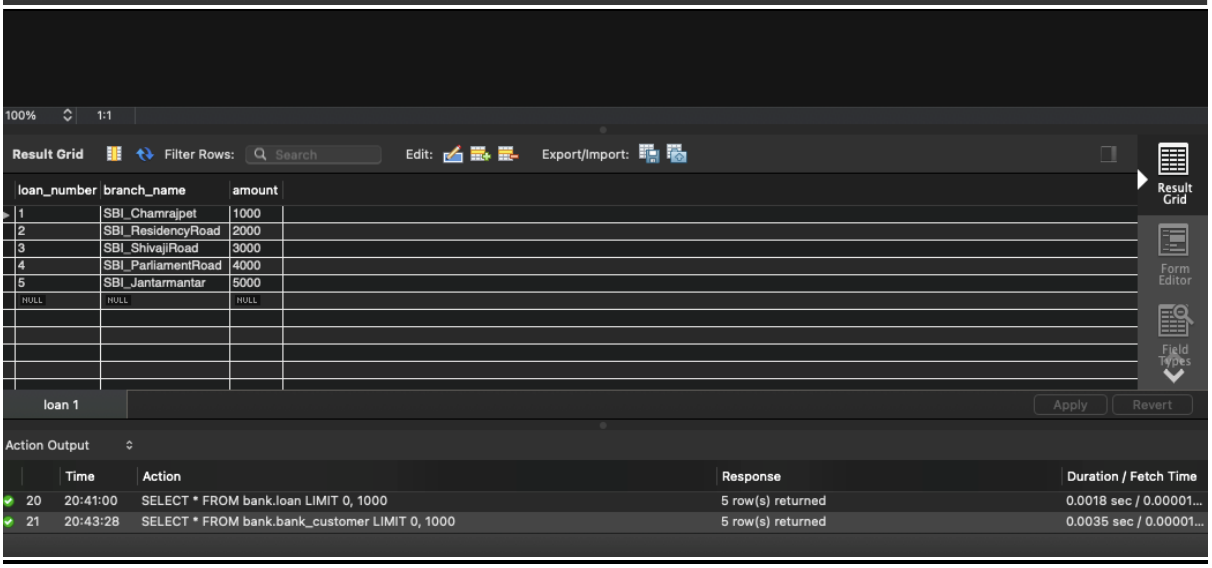


customer_name	accno
Avinash	1
Dinesh	2
Nikhil	4
Ravi	5
Avinash	6
Nikhil	9
Dinesh	10
Nikhil	11
NULL	NULL

depositer 1

Action Output

	Time	Action	Response	Duration / Fetch Time
20	20:41:00	SELECT * FROM bank.loan LIMIT 0, 1000	5 row(s) returned	0.0018 sec / 0.00001...
21	20:43:28	SELECT * FROM bank.bank_customer LIMIT 0, 1000	5 row(s) returned	0.0035 sec / 0.00001...



loan_number	branch_name	amount
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_ParliamentRoad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

loan 1

Action Output

	Time	Action	Response	Duration / Fetch Time
20	20:41:00	SELECT * FROM bank.loan LIMIT 0, 1000	5 row(s) returned	0.0018 sec / 0.00001...
21	20:43:28	SELECT * FROM bank.bank_customer LIMIT 0, 1000	5 row(s) returned	0.0035 sec / 0.00001...

PROGRAM 3: SUPPLIER DATABASE

Consider the following schema:

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

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

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

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

Write the following queries in SQL:

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.

Schema Diagram

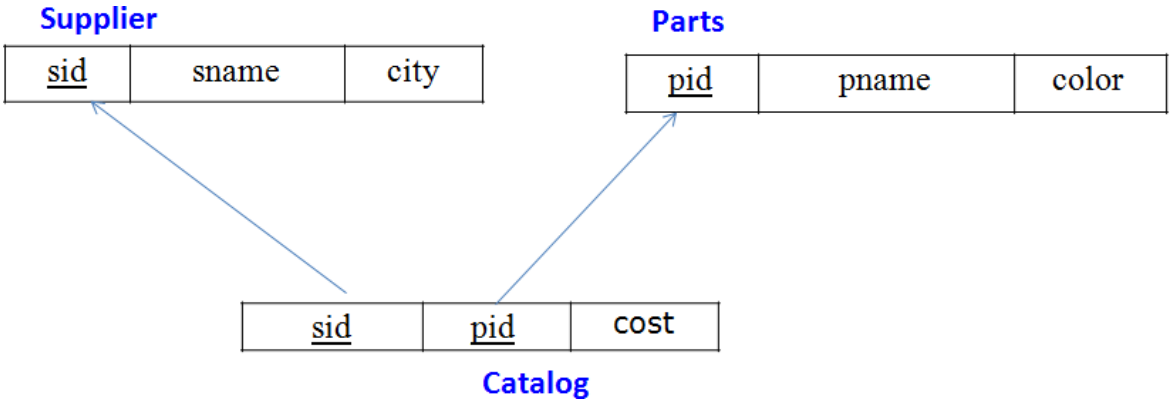


Table Data

SUPPLIERS

SID	SNAME	CITY
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi

PARTS

PID	PNAME	COLOR
20001	Book	Red
20002	Pen	Red
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black

