VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

NIKHITHA S (1BM24CS189)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled “Database Management Systems (23CS3PCDBM)” carried out by **NIKHITHA S (1BMCS24189),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

Ms. Divyashree S Assistant Professor Department of CSE, BMSCE	**Dr. Kavitha Sooda** Professor & HOD Department of CSE, BMSCE

Index

PROGRAM 1: INSURANCE DATABASE

Consider the Insurance database given below.

PERSON (driver_id: String, name: String, address: String)

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

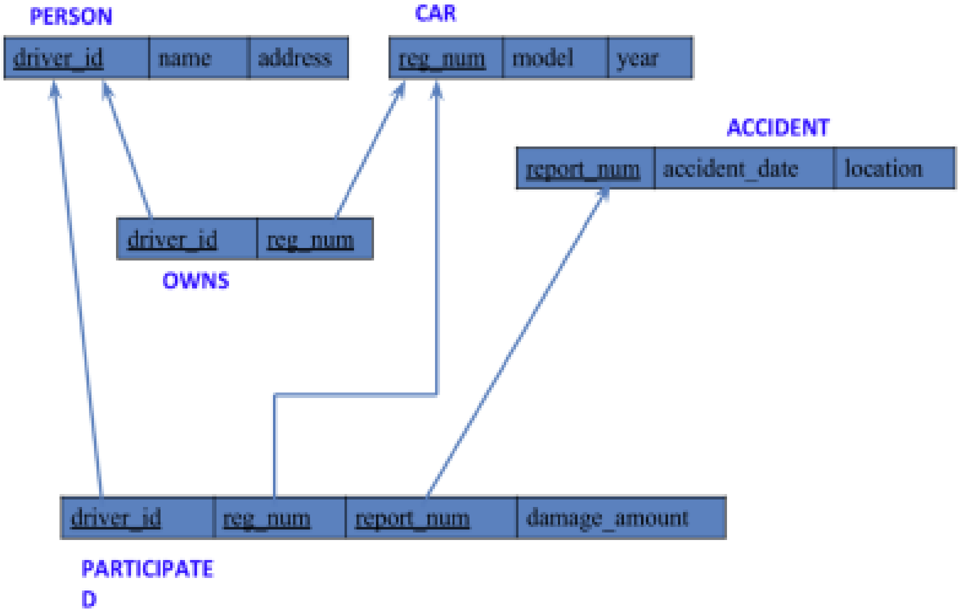
ACCIDENT (report_num: int, accident_date: date, location: String)

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. Display Accident date and location
iv. 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.
v. Add a new accident to the database.
vi. Display Accident date and location
vii. Display driver id who did accident with damage amount greater than or equal to Rs.25000

Schema Diagram

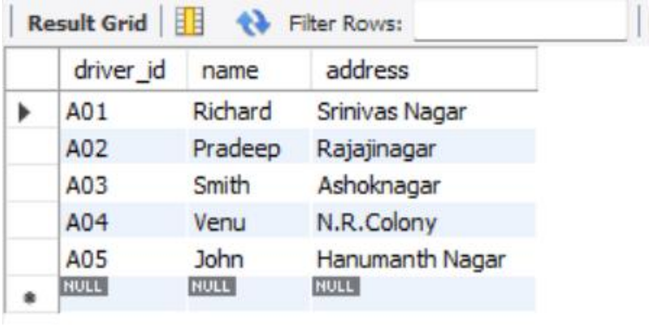


1. Create Tables

```
create database insurance;

use insurance;

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

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


create table accident (
   report_num int primary key,
   accident_date date,
   location varchar(20)
);


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)
);
create table participated (
   driver_id varchar(10),
   reg_num varchar(10),
   report_num integer,
   damage_amount integer,
   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)
);
```

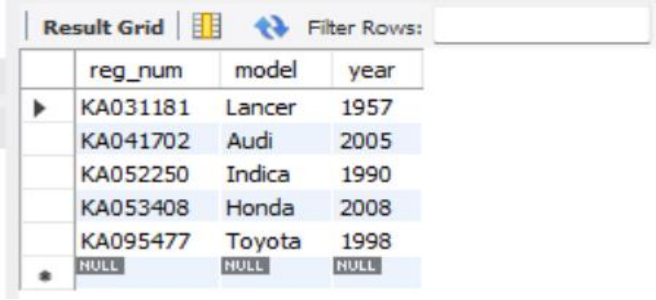
2. Insert Values

```
insert into person values ('a01','richard','srinivas nagar');

insert into person values ('a02','pradeep','rajajinagar');
insert into person values ('a03','smith','ashoknagar');
insert into person values ('a04','venu','n.r.colony');
insert into person values ('a05','john','hanumanth nagar');
select * from person;
```



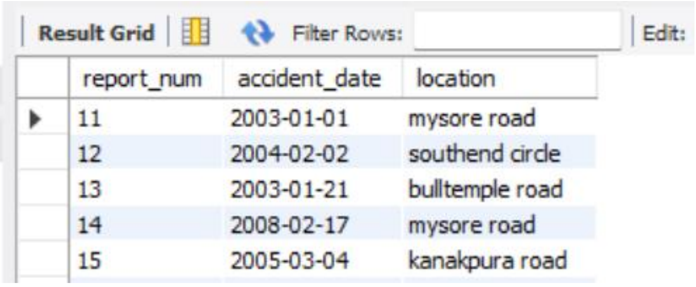
	driver_id	name	address
▶	A01	Richard	Srinivas Nagar
	A02	Pradeep	Rajajinagar
	A03	Smith	Ashoknagar
	A04	Venu	N.R.Colony
	A05	John	Hanumanth Nagar
*	NULL	NULL	NULL

```
insert into car values ('ka052250','indica',1990);
insert into car values ('ka031181','lancer',1957);
insert into car values ('ka095477','toyota',1998);
insert into car values ('ka053408','honda',2008);
insert into car values ('ka041702','audi',2005);

select * from car;
```



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

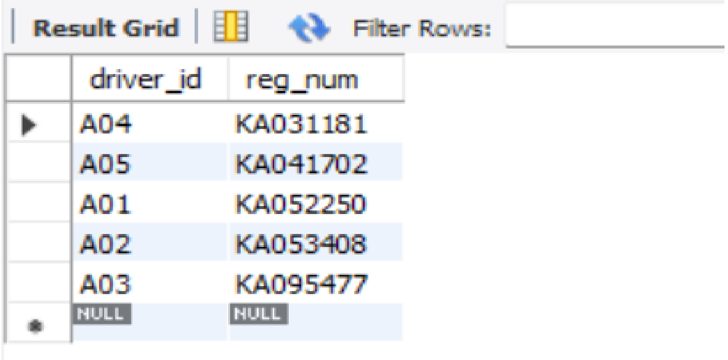
```
insert into accident values (11, '2003-01-01', 'mysore road');
insert into accident values (12, '2004-02-02', 'southend circle');
insert into accident values (13, '2003-01-21', 'bulltemple road');
insert into accident values (14, '2008-02-17', 'mysore road');
insert into accident values (15, '2005-03-04', 'kanakpura road');
select * from accident;
```



report_num	accident_date	location
11	2003-01-01	mysore road
12	2004-02-02	southend circle
13	2003-01-21	bulltemple road
14	2008-02-17	mysore road
15	2005-03-04	kanakpura road

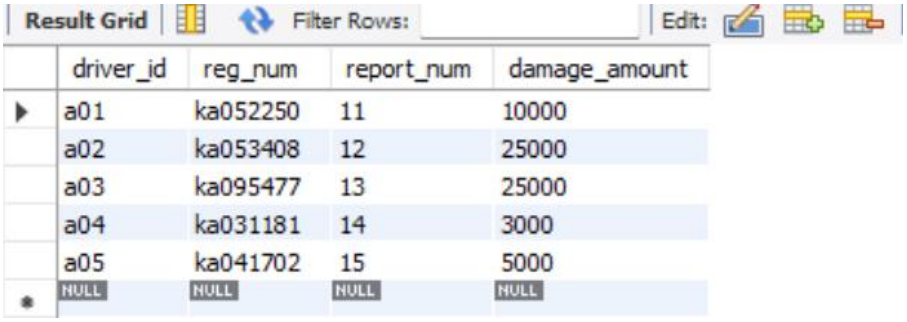
```
insert into owns values ('a01','ka052250');
insert into owns values ('a02','ka053408');
insert into owns values ('a03','ka095477');
insert into owns values ('a04','ka031181');
insert into owns values ('a05','ka041702');
```

```
select * from owns;
```



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

```
insert into participated values ('a01','ka052250',11,10000);
insert into participated values ('a02','ka053408',12,50000);
insert into participated values ('a03','ka095477',13,25000);
insert into participated values ('a04','ka031181',14,3000);
insert into participated values ('a05','ka041702',15,5000);
```

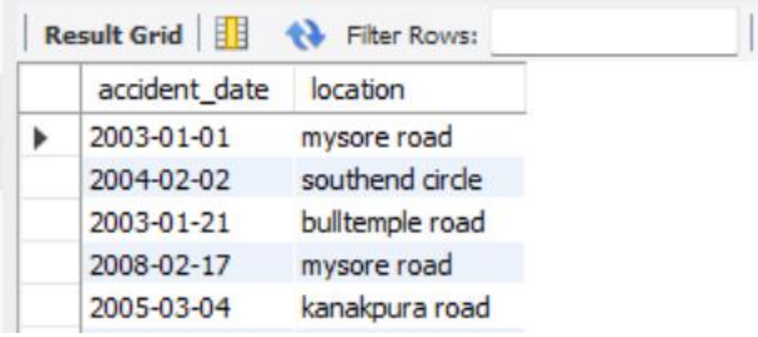


driver_id	reg_num	report_num	damage_amount
a01	ka052250	11	10000
a02	ka053408	12	25000
a03	ka095477	13	25000
a04	ka031181	14	3000
a05	ka041702	15	5000
NULL	NULL	NULL	NULL