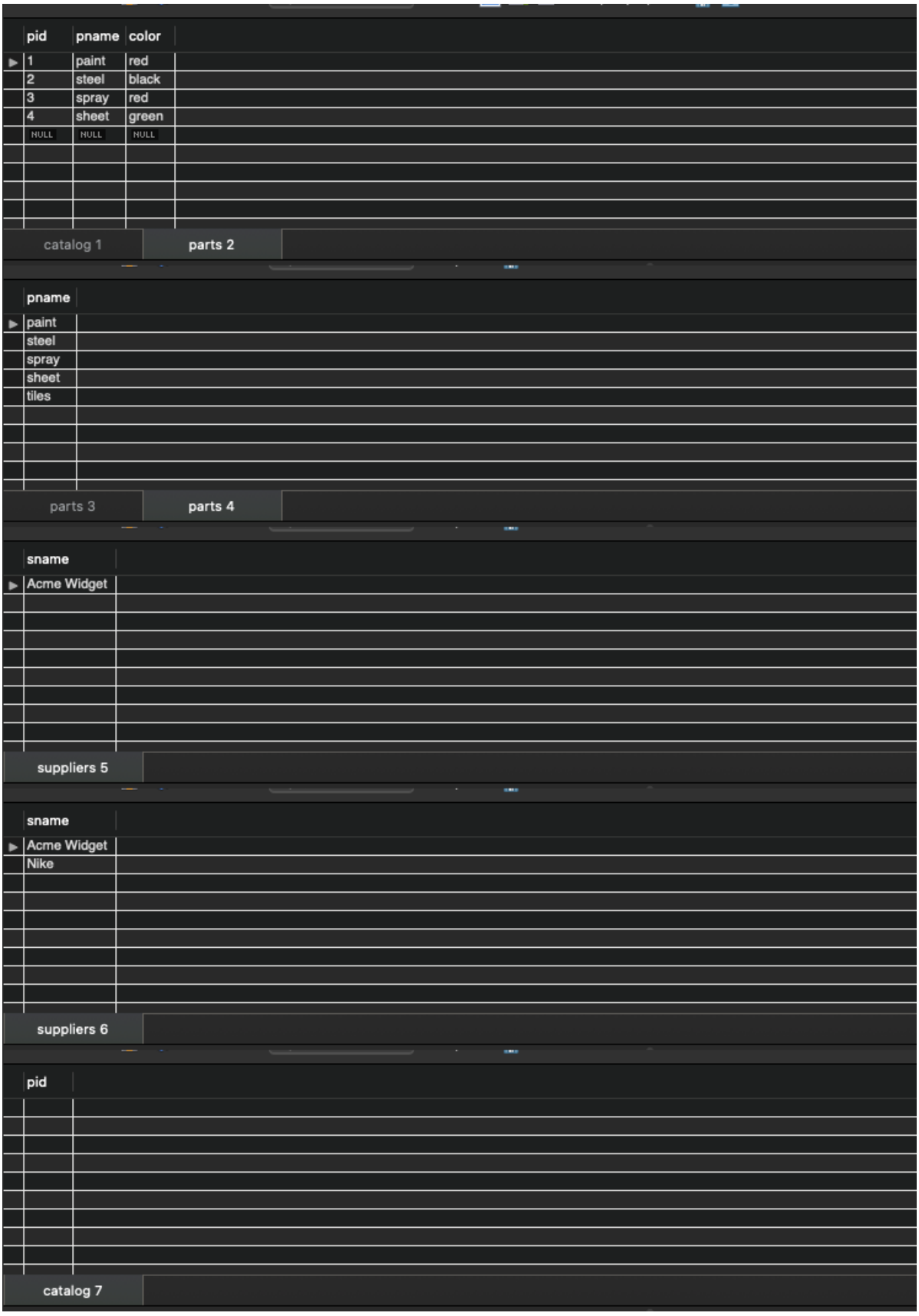
CATALOG

SID	PID	COST
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

```
create database supplier;
use supplier;
create table suppliers(
        sid int primary key,
```

```
   sname varchar(30),
   address varchar(30)
);
create table parts(
        pid int primary key,
   pname varchar(30),
   color varchar(30)
);
create table catalog (
        sid int ,
   pid int ,
   cost real,
   constraint c_sid foreign key(sid) references suppliers(sid) ,
   constraint c_pid foreign key(pid) references parts(pid)
);
insert into suppliers values(1,'Acme Widget','kolkata') ;
insert into suppliers values(2,'Tata','bengaluru') ;
insert into suppliers values(3,'Reebok','delhi') ;
insert into suppliers values(4,'Nike','delhi') ;
insert into suppliers values(5,'Reliance','delhi') ;


insert into parts values(1,'paint','red') ;
insert into parts values(2,'steel','black') ;
insert into parts values(3,'spray','red') ;
insert into parts values(4,'sheet','green');
insert into parts values(5,'tiles','blue');
delete from parts where pid=5;

insert into catalog values(1,1,100);
insert into catalog values(1,2,200);
insert into catalog values(1,3,200);
insert into catalog values(1,4,100);
insert into catalog values(2,1,300);
insert into catalog values(2,2,100);
insert into catalog values(3,2,90);
insert into catalog values(3,3,110);
insert into catalog values(3,4,110);
insert into catalog values(4,1,100);
insert into catalog values(4,3,120);
insert into catalog values(4,4,130);

select * from catalog;
select * from parts;

--      i. Find the pnames of parts for which there is some supplier.
insert into parts values(5,'tiles','blue');
select p.pname from parts p where p.pid in (select pid from catalog c group by c.pid
having count(c.sid)>0);
insert into catalog values(1,5,140);
```

```
select p.pname from parts p where p.pid in (select pid from catalog c group by c.pid
having count(c.sid)>0);
delete from catalog where pid=5;
delete from parts where pid=5;

-- ii. Find the snames of suppliers who supply every part.
select s.sname from suppliers s where s.sid in (select c.sid from catalog c group by c.sid
having count(distinct (c.pid))=(select count(p.pid) from parts p));

-- iii. Find the snames of suppliers who supply every red part.
select s.sname from suppliers s where s.sid in (select ca.sid from catalog ca,parts p
where ca.pid=p.pid and p.color='red' group by ca.sid having count(ca.pid)=(select
count(*) from parts p where p.color='red'));

-- iv. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
select ca.pid from catalog ca where ca.sid=(select s.sid from suppliers s where s.sname
='Acme Widget') having (select count(c.pid) from catalog c where c.pid=ca.pid)=1;

-- v. Find the sids of suppliers who charge more for some part than the average cost of
that part (averaged over
-- all the suppliers who supply that part).
select distinct c.sid,c.pid from catalog c where c.cost > (select avg(ca.cost) from catalog
ca where ca.pid=c.pid);

-- vi. For each part, find the sname of the supplier who charges the most for that part.
select s.sname from suppliers s where s.sid in (select c.sid from catalog c where
c.cost=(select max(cost) from catalog ca where ca.pid=c.pid));

-- vii. select supplier who sell only red parts
select s.sname from suppliers s where s.sid in(select c.sid from catalog c where c.sid not
in (select distinct(ca.sid) from catalog ca,parts p where ca.pid=p.pid and
p.color!='red'));
insert into catalog values(5,1,140);
select s.sname from suppliers s where s.sid in(select c.sid from catalog c where c.sid not
in (select distinct(ca.sid) from catalog ca,parts p where ca.pid=p.pid and
p.color!='red'));
delete from catalog where sid=5;
```



pid	pname	color
1	paint	red
2	steel	black
3	spray	red
4	sheet	green
NULL	NULL	NULL

catalog 1 | parts 2

pname
paint
steel
spray
sheet
tiles

parts 3 | parts 4

sname
Acme Widget

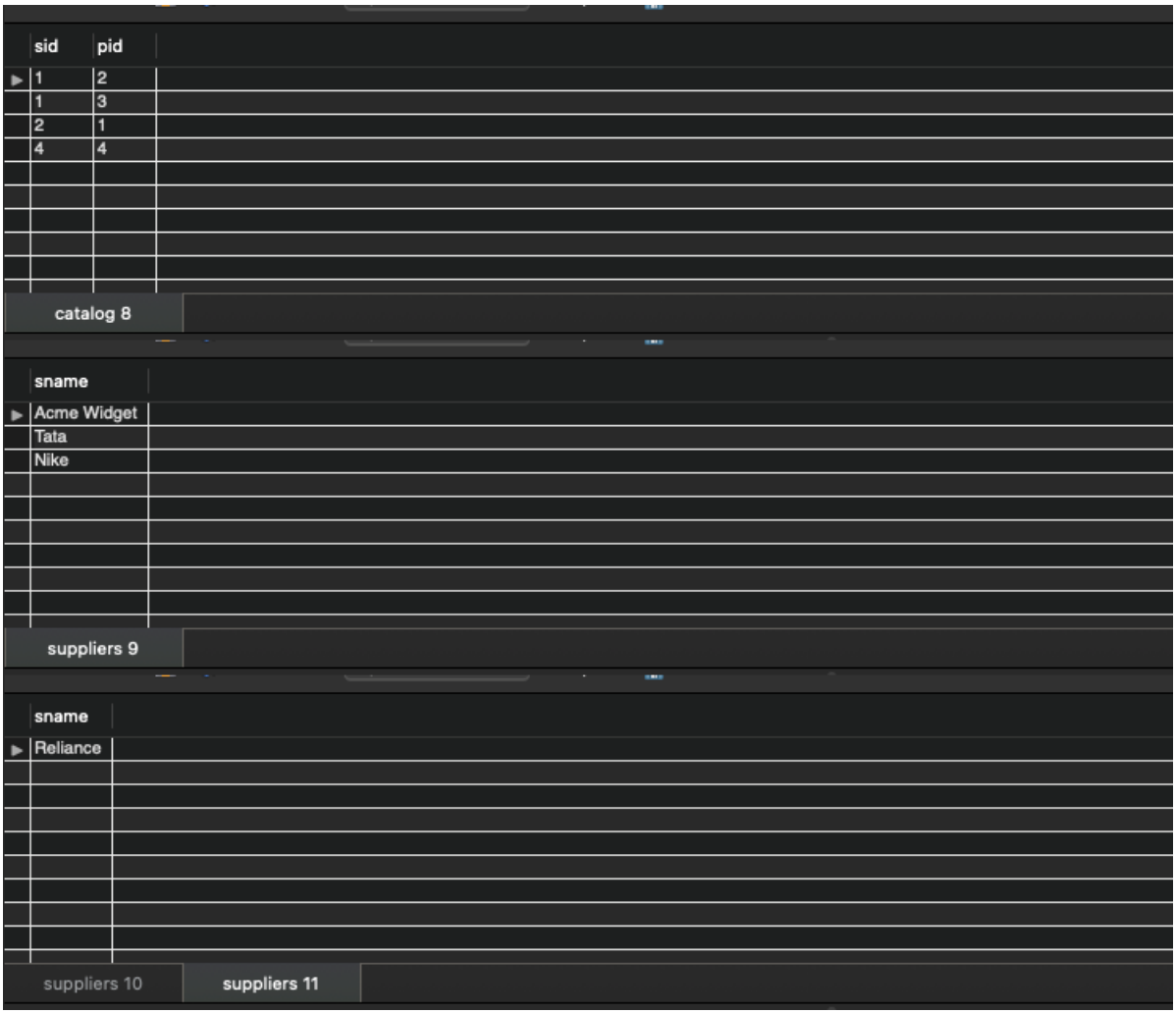
suppliers 5

sname
Acme Widget
Nike

suppliers 6

pid

catalog 7



sid	pid
1	2
1	3
2	1
4	4

catalog 8

sname
Acme Widget
Tata
Nike

suppliers 9

sname
Reliance

suppliers 10 suppliers 11

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

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. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

```
create database studentdb;

use studentdb;



create table student(
        snum int,
    sname varchar(10),
    major varchar(2),
    lvl varchar(2),
    age int,
    primary key(snum)
);
```

```
desc student;
create table class(
        cname varchar(8),
    meetsat time,
    room varchar(5),
    fid int,
    primary key (cname)
);
desc class;
create table enrolled(
        snum int,
    cname varchar(10),
    primary key (snum, cname),
    foreign key (snum) references student(snum),
    foreign key (cname) references class(cname)
);
desc enrolled;
create table faculty(
        fid int,
    fname varchar(10),
    deptid int,
    primary key (fid)
);
```