3. Queries

(i) Display accident date and location

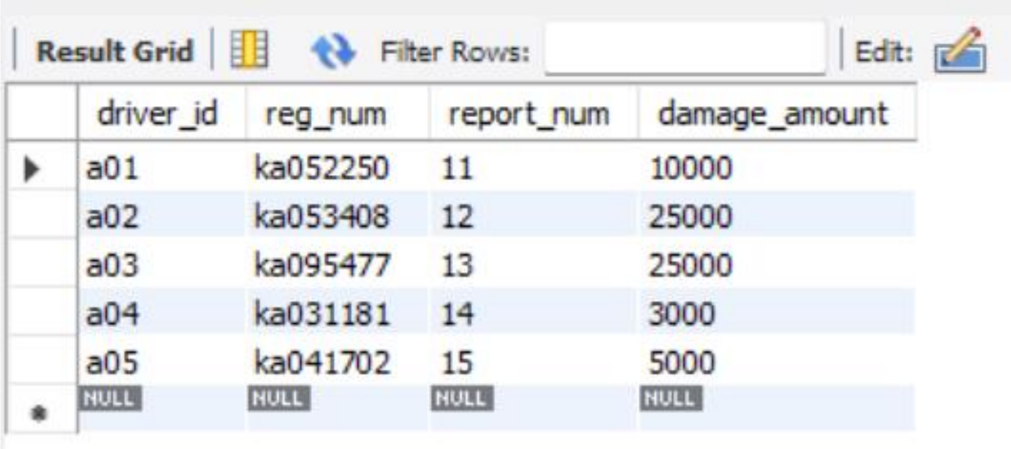
select accident_date, location from accident;



accident_date	location
2003-01-01	mysore road
2004-02-02	southend circle
2003-01-21	bulltemple road
2008-02-17	mysore road
2005-03-04	kanakpura road

(ii) Update the damage amount to 25000 for a specific car and report number

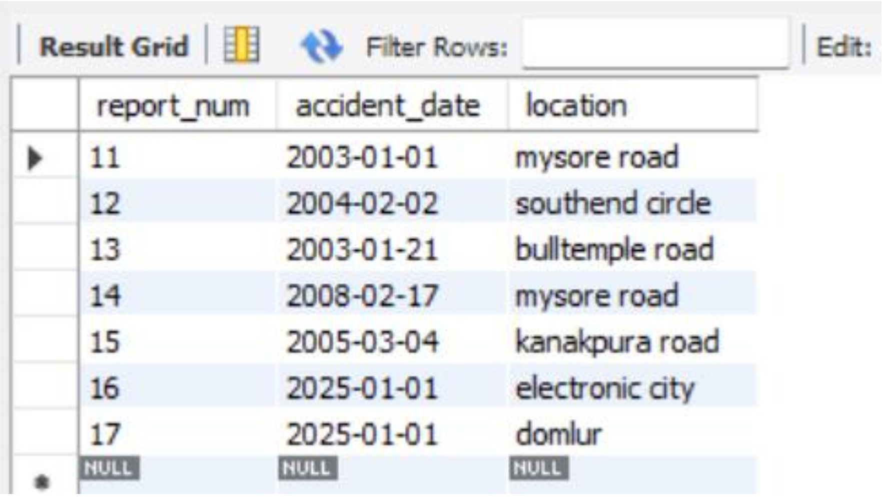
```
update participated
set damage_amount = 25000
where reg_num = 'ka053408' and report_num = 12;
select * from participated;
```



driver_id	reg_num	report_num	damage_amount
a01	ka052250	11	10000
a02	ka053408	12	25000
a03	ka095477	13	25000
a04	ka031181	14	3000
a05	ka041702	15	5000
NULL	NULL	NULL	NULL

(iii) Add a new accident to the database

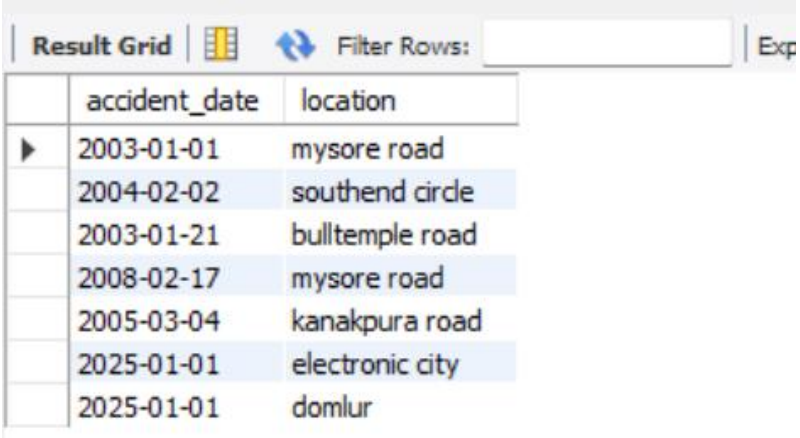
```
insert into accident (report_num, accident_date, location)
values (16, '2025-01-01', 'electronic city');
```



report_num	accident_date	location
11	2003-01-01	mysore road
12	2004-02-02	southend circle
13	2003-01-21	bulltemple road
14	2008-02-17	mysore road
15	2005-03-04	kanakpura road
16	2025-01-01	electronic city
17	2025-01-01	domlur
NULL	NULL	NULL

(iv) Display accident date and location again to verify the new record

select accident_date, location from accident;



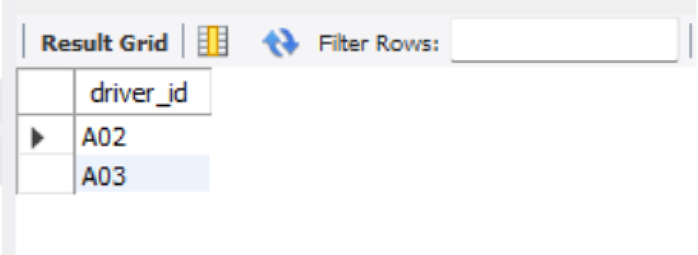
accident_date	location
2003-01-01	mysore road
2004-02-02	southend circle
2003-01-21	bulltemple road
2008-02-17	mysore road
2005-03-04	kanakpura road
2025-01-01	electronic city
2025-01-01	domlur

(v) Display driver id who did accident with damage amount greater than or equal to Rs.25000

select driver_id

from participated

where damage_amount >= 25000;



driver_id
A02
A03

PROGRAM 2. More Queries on Insurance Database

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)

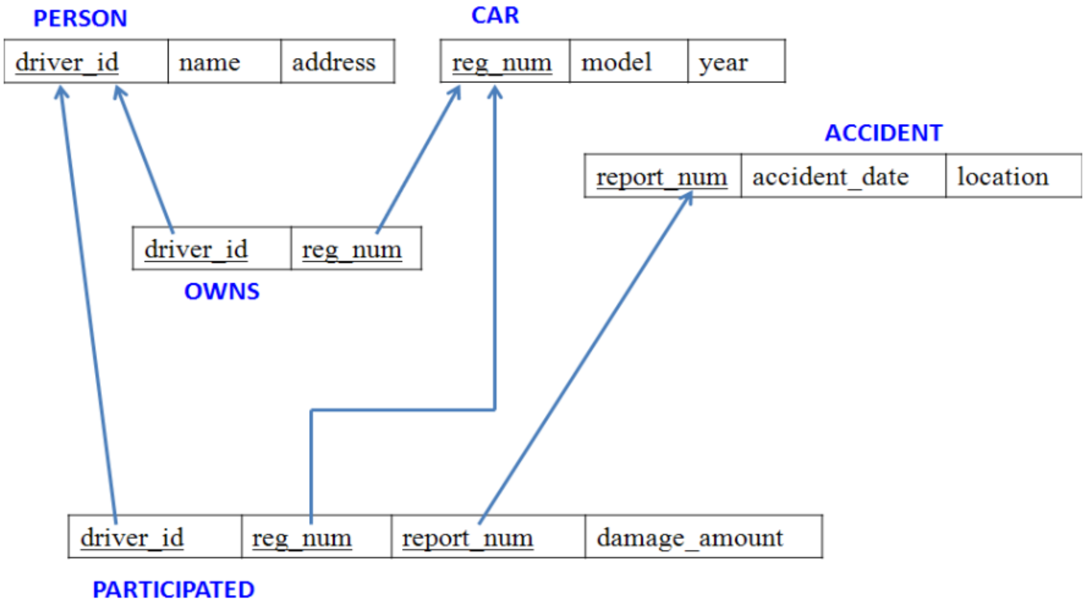
OWNS (driver_id: String, reg_num: String)

PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)

Create the above tables by properly specifying the primary keys and the foreign keys as done in “Program-1”week’s lab and Enter at least five tuples for each relation.

i. Display the entire CAR relation in the ascending order of manufacturing year.
ii. Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
iii. Find the total number of people who owned cars that involved in accidents in 2008.
iv. List the entire participated relation in the Descending Order of Damage Amount.
v. Find the Average Damage Amount.
vi. Delete the tuple whose Damage Amount is below the Average Damage Amount
vii. List the name of drivers whose Damage is Greater than the Average Damage Amount. viii. Find Maximum Damage Amount.

Schema Diagram

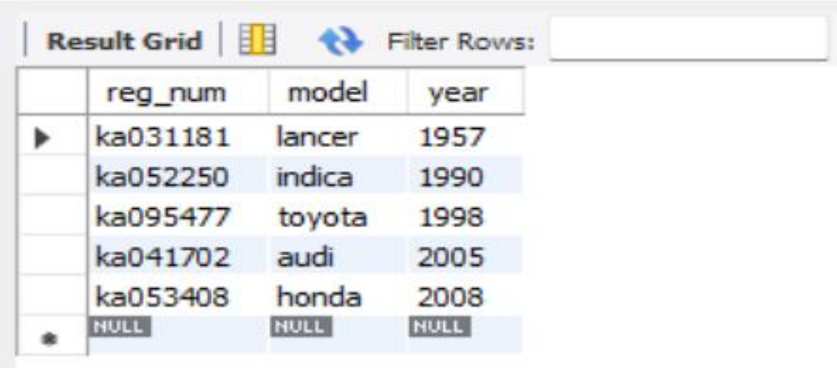


3. Queries

```
use insurance;
```

(i) Display entire car relation in ascending order of manufacturing year

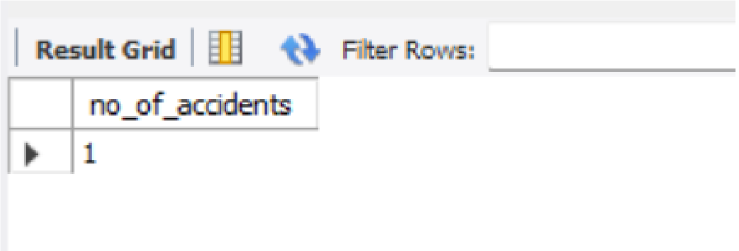
```
select *
from car
order by year asc;
```



	reg_num	model	year
▶	ka031181	lancer	1957
	ka052250	indica	1990
	ka095477	toyota	1998
	ka041702	audi	2005
	ka053408	honda	2008
*	NULL	NULL	NULL

(ii) Number of accidents involving cars of model 'Honda'