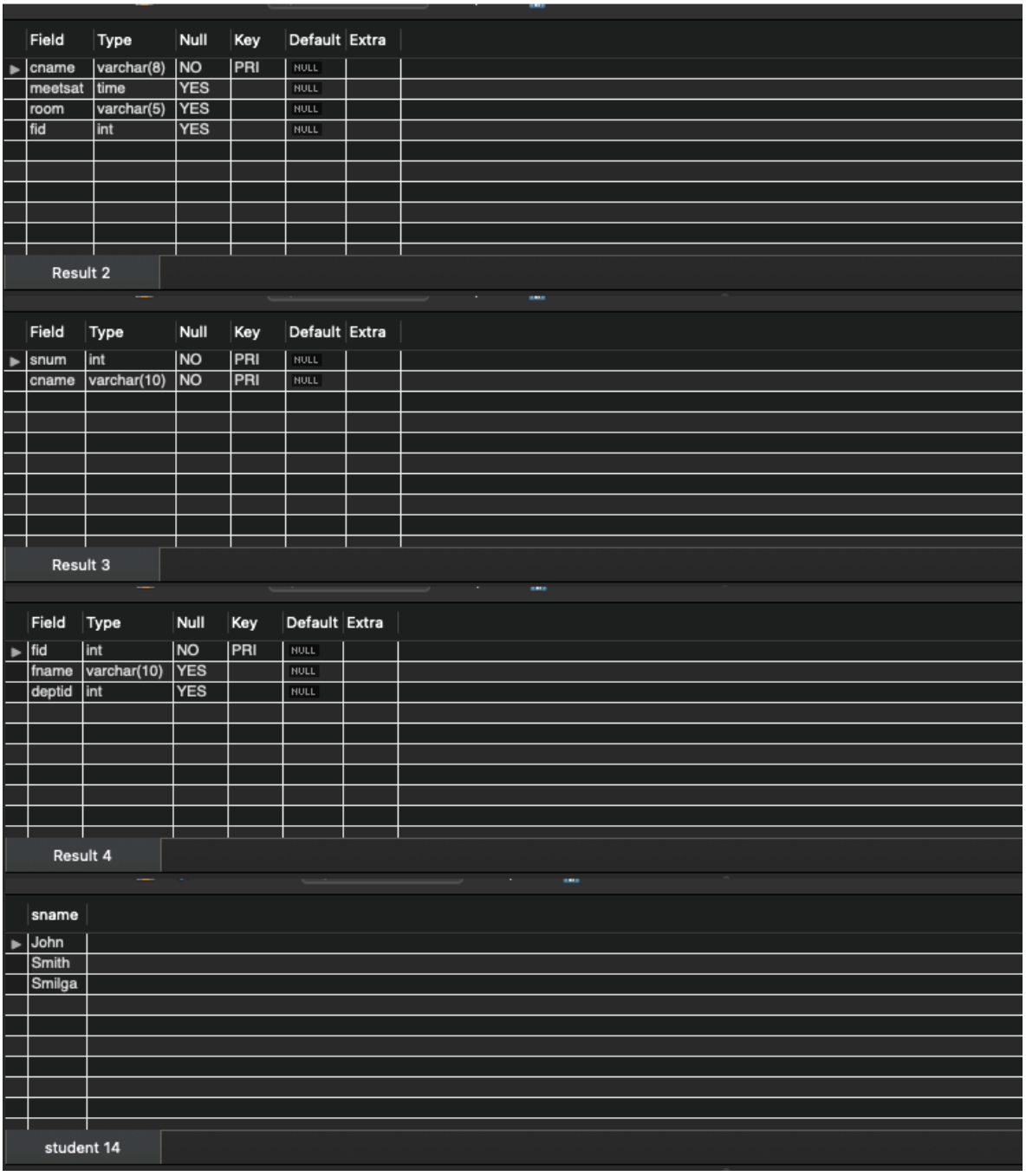
```
desc faculty;

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

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

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

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

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

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

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

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

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

commit;


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

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

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

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

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

commit;


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

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

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

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

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

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

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

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

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

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

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

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


commit;

select * from class;

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

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

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

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

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

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

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

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


commit;


select * from student;
```

```
select * from faculty;

select * from class;

select * from enrolled;


-- Query 1

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

        (select snum from enrolled where cname in

        (select cname from class where fid in

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

));




-- Query 2

select cname from class where cname in(

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

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


-- Query 3

select sname from student where snum in(

select snum from enrolled where cname in(

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

```
-- Query 4

SELECT f.fname,f.fid

                FROM faculty f

        WHERE f.fid in ( SELECT fid FROM class

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

-- Query 5

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

-- Query 6


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


-- Query 7


SELECT S.age, S.lvl

FROM student S

GROUP BY S.age, S.lvl

HAVING S.lvl IN(SELECT S1.lvl

        FROM student S1

        WHERE S1.age=S.age

        GROUP BY S1.age, S1.lvl

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

```
        FROM student S2
        WHERE S1.age=S2.age
        GROUP BY S2.lvl, S2.age))
ORDER BY S.age;
```



Field	Type	Null	Key	Default	Extra
cname	varchar(8)	NO	PRI	NULL	
meetsat	time	YES		NULL	
room	varchar(5)	YES		NULL	
fid	int	YES		NULL	

Result 2

Field	Type	Null	Key	Default	Extra
snum	int	NO	PRI	NULL	
cname	varchar(10)	NO	PRI	NULL	

Result 3

Field	Type	Null	Key	Default	Extra
fid	int	NO	PRI	NULL	
fname	varchar(10)	YES		NULL	
deptid	int	YES		NULL	

Result 4

sname
John
Smith
Smilga

student 14

cname
Class1
Class10
NULL

class 15

sname
Jacob

student 16

fname	fid
Harish	11
NULL	NULL

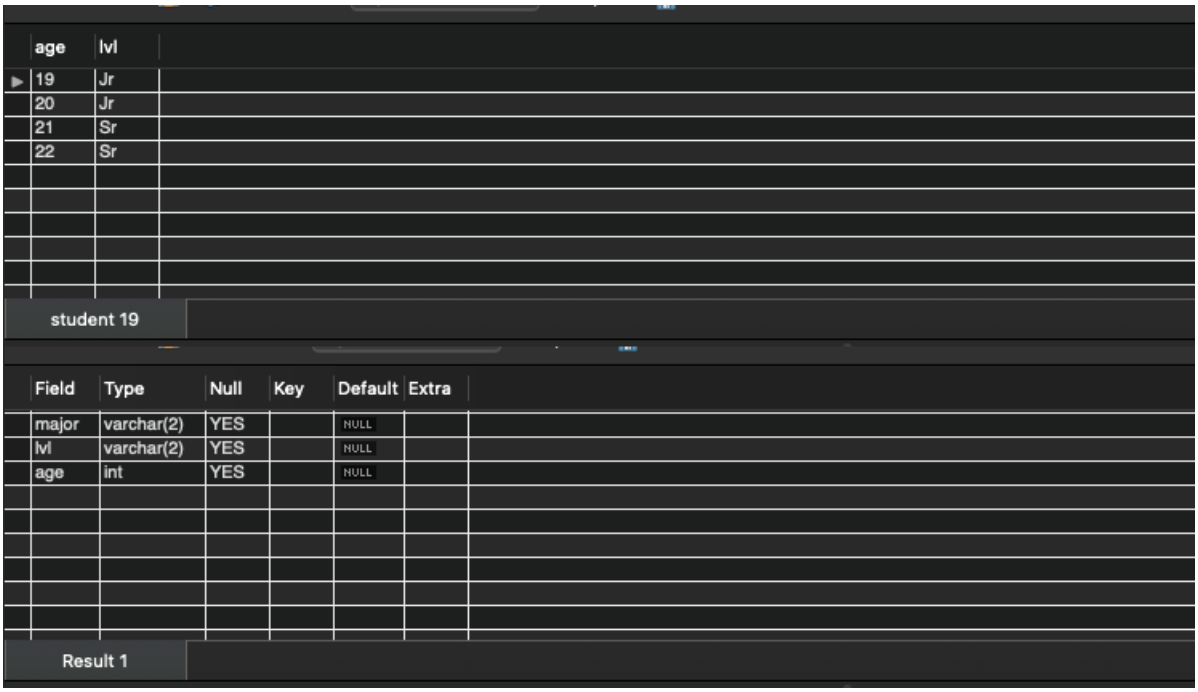
faculty 17

fid
11
12

class 18

fid
11
12

class 18



age	lvl
19	Jr
20	Jr
21	Sr
22	Sr

student 19

Field	Type	Null	Key	Default	Extra
major	varchar(2)	YES		NULL	
lvl	varchar(2)	YES		NULL	
age	int	YES		NULL	

Result 1

PROGRAM 5: AIRLINE FLIGHT DATABASE

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

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

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

CERTIFIED(eid: integer, aid: integer)

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

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

Write each of the following queries in SQL.

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 cruisingrange 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 flightdb;
use flightdb;


create table flights(
        flno int,
    fromplace varchar(15),
    toplace varchar(15),
    distance int,
    departs datetime,
    arrives  datetime,
    price int,
    primary key (flno)
);
desc flights;
create table aircraft(
        aid int,
    aname varchar(15),
    cruisingrange int,
```

```
    primary key (aid)
);
desc aircraft;
create table employees (
        eid int,
    ename varchar(15),
    salary int,
    primary key (eid)
);
desc employees;
create table certified (
        eid int,
    aid int,
    foreign key (eid) references employees(eid),
    foreign key (aid) references aircraft(aid)
);
desc certified;
insert into flights values(101, 'Bangalore', 'Delhi', 2500, '2005-05-13 07:15:31', '2005-05-13 18:15:31', 5000);
insert into flights values(102, 'Bangalore', 'Lucknow', 3000, '2013-05-05 07:15:31', '2013-05-05 11:15:31', 6000);
insert into flights values(103, 'Lucknow', 'Delhi', 500, '2013-05-05 12:15:31', '2013-05-05 17:15:31', 3000);
```

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

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

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

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

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


commit;


select * from flights;


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

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

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

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

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

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

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

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

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

```
commit;

select * from aircraft;

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

select * from employees;

insert into certified values(701, 101);
insert into certified values(701, 102);
insert into certified values(701, 106);
insert into certified values(701, 105);

insert into certified values(702, 104);
insert into certified values(703, 104);
insert into certified values(704, 104);
```

```
insert into certified values(702, 107);
insert into certified values(703, 107);
insert into certified values(704, 107);

insert into certified values(702, 101);
insert into certified values(702, 108);
insert into certified values(701, 109);
commit;
select * from certified;

-- Query 1
select distinct a.aname from aircraft a where a.aid in (
        select c.aid from certified c, employees e where
   c.eid = e.eid and not exists(
                select * from employees e1 where e1.eid=e.eid and e1.salary<80000
   )
);

-- Query 2
select max(a.cruisingrange), c.eid from certified c, aircraft a where c.aid = a.aid group by
c.eid having count(c.eid)>3;
```

```
-- Query 3

select ename from employees where salary <(

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


-- Query 4


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

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


-- Query 5

select ename from employees where eid in(

select eid from certified where aid in(

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


-- Query 6


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



-- Query 7

SELECT F.flno, F.departs

FROM flights F
```

```
WHERE F.flno IN ( ( SELECT F0.flno
 FROM flights F0
 WHERE F0.fromplace = 'Bangalore' AND F0.toplace = 'Kolkata'
 AND extract(hour from F0.arrives) < 18 )
 UNION
( SELECT F0.flno
 FROM flights F0, flights F1
 WHERE F0.fromplace = 'Bangalore' AND F0.toplace <> 'Kolkata'
 AND F0.toplace = F1.fromplace AND F1.toplace = 'Kolkata'
 AND F1.departs > F0.arrives
 AND extract(hour from F1.arrives) < 18)
 UNION
( SELECT F0.flno
 FROM flights F0, flights F1, flights F2
 WHERE F0.fromplace = 'Bangalore'
 AND F0.toplace = F1.fromplace
 AND F1.toplace = F2.fromplace
 AND F2.toplace = 'Kolkata'
 AND F0.toplace <> 'Kolkata'
 AND F1.toplace <> 'Kolkata'
 AND F1.departs > F0.arrives
 AND F2.departs > F1.arrives
 AND extract(hour from F2.arrives) < 18));
```

Field	Type	Null	Key	Default	Extra
aid	int	NO	PRI	NULL	
aname	varchar(15)	YES		NULL	
cruisingrange	int	YES		NULL	

Result 2

aid	aname	cruisingrange
101	747	3000
102	Boeing	900
1€O	647	
104	Dreamliner	10000
105	Booing	3500
106	707	1500
107	Dream	120000
108	707	760
109	747	1000
NULL	NULL	NULL

aircraft 6

Field	Type	Null	Key	Default	Extra
eid	int	YES	MUL	NULL	
aid	int	YES	MUL	NULL	

Result 4

eid	aid
701	101
701	102
701	106
764	104
702	107
703	107
704	107

certified 9

Field	Type	Null	Key	Default	Extra
eid	int	NO	PRI	NULL	
ename	varchar(15)	YES		NULL	
salary	int	YES		NULL	

Result 3

eid	ename	salary
701	A	50000
702	B	100000
7IO	C	150000
704	D	90000
705	E	40000
706	F	60000
707	G	90000
NULL	NULL	NULL

employees 7

flno	fromplace	toplace	distance	departs	arrives	price
101	Bangalore	Delhi	2500	2005-05-13 07:15:31	2005-05-13 18:15:31	5000
102	Bangalore	Lucknow	3000	2013-05-05 07:15:31	2013-05-05 11:15:31	6000
103	Lucknow	Delhi	500	2013-05-05 12:15:31	2013-05-05 17:15:31	2000
104	Bangalore	Frankfurt	8500	2013-05-05 07:15:31	2013-05-05 23:15:31	75000
105	Kolkata	Delhi	3400	2013-05-05 07:15:31	2013-05-05 09:15:31	7000
106	Bangalore	Kolkata	1000	2013-05-05 01:15:30	2013-05-05 09:20:30	10000
107	Bangalore	Frankfurt	8500	2013-05-05 07:15:31	2013-05-05 23:15:31	80000
108	Lucknow	Kolkata	1000	2013-05-05 11:30:30	2013-05-05 15:20:30	10000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

flights 5

Field	Type	Null	Key	Default	Extra
toplace	varchar(15)	YES		NULL	
distance	int	YES		NULL	
departs	datetime	YES		NULL	
arrives	datetime	YES		NULL	
price	int	YES		NULL	

Result 1

aname
747
Dreamliner

707

aircraft 10

max(a.cruisingran...	eid
3500	701
120000	702

ename
A
E

employees 12

avg(e.salary)	aid
75000.0000	101
113333.3333	104
MD00.0000	US
50000.0000	106
113333.9333	107

Result 13

ename
A

employees 14

aname
747
Dreamliner
Booing
Dream

aircraft 15

flno	departs
102	2013-05-05 07:15:31
106	2013-05-05 01:15:30

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.

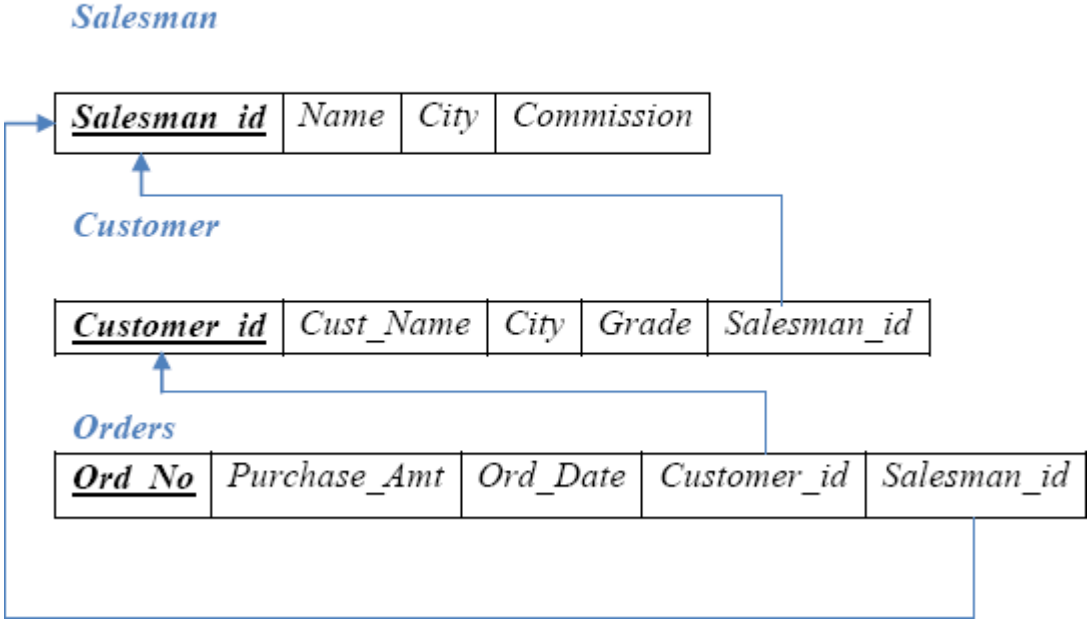
2. Find the name and numbers of all salesmen who had more than one customer.

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.

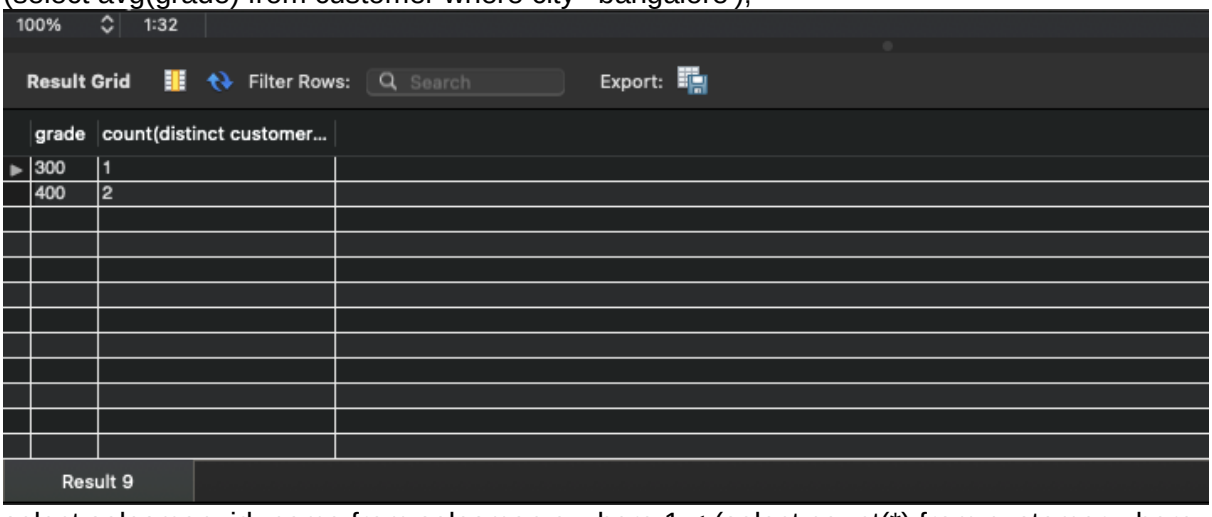
5. Demonstrate the DELETE operation by removing salesman with id 1000. All his orders must also be deleted.

Schema Diagram



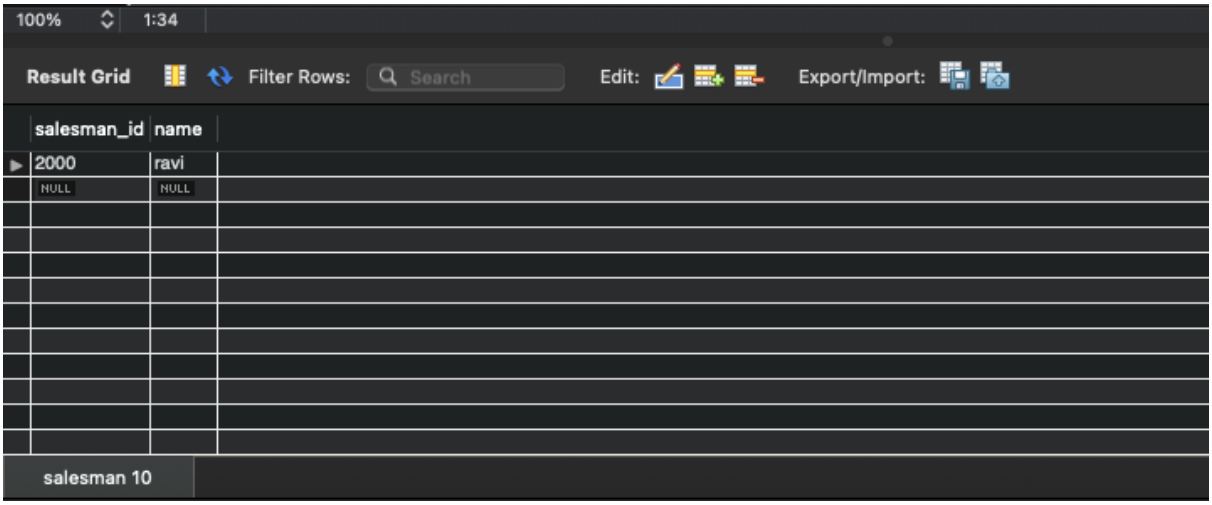