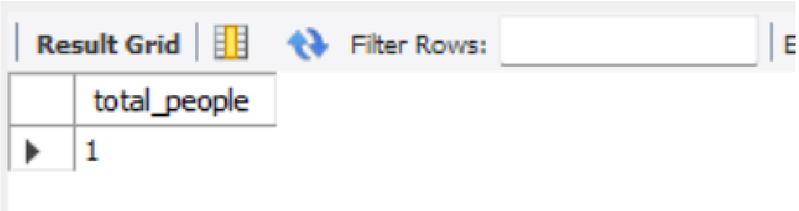
```
select count(*) as no_of_accidents
from participated p
join car c on p.reg_num = c.reg_num
where c.model = 'Honda';
```



	no_of_accidents
▶	1

(iii) Total number of people who owned cars involved in accidents in 2008

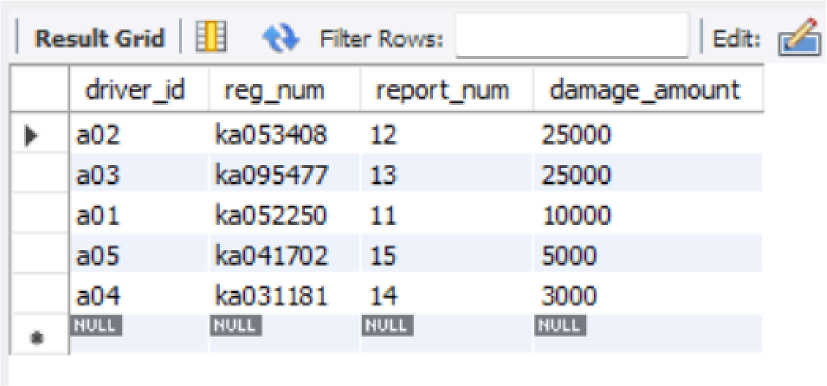
```
select count(distinct o.driver_id) as total_people
from owns o
join participated p on o.reg_num = p.reg_num
join accident a on p.report_num = a.report_num
where year(a.accident_date) = 2008;
```



	total_people
▶	1

(iv) List participated relation in descending order of damage amount

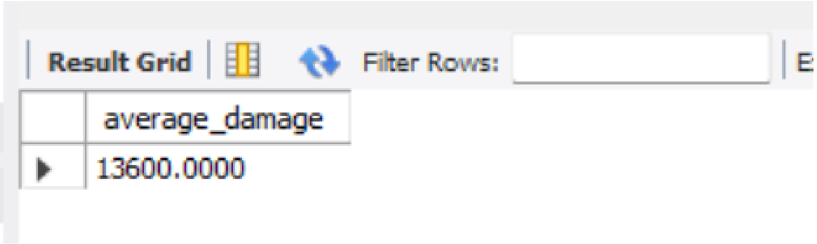
```
select *
from participated
order by damage_amount desc;
```



	driver_id	reg_num	report_num	damage_amount
▶	a02	ka053408	12	25000
	a03	ka095477	13	25000
	a01	ka052250	11	10000
	a05	ka041702	15	5000
	a04	ka031181	14	3000
*	NULL	NULL	NULL	NULL

(v) Find average damage amount

```
select avg(damage_amount) as average_damage
from participated;
```

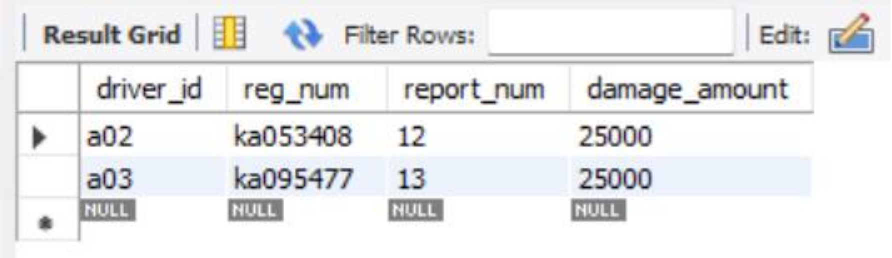


	average_damage
▶	13600.0000

(vi) Delete tuples with damage amount below average damage

```
set sql_safe_updates = 0;
delete from participated
where damage_amount < (
   select avg(damage_amount)
```

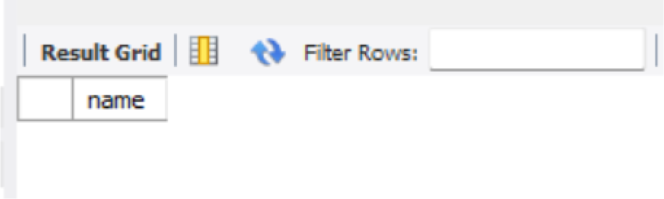
```
    from (select damage_amount from participated) as temp
);
select * from participated;
```



	driver_id	reg_num	report_num	damage_amount
▶	a02	ka053408	12	25000
	a03	ka095477	13	25000
*	NULL	NULL	NULL	NULL

(vii) List names of drivers whose damage is greater than average damage

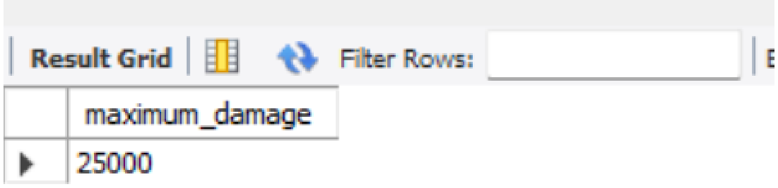
```
select p.name
from person p
join participated pa on p.driver_id = pa.driver_id
where pa.damage_amount > (
    select avg(damage_amount)
    from participated
);
```



	name

(viii) Find maximum damage amount

```
select max(damage_amount) as maximum_damage
from participated;
```



	maximum_damage
▶	25000

PROGRAM 3: Bank Database

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.

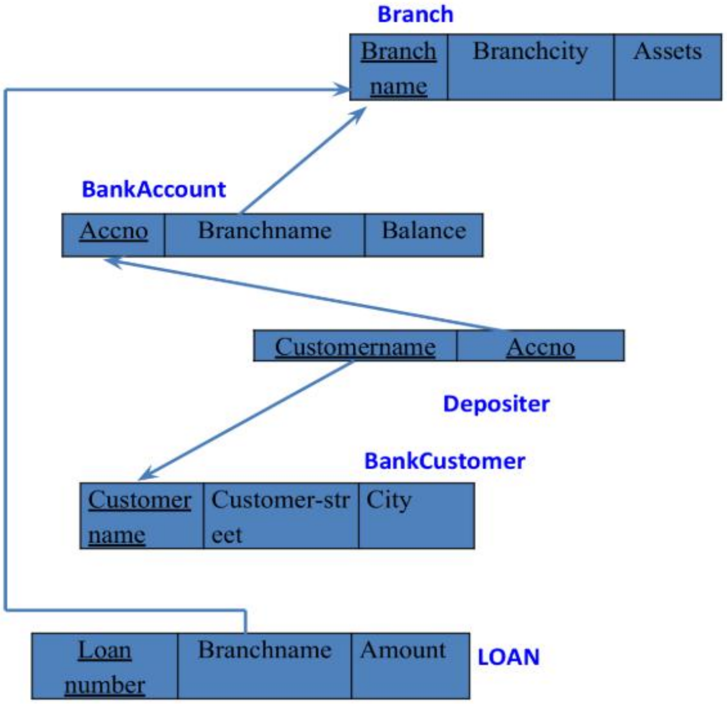
ii. Enter at least five tuples for each relation.

iii. Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.

iv. Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).

v. Create a view which gives each branch the sum of the amount of all the loans at the branch.

Schema Diagram

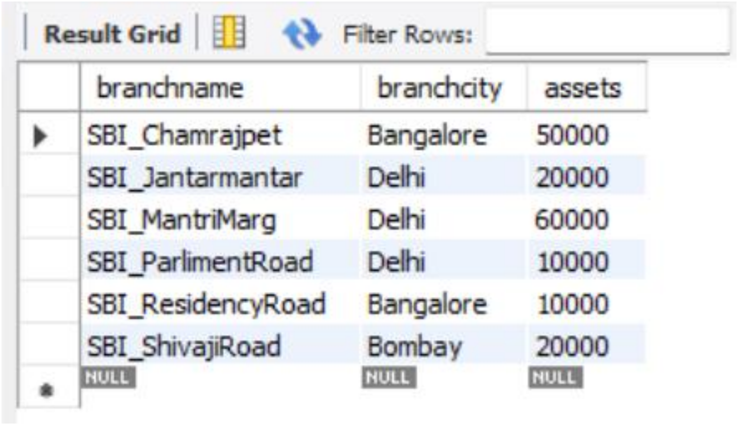


1. Create tables

```
create database bank2;
use bank2;

create table branch (
   branchname varchar(20) primary key,
   branchcity varchar(30),
   assets int
);

create table bankaccount (
   accno int primary key,
   branchname varchar(30),
   balance int,
   foreign key (branchname) references branch(branchname)
);

create table bankcustomer (
   customername varchar(20) primary key,
   customer_street varchar(30),
   customercity varchar(35)
);

create table depositer (
   customername varchar(20),
   accno int,
   primary key(customername, accno),
   foreign key (accno) references bankaccount(accno),
   foreign key (customername) references bankcustomer(customername)
);

create table loan (
   loan_number int primary key,
   branchname varchar(30),
   amount int,
   foreign key (branchname) references branch(branchname)
);
```

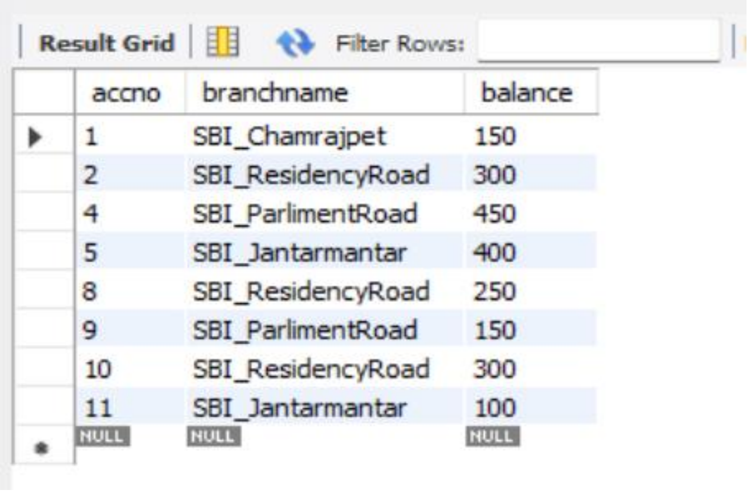
2. Insert Values

```
insert into branch values
('SBI_Chamrajpet','Bangalore',50000),
('SBI_ResidencyRoad','Bangalore',10000),
('SBI_ShivajiRoad','Bombay',20000),
('SBI_ParlimentRoad','Delhi',10000),
('SBI_Jantarmantar','Delhi',20000);
insert into branch values('SBI_MantriMarg','Delhi',60000);
select * from branch;
```