```
create database order_processsing;
use order_processsing;
```

```
create table salesman (salesman_id int, name varchar (20), city varchar (20), commission
varchar (20), primary key (salesman_id));
create table customer (customer_id int, cust_name varchar (20), city varchar (20), grade int,
salesman_id int, primary key (customer_id), foreign key(salesman_id) references
salesman(salesman_id)
on delete set null);
create table orders (ord_no int, purchase_amt real, ord_date date, customer_id int,
salesman_id int, primary key (ord_no), foreign key (customer_id) references
customer(customer_id)
 on delete cascade, foreign key(salesman_id) references salesman(salesman_id) on delete
cascade);

insert into salesman values (1000, 'john','bangalore','25 %');
insert into salesman values (2000, 'ravi','bangalore','20 %');
insert into salesman values (3000, 'kumar','mysore','15 %');
insert into salesman values (4000, 'smith','delhi','30 %');
insert into salesman values (5000, 'harsha','hydrabad','15 %');

insert into customer values (10, 'preethi','bangalore', 100, 1000);
insert into customer values (11, 'vivek','mangalore', 300, 1000);
insert into customer values (12, 'bhaskar','chennai', 400, 2000);
insert into customer values (13, 'chethan','bangalore', 200, 2000);
insert into customer values (14, 'mamatha','bangalore', 400, 3000);

insert into orders values (50, 5000, '04-05-17', 10, 1000);
insert into orders values (51, 450, '20-01-17', 10, 2000);
insert into orders values (52, 1000, '24-02-17', 13, 2000);
insert into orders values (53, 3500, '13-04-17', 14, 3000);
insert into orders values (54, 550, '09-03-17', 12, 2000);

select * from salesman;
select * from customer;
select * from orders;

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

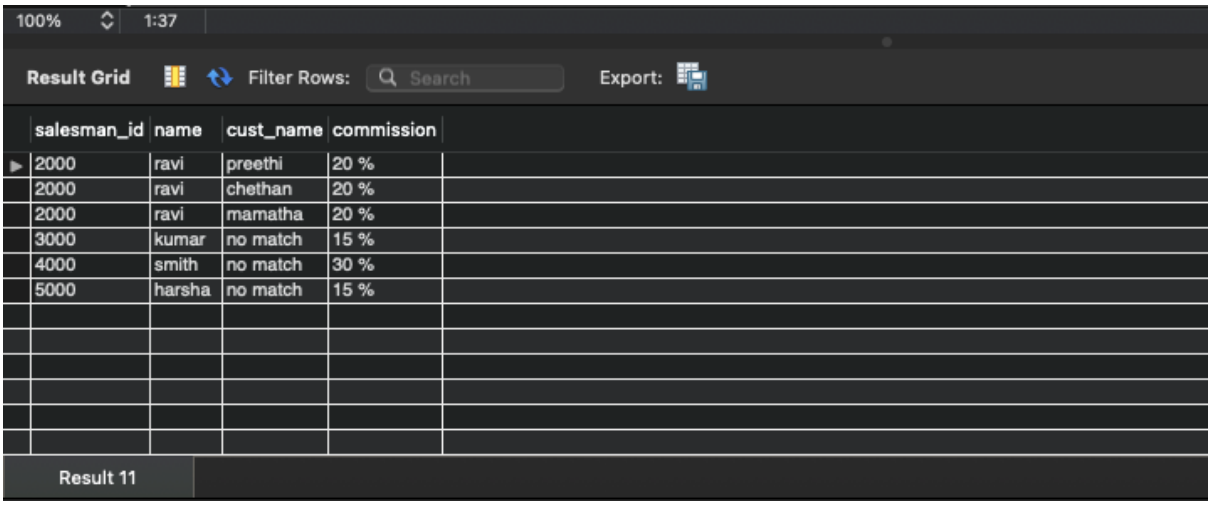



```
select salesman_id, name from salesman a where 1 < (select count(*) from customer where
salesman_id=a.salesman_id);
```

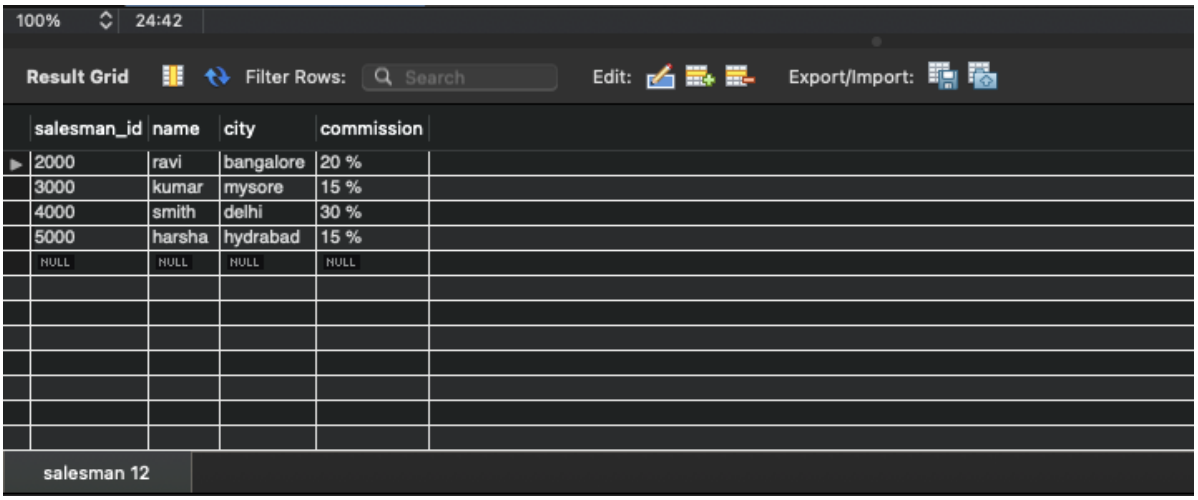
select salesman.salesman_id, name, cust_name, commission from salesman, customer where salesman.city = customer.city union
select salesman_id, name, 'no match', commission from salesman where not city = any (select city from customer);



create view salesman_view as select b.ord_date, a.salesman_id, a.name from salesman a, orders b where a.salesman_id = b.salesman_id and b.purchase_amt=(select max(purchase_amt) from orders c where c.ord_date = b.ord_date);
select * from salesman_view;

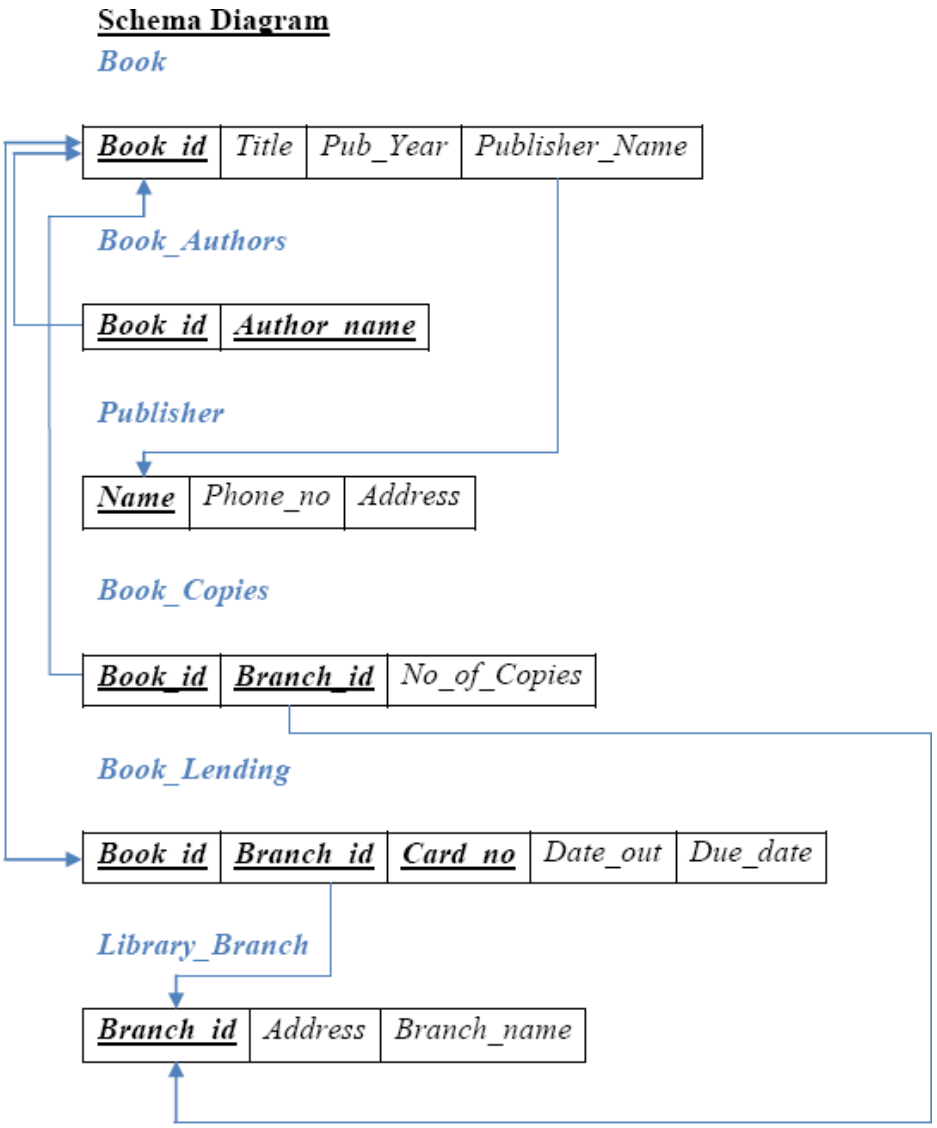


delete from salesman where salesman_id=1000;
select * from salesman;