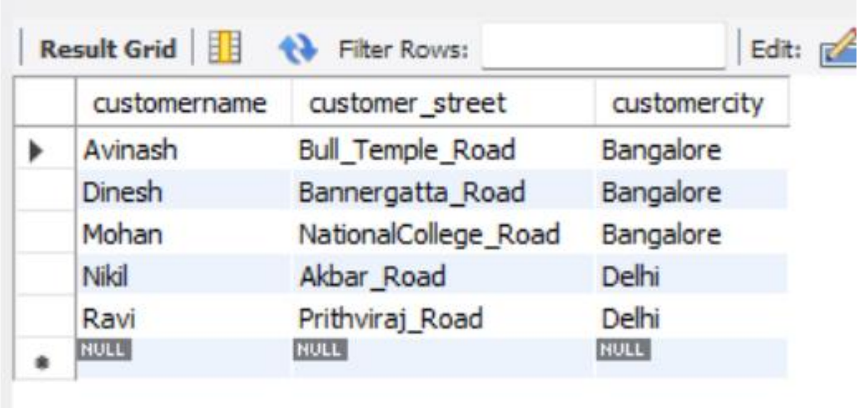
branchname	branchcity	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_MantriMarg	Delhi	60000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

```
insert into bankaccount values
(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);
```



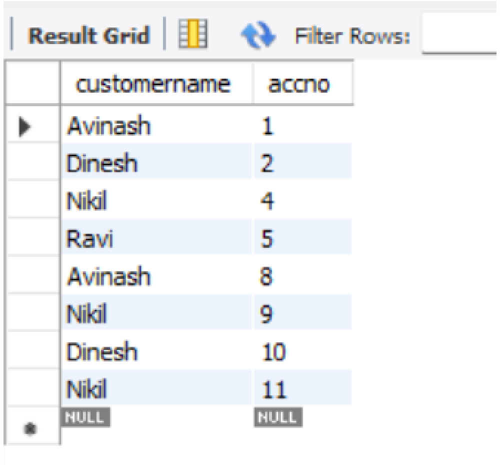
accno	branchname	balance
1	SBI_Chamrajpet	150
2	SBI_ResidencyRoad	300
4	SBI_ParlimentRoad	450
5	SBI_Jantarmantar	400
8	SBI_ResidencyRoad	250
9	SBI_ParlimentRoad	150
10	SBI_ResidencyRoad	300
11	SBI_Jantarmantar	100
NULL	NULL	NULL

```
insert into bankcustomer values
('Avinash','Bull_Temple_Road','Bangalore'),
('Dinesh','Bannergatta_Road','Bangalore'),
('Mohan','NationalCollege_Road','Bangalore'),
('Nikil','Akbar_Road','Delhi'),
('Ravi','Prithviraj_Road','Delhi');
```



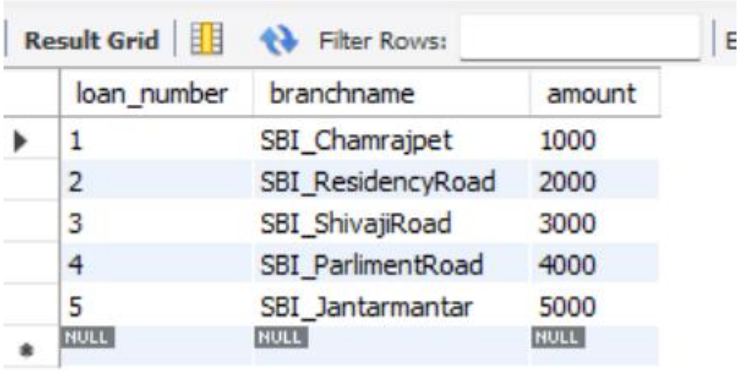
customername	customer_street	customercity
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

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



	customername	accno
▶	Avinash	1
	Dinesh	2
	Nikil	4
	Ravi	5
	Avinash	8
	Nikil	9
	Dinesh	10
	Nikil	11
*	NULL	NULL

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

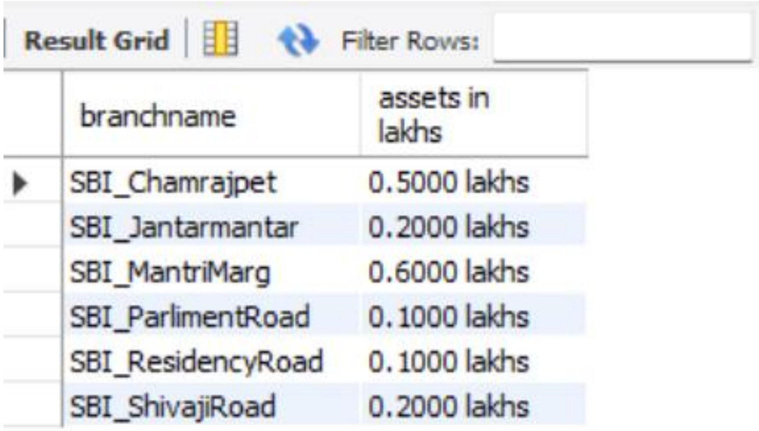


	loan_number	branchname	amount
▶	1	SBI_Chamrajpet	1000
	2	SBI_ResidencyRoad	2000
	3	SBI_ShivajiRoad	3000
	4	SBI_ParlimentRoad	4000
	5	SBI_Jantarmantar	5000
*	NULL	NULL	NULL

3. Queries

(i) Display branch name and assets in lakhs of rupees

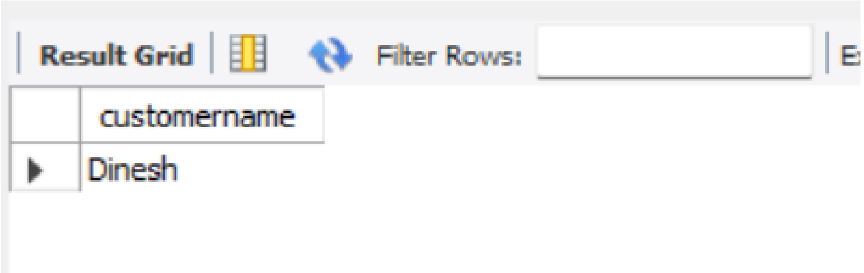
```
select branchname, concat(assets/100000,' lakhs') as `assets in lakhs`
from branch;
```



branchname	assets in lakhs
SBI_Chamrajpet	0.5000 lakhs
SBI_Jantarmantar	0.2000 lakhs
SBI_MantriMarg	0.6000 lakhs
SBI_ParlimentRoad	0.1000 lakhs
SBI_ResidencyRoad	0.1000 lakhs
SBI_ShivajiRoad	0.2000 lakhs

(ii) Find customers who have at least two accounts at the same branch

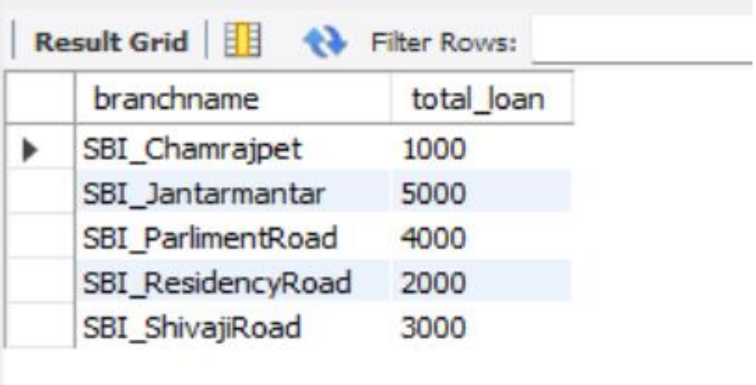
```
select d.customername
from depositer d
join bankaccount b on d.accno = b.accno
where b.branchname = 'SBI_ResidencyRoad'
group by d.customername
having count(d.accno) >= 2;
```



customername
Dinesh

(iii) Create a view which gives each branch the sum of the amount of all the loans at the branch

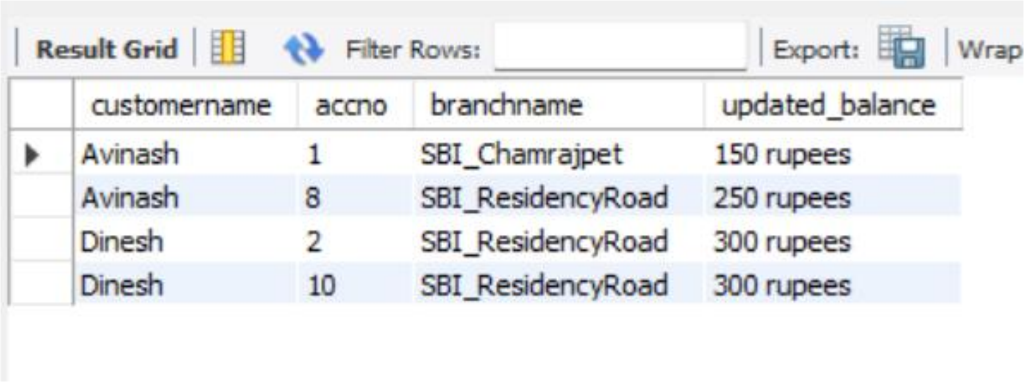
```
create view sum_of_loans as
select branchname, sum(amount) as total_loan
from loan
group by branchname;

-- to check the view
select * from sum_of_loans;
```



branchname	total_loan
SBI_Chamrajpet	1000
SBI_Jantarmantar	5000
SBI_ParlimentRoad	4000
SBI_ResidencyRoad	2000
SBI_ShivajiRoad	3000

(iv) Add 1000 rupees to account balance for customers residing in Bangalore

```
-- display updated balances for customers in Bangalore
select
    c.customername,
    b.accno,
    b.branchname,
    concat(b.balance, ' rupees') as updated_balance
from bankaccount b
join depositer d on b.accno = d.accno
join bankcustomer c on d.customername = c.customername
where c.customercity = 'Bangalore';
```



customername	accno	branchname	updated_balance
Avinash	1	SBI_Chamrajpet	150 rupees
Avinash	8	SBI_ResidencyRoad	250 rupees
Dinesh	2	SBI_ResidencyRoad	300 rupees
Dinesh	10	SBI_ResidencyRoad	300 rupees

PROGRAM 4: More Queries on Bank Database

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. Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi). ii. iii. iv. v. vi.

ii. Find all customers who have a loan at the bank but do not have an account.