100%
24:42
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
salesman_id
name
city
commission
2000
ravi
bangalore
20 %
3000
kumar
mysore
15 %
4000
smith
delhi
30 %
5000
harsha
hydrabad
15 %
NULL
NULL
NULL
NULL
salesman 12

Program 7 : Book Database

BOOK (Book_id, Title, Publisher_Name, Pub_Year)
BOOK_AUTHORS (Book_id, Author_Name)
PUBLISHER (Name, Address, Phone)
BOOK_COPIES (Book_id, Branch_id, No-of_Copies)
BOOK_LENDING (Book_id, Branch_id, Card_No, Date_Out, Due_Date)
LIBRARY_BRANCH (Branch_id, Branch_Name, Address)



Write SQL queries to

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

100% 39:93

Result Grid Filter Rows: Search Export:

author_id	author_city	author_name	author_city
1001	London	JK Rowling	London
1001	London	JK Rowling	London

Result 23

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

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

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

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

book_id	book_title	author_id	publisher_id	category_id	year	price
4001	HP and Goblet Of Fire	1001	101	3001	2002	600
4002	HP and Order Of Phoenix	1001	102	3001	2005	650
4003	Two States	1002	104	3001	2009	65
4004	3 Mistakes of my life	1002	104	3001	2007	55
4005	Da Vinci Code	1004	103	3001	2004	450
4006	Angels and Demons	1004	103	3001	2003	350
4007	Artificial Intelligence	1003	102	3002	1970	500
NULL	NULL	NULL	NULL	NULL	NULL	NULL

catalog 24

Program 8:

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

STUDENT (regno: string, name: string, major: string, bdate:date)
COURSE (course #:int, cname:string, dept:string)
ENROLL (regno:string, course#:int, sem:int, marks:int)
BOOK _ ADOPTION (course# :int, sem:int, book-ISBN:int)
TEXT (book-ISBN:int, book-title:string, publisher:string, author:string)

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

100% 72:124

Result Grid Filter Rows: Search Export:

USN	SNAME	ADDRESS	PHONE	GENDER	SEM	SEC
1RN15CS011	AJAY	TUMKUR	1232654578	M	4	A
1RN15CS029	CHITRA	DAVANGERE	1232654578	F	4	A

Result 14

ii. Enter at least five tuples for each relation.

100% 1:134

Result Grid Filter Rows: Search Export:

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

Result 15

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

100% 14:47 1 error found

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
SSID	varchar(5)	NO	PRI	NULL	
SEM	int	YES		NULL	
SEC	char(1)	YES		NULL	

Result 1 | Result 2 | Result 3 | Result 4 | Result 5

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.

100% 14:47 1 error found

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
USN	varchar(10)	NO	PRI	NULL	
SNAME	varchar(25)	YES		NULL	
ADDRESS	varchar(25)	YES		NULL	
PHONE	int	YES		NULL	
GENDER	char(1)	YES		NULL	

Result 1 | Result 2 | Result 3 | Result 4 | Result 5

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

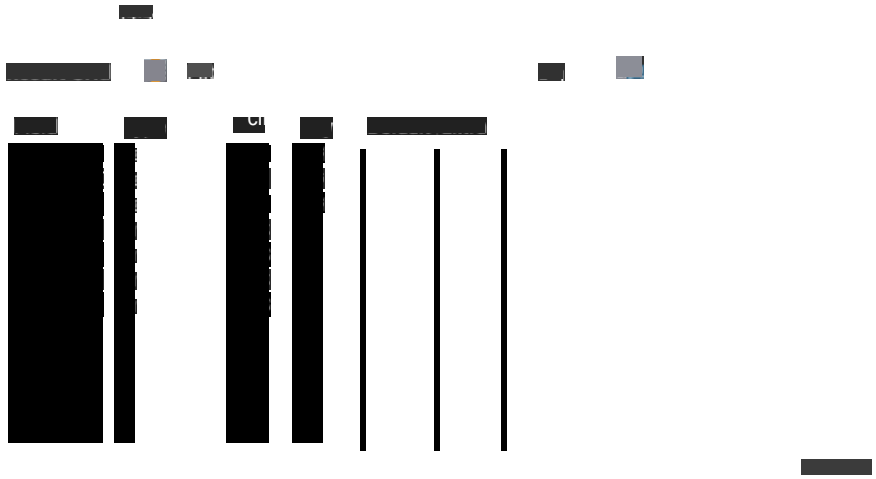
100% 14:47 1 error found

Result Grid Filter Rows: Search Export:

Field	Type	Null	Key	Default	Extra
SUBCODE	varchar(8)	NO	PRI	NULL	
TITLE	varchar(20)	YES		NULL	
SEM	int	YES		NULL	
CREDITS	int	YES		NULL	

Result 1 | Result 2 | Result 3 | Result 4 | Result 5

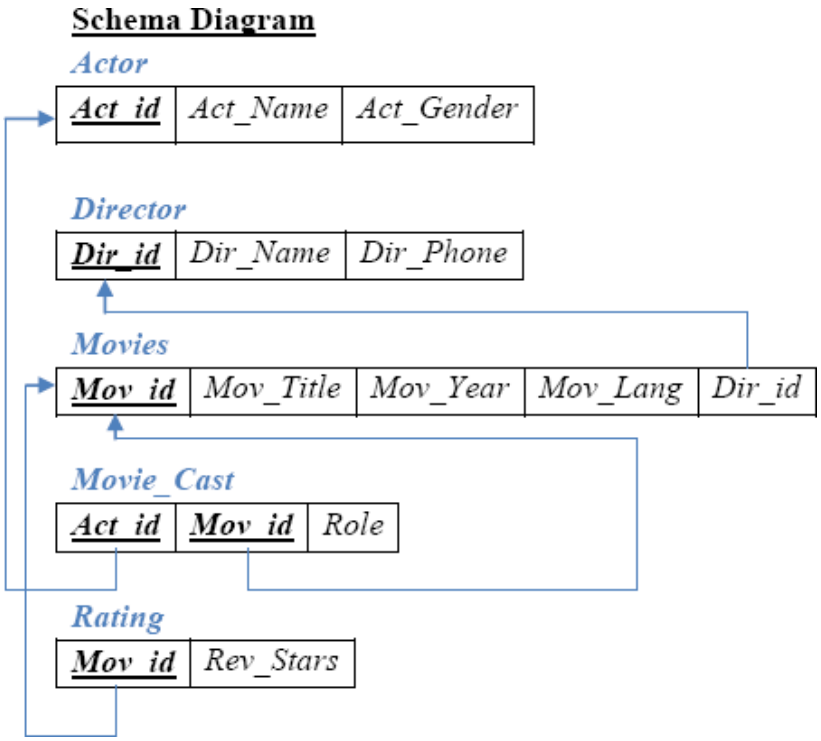
vii. Create suitable front end for querying and displaying the results.