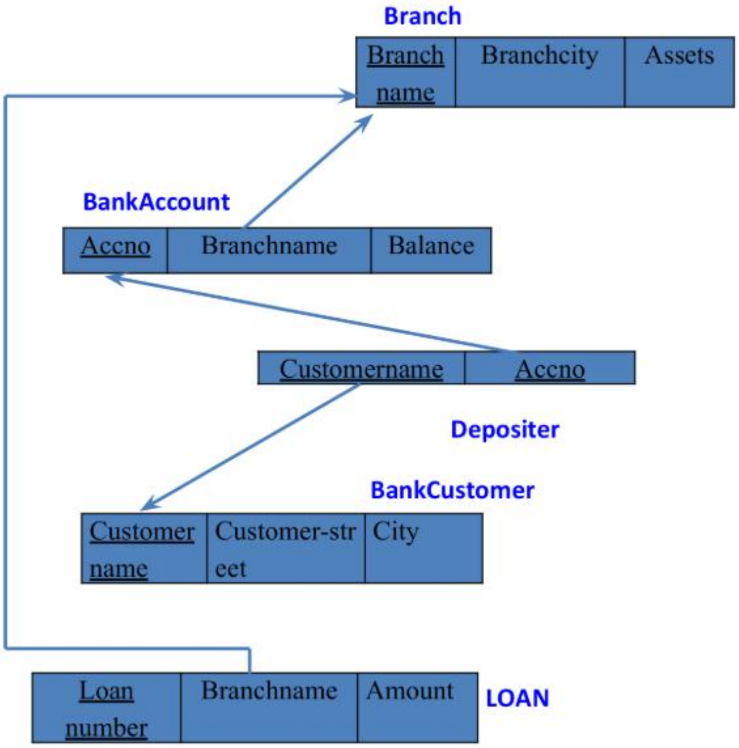
iii. Find all customers who have both an account and a loan at the Bangalore branch

iv. Find the names of all branches that have greater assets than all branches located in Bangalore.

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

vi. Update the Balance of all accounts by 5%

Schema Diagram

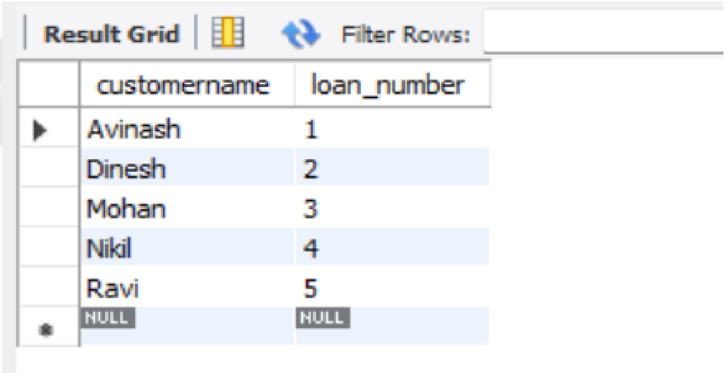


1. Create Tables

```
use bank2;
create table  borrower (
    customername varchar(20),
    loan_number int,
    primary key(customername, loan_number),
    foreign key (customername) references bankcustomer(customername),
    foreign key (loan_number) references loan(loan_number)
);
```

2. Insert Values

```
-- insert sample data into borrower
insert into borrower values
('Avinash', 1),
('Dinesh', 2),
('Nikil', 4),
('Ravi', 5),
('Mohan', 3);
```

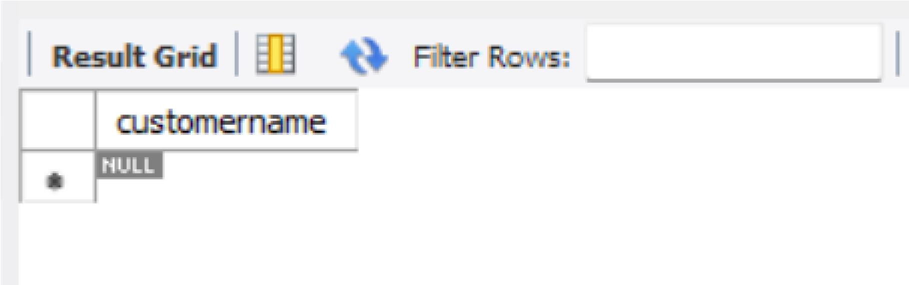


customername	loan_number
Avinash	1
Dinesh	2
Mohan	3
Nikil	4
Ravi	5
NULL	NULL

3. Queries

(i) Find all customers who have an account at all the branches located in a specific city (Delhi)

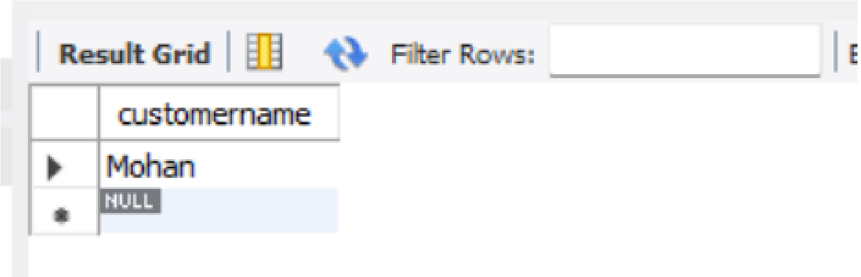
```
select bc.customername
from bankcustomer bc
where not exists (
    select b.branchname
    from branch b
    where b.branchcity = 'Delhi'
    and b.branchname not in (
            select ba.branchname
            from bankaccount ba
          join depositer d on ba.accno = d.accno
          where d.customername = bc.customername
      )
   );
```



	customername
*	NULL

(ii) Find all customers who have a loan at the bank but do not have an account

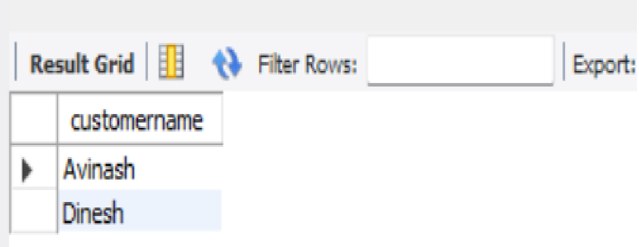
```
select bc.customername
from bankcustomer bc
where bc.customername in (select customername from borrower)
and bc.customername not in (select customername from depositer);
```



	customername
▶	Mohan
*	NULL

(iii) Find all customers who have both an account and a loan at the Bangalore branch

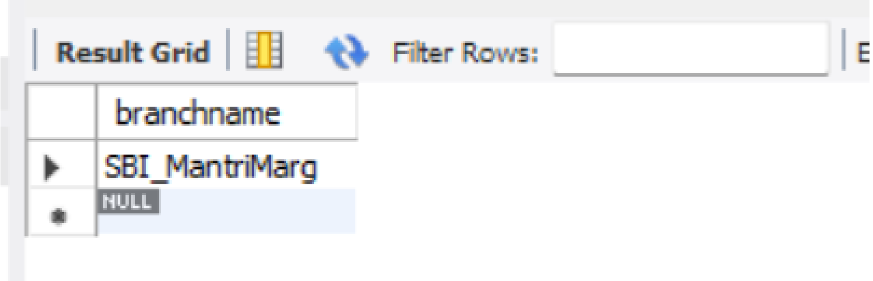
```
select distinct bc.customername
from bankcustomer bc
join depositer d on bc.customername = d.customername
join bankaccount ba on d.accno = ba.accno
join borrower b on bc.customername = b.customername
join loan l on b.loan_number = l.loan_number
where ba.branchname in (select branchname from branch where
branchcity='Bangalore')
  and l.branchname in (select branchname from branch where branchcity='Bangalore');
```



customername
Avinash
Dinesh

(iv) **Find the names of all branches that have greater assets than all branches located in Bangalore**

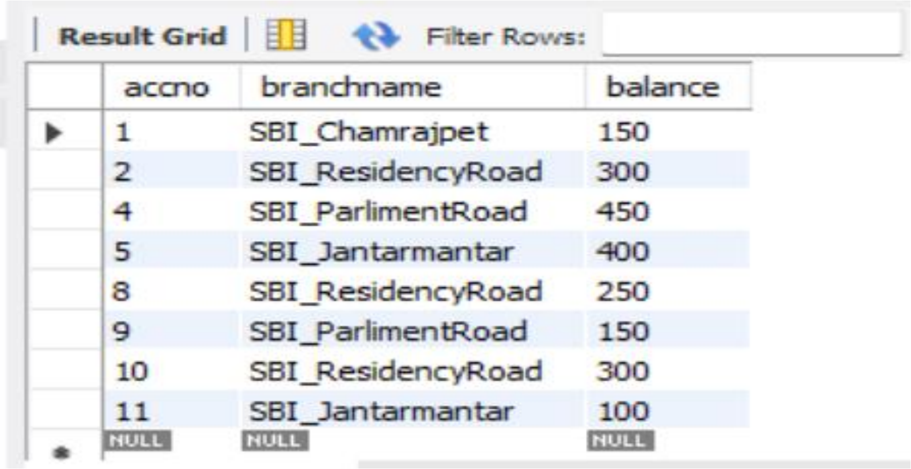
```
select branchname
from branch
where assets > (select max(assets) from branch where branchcity='Bangalore');
```



branchname
SBI_MantriMarg
NULL

(v) **Demonstrate how to delete all account tuples at every branch located in a specific city (Bombay)**

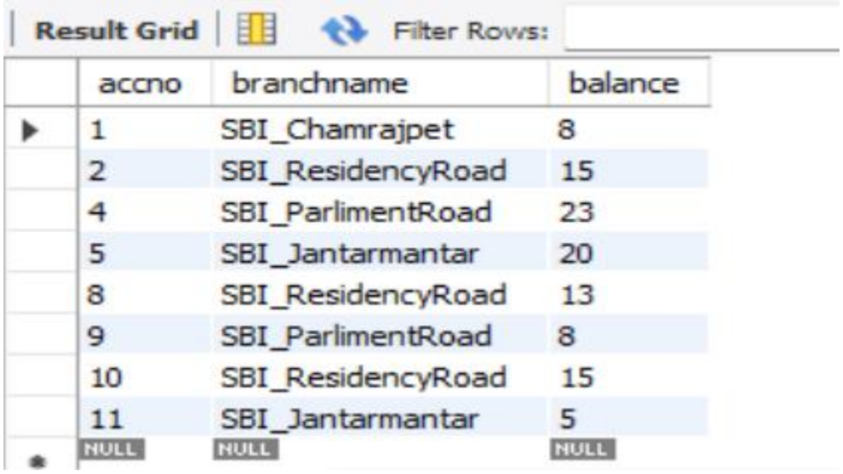
```
delete ba
from bankaccount ba
join branch b on ba.branchname = b.branchname
where b.branchcity = 'Bombay';
```



accno	branchname	balance
1	SBI_Chamrajpet	150
2	SBI_ResidencyRoad	300
4	SBI_ParlimentRoad	450
5	SBI_Jantarmantar	400
8	SBI_ResidencyRoad	250
9	SBI_ParlimentRoad	150
10	SBI_ResidencyRoad	300
11	SBI_Jantarmantar	100
NULL	NULL	NULL

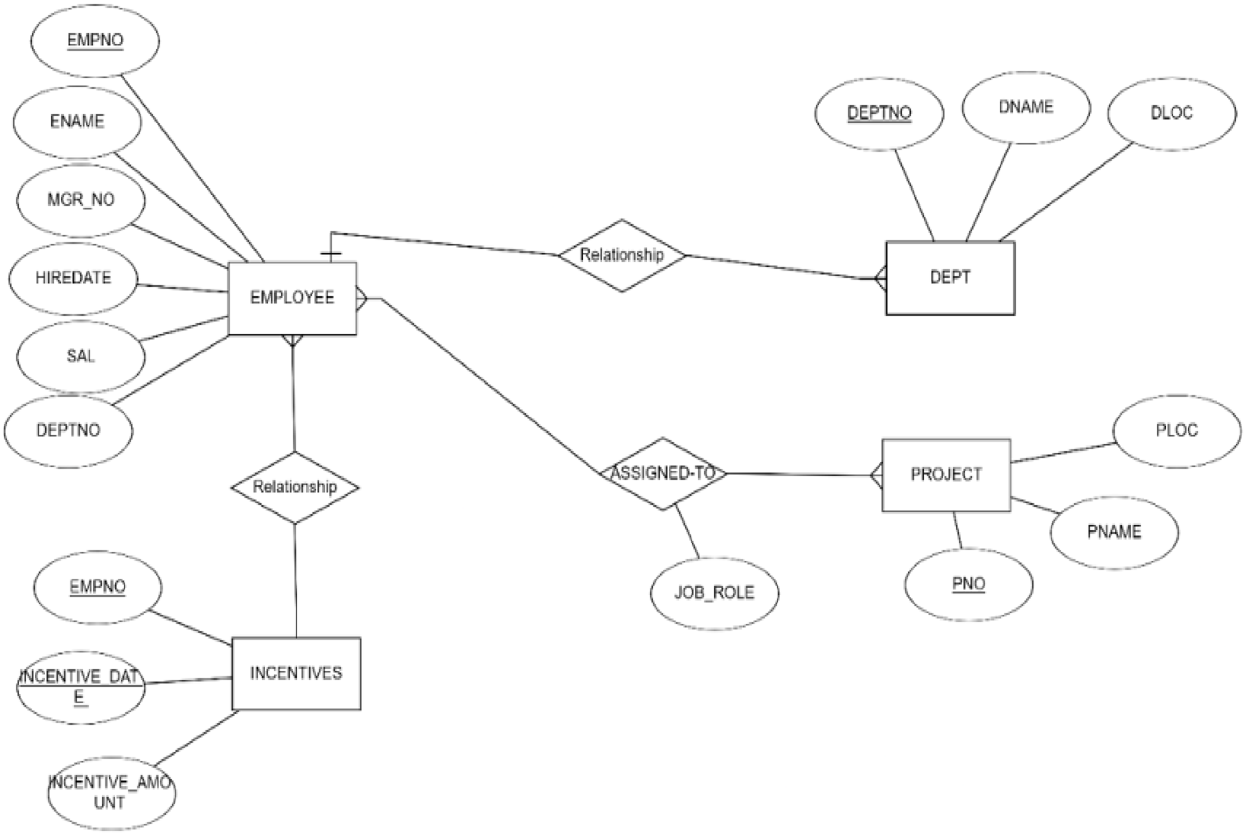
(vi) Update the Balance of all accounts by 5%

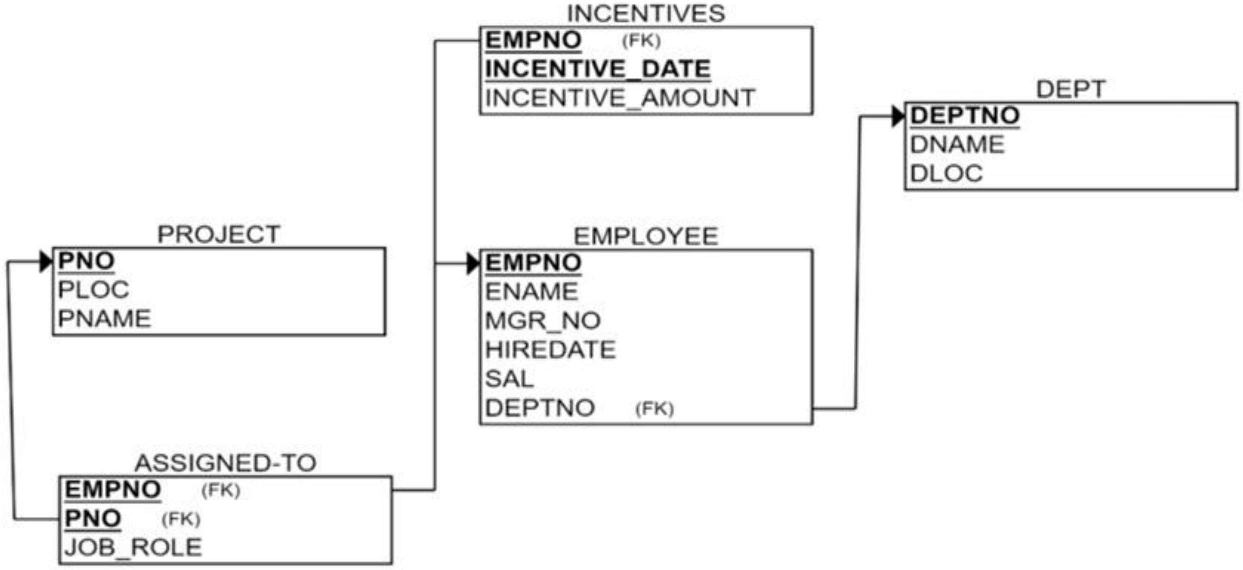
```
update bankaccount
set balance = balance * 0.05
where accno > 0;
select * from bankaccount;
```



accno	branchname	balance
1	SBI_Chamrajpet	8
2	SBI_ResidencyRoad	15
4	SBI_ParlimentRoad	23
5	SBI_Jantarmantar	20
8	SBI_ResidencyRoad	13
9	SBI_ParlimentRoad	8
10	SBI_ResidencyRoad	15
11	SBI_Jantarmantar	5
NULL	NULL	NULL

PROGRAM 5: Employee Database





Using Scheme diagram,

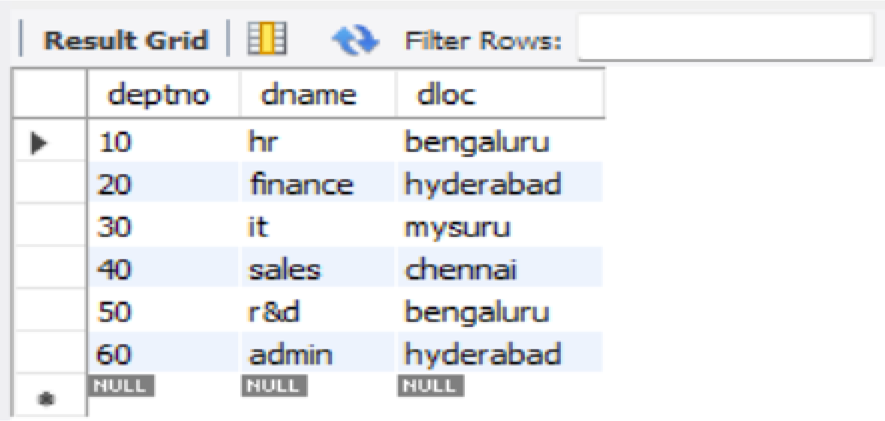
1. create tables by properly specifying the primary keys and the foreign keys.
2. Enter greater than five tuples for each table.
3. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru
4. Get Employee ID's of those employees who didn't receive incentives
5. Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.

1. Create Tables

```
create database employe;
use employe;
-- create dept table
create table dept (
   deptno int primary key,
   dname varchar(50),
   dloc varchar(50)
);

-- create employee table
create table employee (
   empno int primary key,
   ename varchar(50),
   mgrno int,
   hiredate date,
   sal decimal(10,2),
   deptno int,
   foreign key (deptno) references dept(deptno)
);

-- create project table
create table project (
   pno int primary key,
   ploc varchar(50),
   pname varchar(50)
);
-- create assignedto table
create table assignedto (
    empno int,
    pno int,
    jobrole varchar(50),
    primary key (empno, pno),
    foreign key (empno) references employee(empno),
    foreign key (pno) references project(pno)
);

-- create incentives table
create table incentives (
    empno int,
    incentivedate date,
    incentiveamount decimal(10,2),
    primary key (empno, incentivedate),
    foreign key (empno) references employee(empno)
);
```

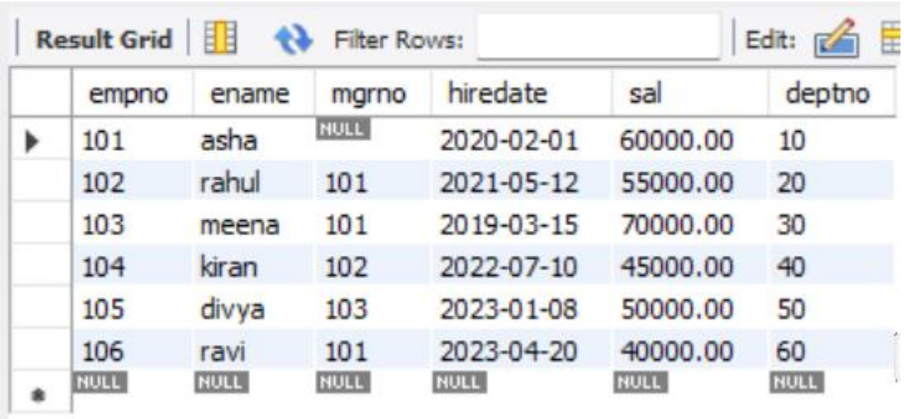
2. Insert Values

```
-- insert values into dept
insert into dept values
(10, 'hr', 'bengaluru'),
(20, 'finance', 'hyderabad'),
(30, 'it', 'mysuru'),
(40, 'sales', 'chennai'),
(50, 'r&d', 'bengaluru'),
(60, 'admin', 'hyderabad');
```