Program 9: Movie database

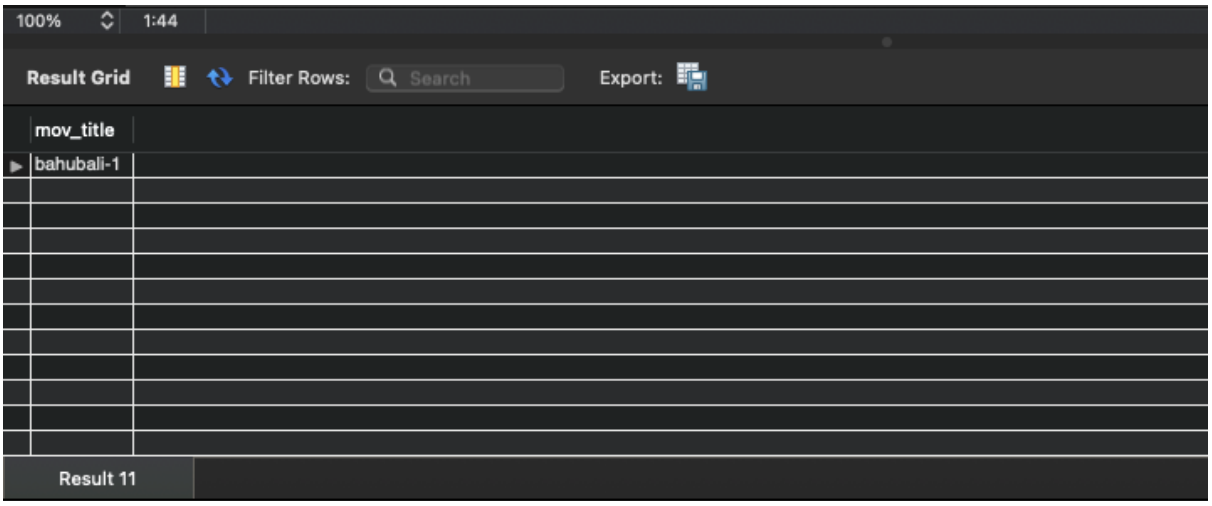
Consider the schema for Movie Database:

ACTOR (*Act_id, Act_Name, Act_Gender*)
DIRECTOR (*Dir_id, Dir_Name, Dir_Phone*)
MOVIES (*Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id*)
MOVIE_CAST (*Act_id, Mov_id, Role*)
RATING (*Mov_id, Rev_Stars*)
Write SQL queries to



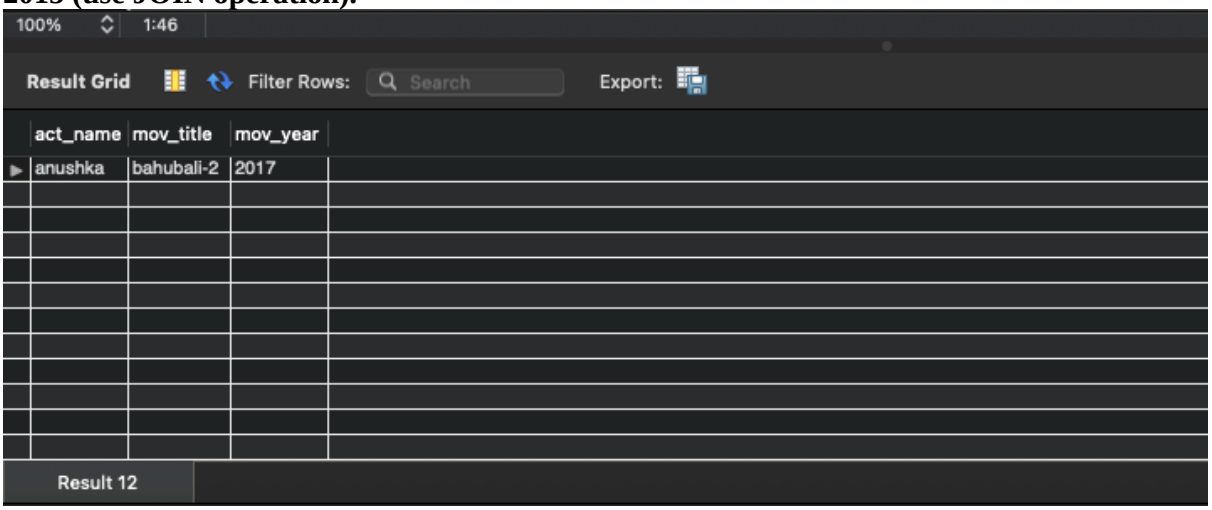
1. List the titles of all movies directed by 'Hitchcock'.

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

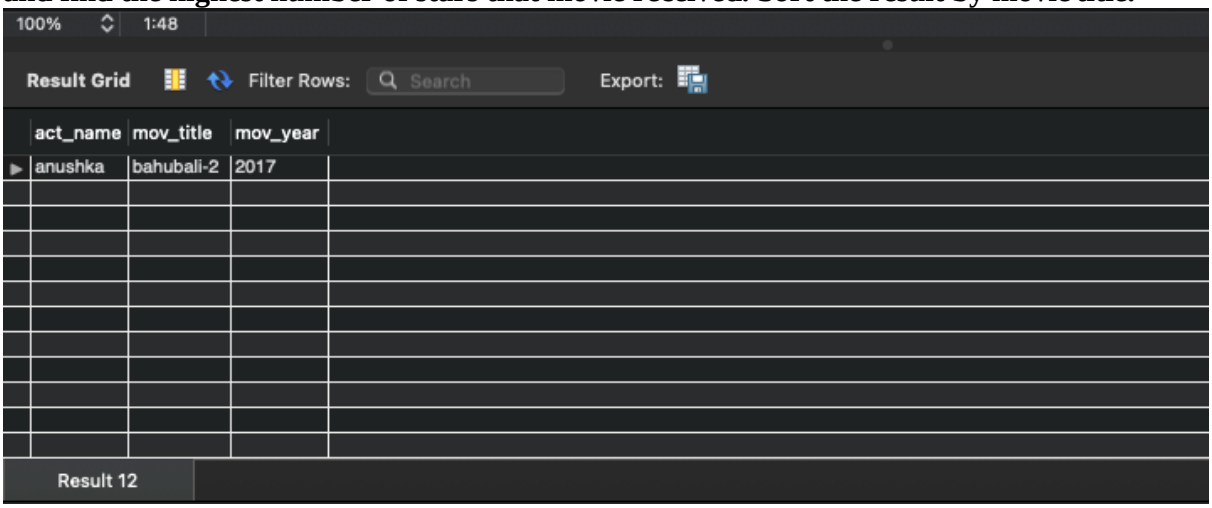


3. List all actors who acted in a movie before 2000 and also in a movie after

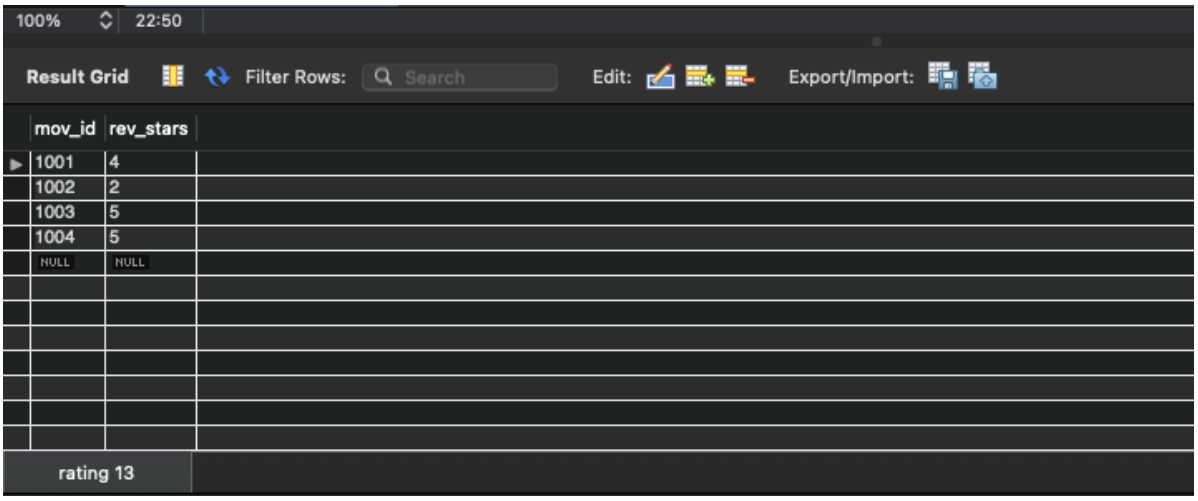
2015 (use JOIN operation).



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



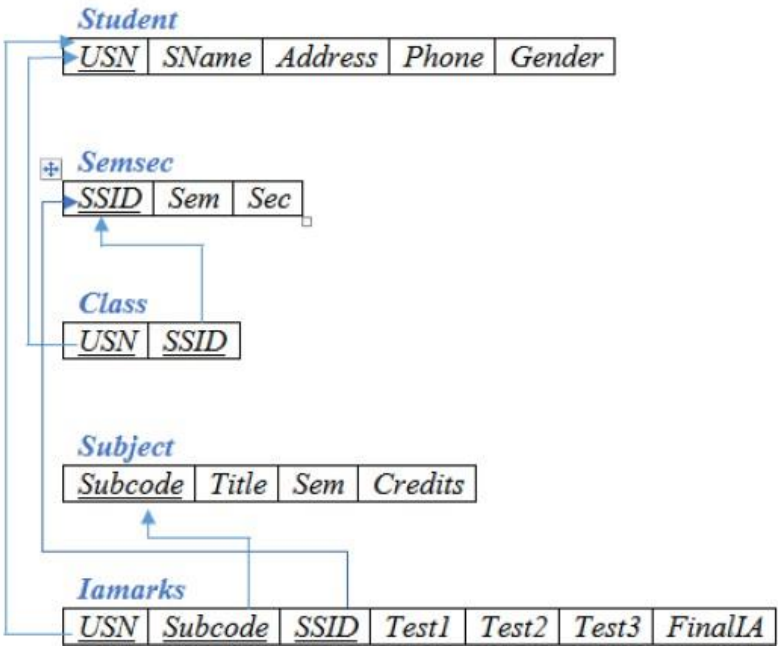
5. Update rating of all movies directed by 'Steven Spielberg' to 5.