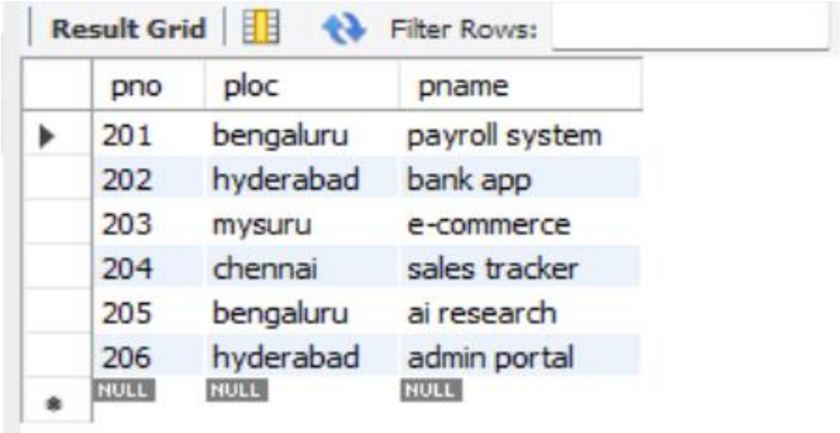
deptno	dname	dloc
10	hr	bengaluru
20	finance	hyderabad
30	it	mysuru
40	sales	chennai
50	r&d	bengaluru
60	admin	hyderabad
NULL	NULL	NULL

```
-- insert values into employee
insert into employee values
(101, 'asha', null, '2020-02-01', 60000, 10),
(102, 'rahul', 101, '2021-05-12', 55000, 20),
(103, 'meena', 101, '2019-03-15', 70000, 30),
(104, 'kiran', 102, '2022-07-10', 45000, 40),
(105, 'divya', 103, '2023-01-08', 50000, 50),
(106, 'ravi', 101, '2023-04-20', 40000, 60);
```



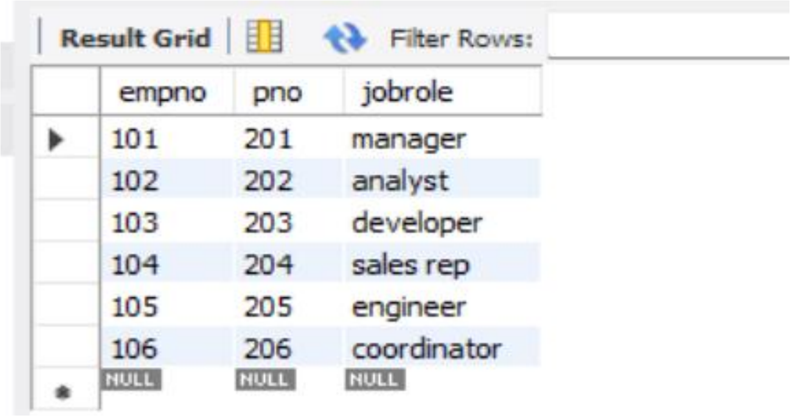
	empno	ename	mgrno	hiredate	sal	deptno
▶	101	asha	NULL	2020-02-01	60000.00	10
	102	rahul	101	2021-05-12	55000.00	20
	103	meena	101	2019-03-15	70000.00	30
	104	kiran	102	2022-07-10	45000.00	40
	105	divya	103	2023-01-08	50000.00	50
	106	ravi	101	2023-04-20	40000.00	60
*	NULL	NULL	NULL	NULL	NULL	NULL

```
-- insert values into project
insert into project values
(201, 'bengaluru', 'payroll system'),
(202, 'hyderabad', 'bank app'),
(203, 'mysuru', 'e-commerce'),
(204, 'chennai', 'sales tracker'),
(205, 'bengaluru', 'ai research'),
(206, 'hyderabad', 'admin portal');
```



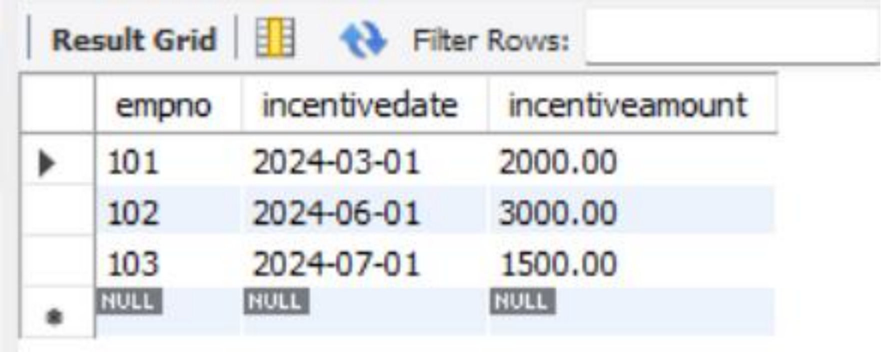
	pno	ploc	pname
▶	201	bengaluru	payroll system
	202	hyderabad	bank app
	203	mysuru	e-commerce
	204	chennai	sales tracker
	205	bengaluru	ai research
	206	hyderabad	admin portal
*	NULL	NULL	NULL

```
-- insert values into assignedto
insert into assignedto values
(101, 201, 'manager'),
(102, 202, 'analyst'),
(103, 203, 'developer'),
(104, 204, 'sales rep'),
(105, 205, 'engineer'),
(106, 206, 'coordinator');
```



	empno	pno	jobrole
▶	101	201	manager
	102	202	analyst
	103	203	developer
	104	204	sales rep
	105	205	engineer
	106	206	coordinator
*	NULL	NULL	NULL

```
-- insert values into incentives

insert into incentives values
(101, '2024-03-01', 2000),
(102, '2024-06-01', 3000),
(103, '2024-07-01', 1500);
```

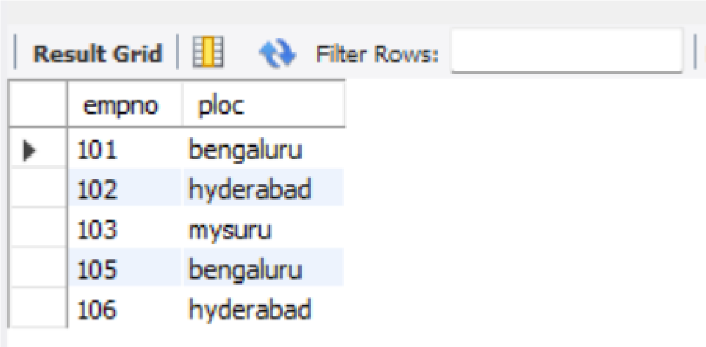


empno	incentivedate	incentiveamount
101	2024-03-01	2000.00
102	2024-06-01	3000.00
103	2024-07-01	1500.00
NULL	NULL	NULL

3. Queries

(i) Retrieve employee numbers of employees working on projects in bengaluru, hyderabad, or Mysuru

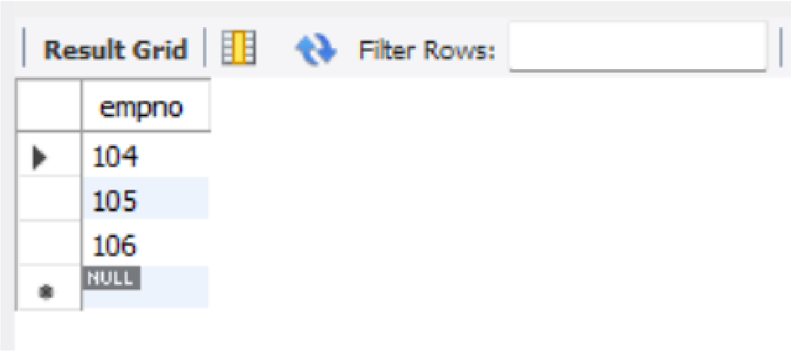
```
select distinct e.empno,p.ploc
from employee e
join assignedto a on e.empno = a.empno
join project p on a.pno = p.pno
where p.ploc in ('bengaluru', 'hyderabad', 'mysuru');
```



empno	ploc
101	bengaluru
102	hyderabad
103	mysuru
105	bengaluru
106	hyderabad

(ii) Get employee ids of employees who did not receive incentives

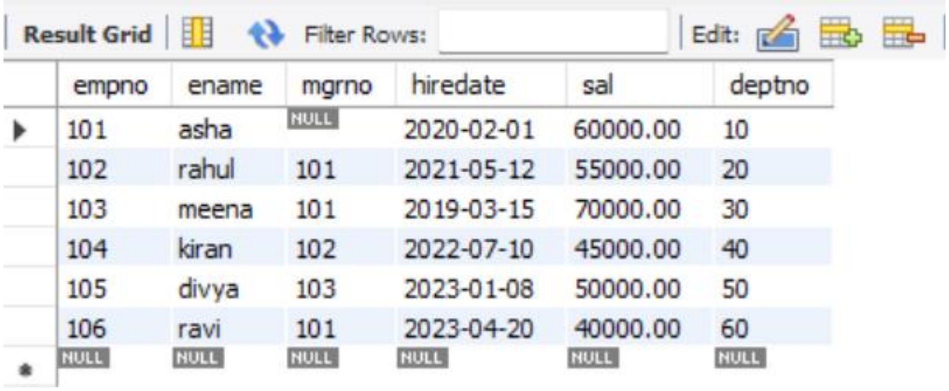
```
select e.empno
from employee e
where e.empno not in (
    select empno from incentives
);
```



empno
104
105
106
NULL

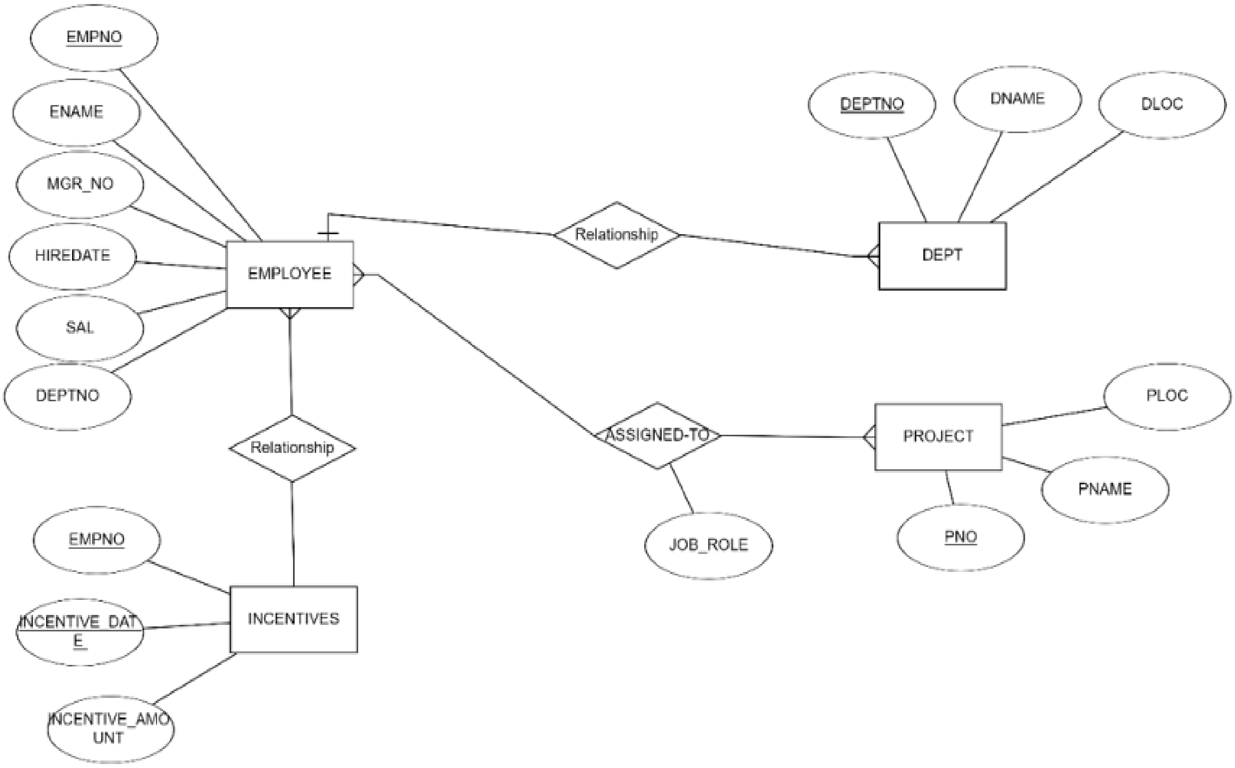
(iii) Employees working on project location same as department location