100%
22:50
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
mov_id
rev_stars
1001
4
1002
2
1003
5
1004
5
NULL
NULL
rating 13

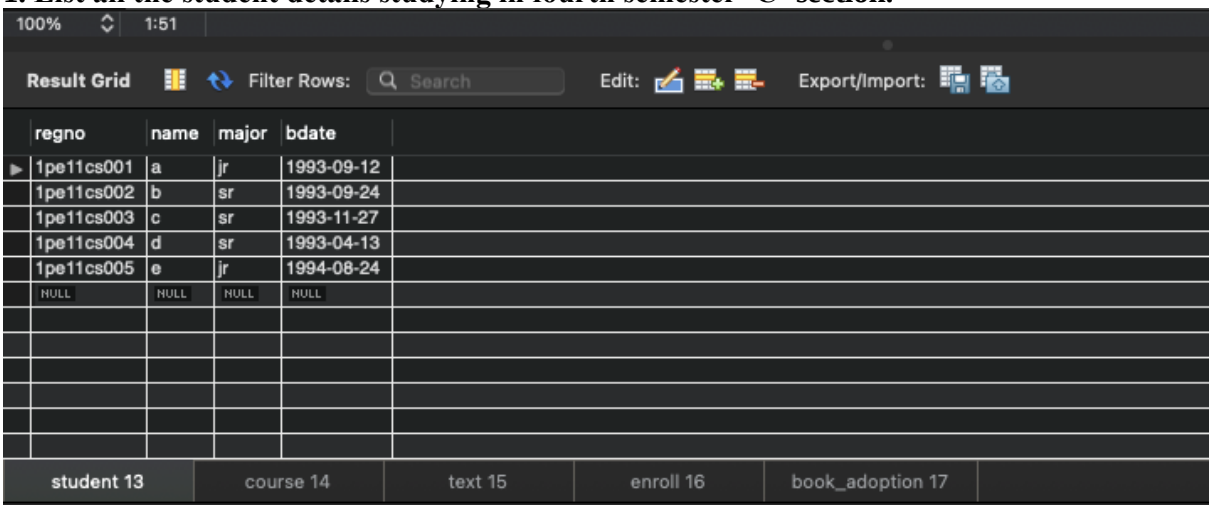
Program 10

Consider the schema for College Database:
STUDENT (*USN, SName, Address, Phone, Gender*)
SEMSEC (*SSID, Sem, Sec*)
CLASS (*USN, SSID*)
SUBJECT (*Subcode, Title, Sem, Credits*)
IAMARKS (*USN, Subcode, SSID, Test1, Test2, Test3, FinalIA*)
Write SQL queries to

Schema Diagram



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.

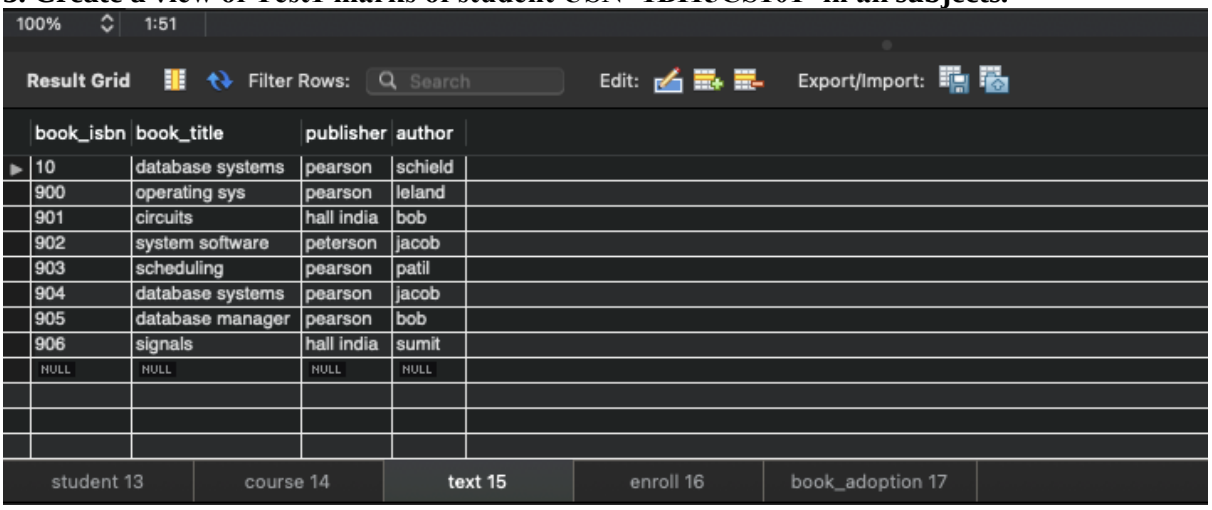
100% 1:51

Result Grid | Filter Rows: Search | Edit: | Export/Import:

courseno	cname	dept
111	os	cse
112	ec	ece
113	ss	ise
114	dbms	cse
115	signals	ece
NULL	NULL	NULL

student 13 | course 14 | text 15 | enroll 16 | book_adoption 17

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



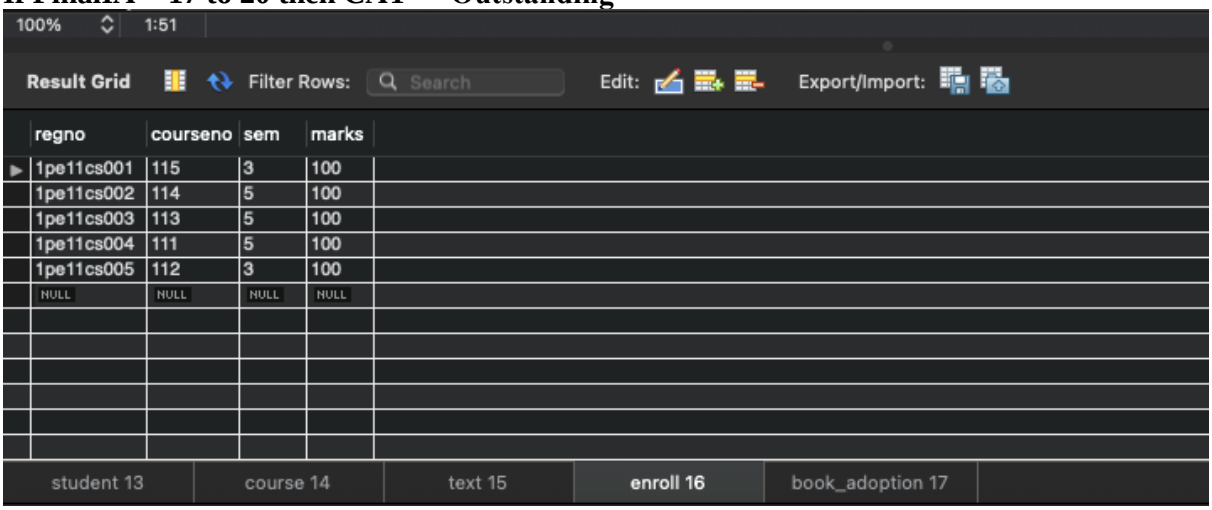
100% 1:51

Result Grid | Filter Rows: Search | Edit: | Export/Import:

book_isbn	book_title	publisher	author
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
NULL	NULL	NULL	NULL

student 13 | course 14 | text 15 | enroll 16 | book_adoption 17

4. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = 'Outstanding'



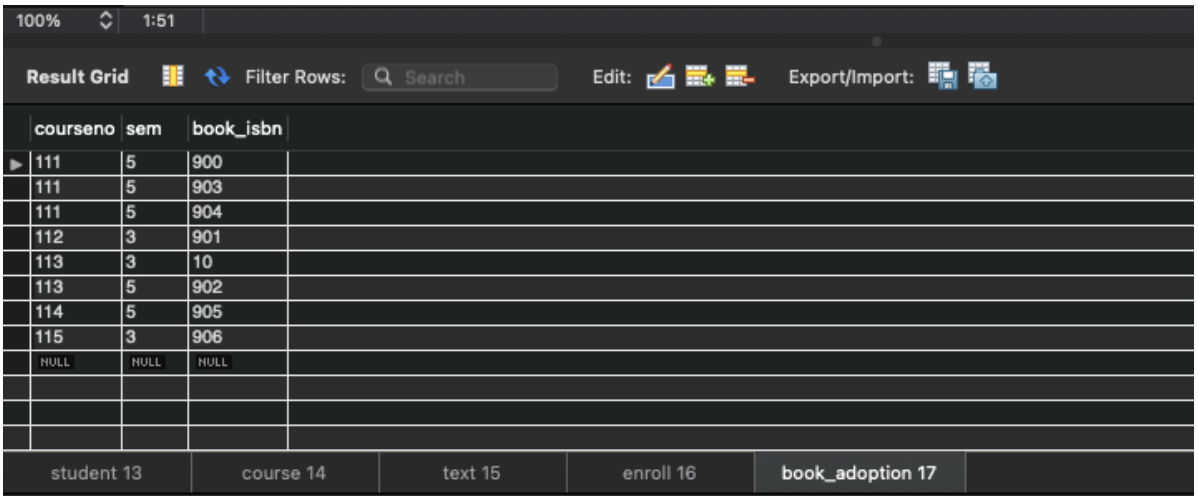
100% 1:51

Result Grid | Filter Rows: Search | Edit: | Export/Import:

regno	courseno	sem	marks
1pe11cs001	115	3	100
1pe11cs002	114	5	100
1pe11cs003	113	5	100
1pe11cs004	111	5	100
1pe11cs005	112	3	100
NULL	NULL	NULL	NULL

student 13 | course 14 | text 15 | enroll 16 | book_adoption 17

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.


100%
1:51
Result Grid
Filter Rows:
Search
Edit:
Export/Import:
courseno
sem
book_isbn
111 5 900
111 5 903
111 5 904
112 3 901
113 3 10
113 5 902
114 5 905
115 3 906
NULL NULL NULL
student 13
course 14
text 15
enroll 16
book_adoption 17