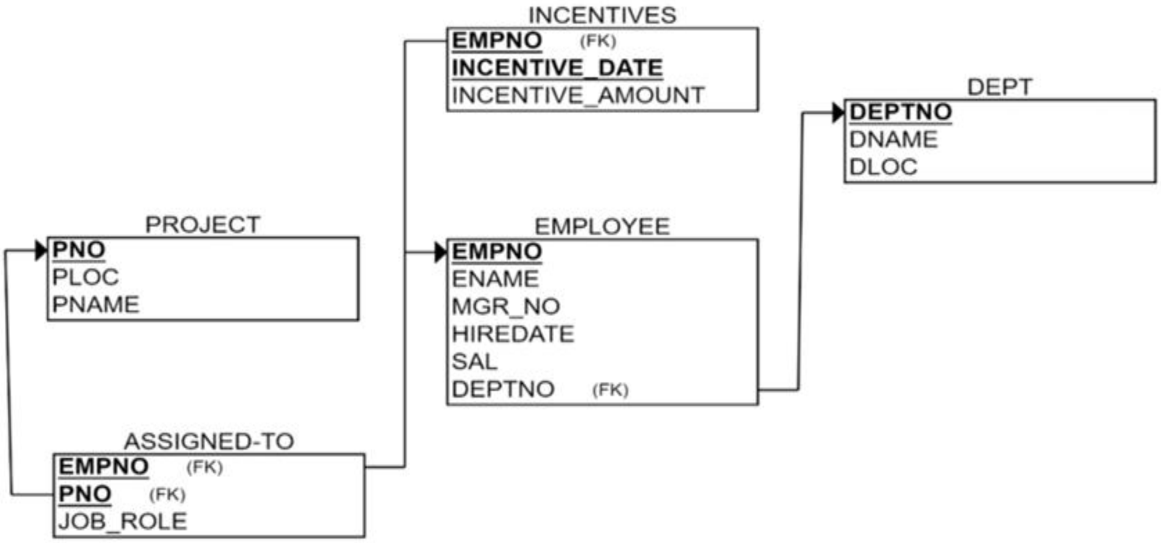
```
select
e.ename,
e.empno,
d.dname,
a.jobrole,
d.dloc as dept_location,
p.ploc as project_location
from employee e
join dept d on e.deptno = d.deptno
join assignedto a on e.empno = a.empno
join project p on a.pno = p.pno
where d.dloc = p.ploc;
```



empno	ename	mgrno	hiredate	sal	deptno
101	asha	NULL	2020-02-01	60000.00	10
102	rahul	101	2021-05-12	55000.00	20
103	meena	101	2019-03-15	70000.00	30
104	kiran	102	2022-07-10	45000.00	40
105	divya	103	2023-01-08	50000.00	50
106	ravi	101	2023-04-20	40000.00	60
NULL	NULL	NULL	NULL	NULL	NULL

PROGRAM 6: More Queries on Employee Database





Using Scheme diagram (under Program-5)

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

ii. Enter greater than five tuples for each table.

iii. List the name of the managers with the maximum employees

iv. Display those managers name whose salary is more than average salary of his employee.

v. Find the name of the second top level managers of each department.

vi. Find the employee details who got second maximum incentive in January 2019.

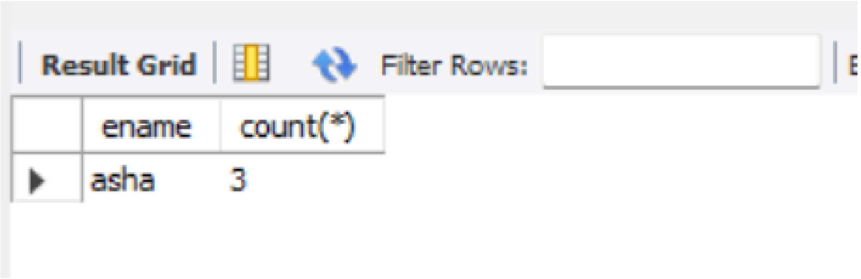
vii. Display those employees who are working in the same department where his manager is working.

1. Queries

```
use employe;
```

(i) List the name of the managers with the maximum employees

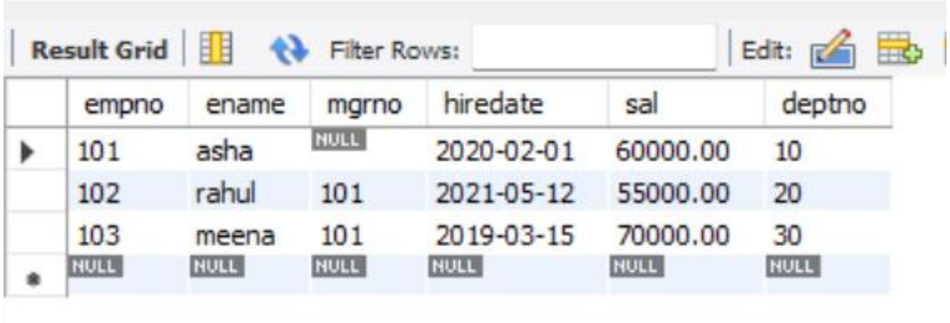
```
select m.ename,count(*)
from employee e
join employee m on e.mgrno = m.empno
group by m.empno, m.ename
having count(*) = (
    select max(cnt)
    from (
        select count(*) as cnt
        from employee
        where mgrno is not null
        group by mgrno
    ) t
);
```



```
ename   count(*)
asha    3
```

(ii) Display those managers name whose salary is more than average salary of his employee

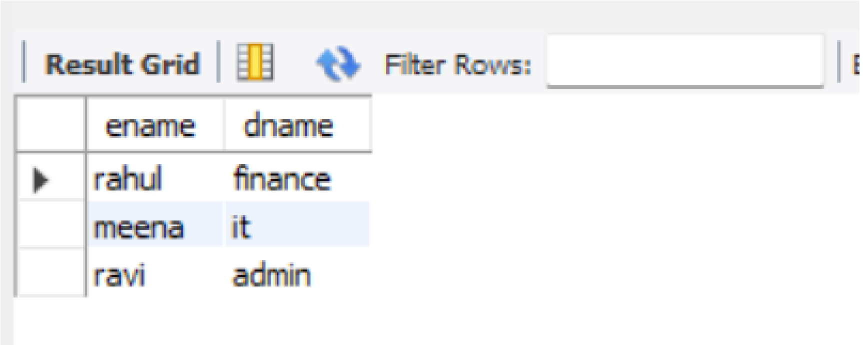
```
select distinct *
from employee m
where m.empno in (select mgrno from employee)
and m.sal > (
    select avg(e.sal)
    from employee e
    where e.mgrno = m.empno
);
```



empno	ename	mgrno	hiredate	sal	deptno
101	asha	NULL	2020-02-01	60000.00	10
102	rahul	101	2021-05-12	55000.00	20
103	meena	101	2019-03-15	70000.00	30
NULL	NULL	NULL	NULL	NULL	NULL

(iii) Find the name of the second top level managers of each department

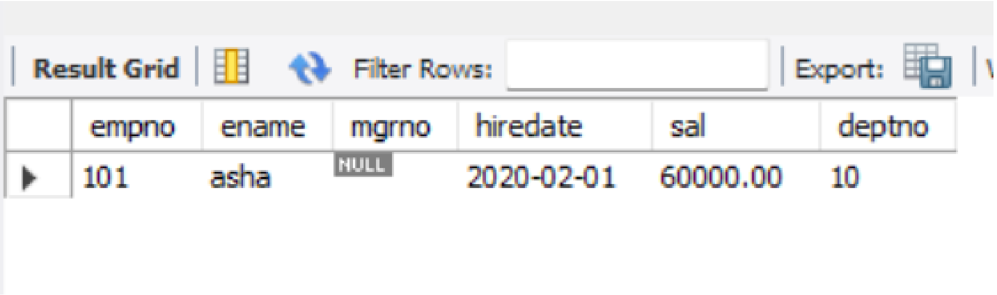
```
select distinct e.ename, d.dname
from employee e
join dept d on e.deptno = d.deptno
where e.mgrno in (
    select empno
    from employee
    where mgrno is null
);
```



ename	dname
rahul	finance
meena	it
ravi	admin

(iv) Find the employee details who got second maximum incentive in january 2019

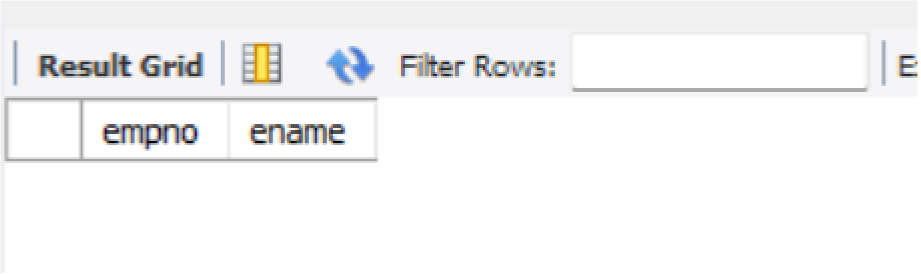
```
select e.*
from employee e
join incentives i on e.empno = i.empno
where year(i.incentivedate) = 2024
and i.incentiveamount = (
    select max(incentiveamount)
    from incentives
    where incentiveamount < (
        select max(incentiveamount) from incentives
    )
);
```



empno	ename	mgrno	hiredate	sal	deptno
101	asha	NULL	2020-02-01	60000.00	10

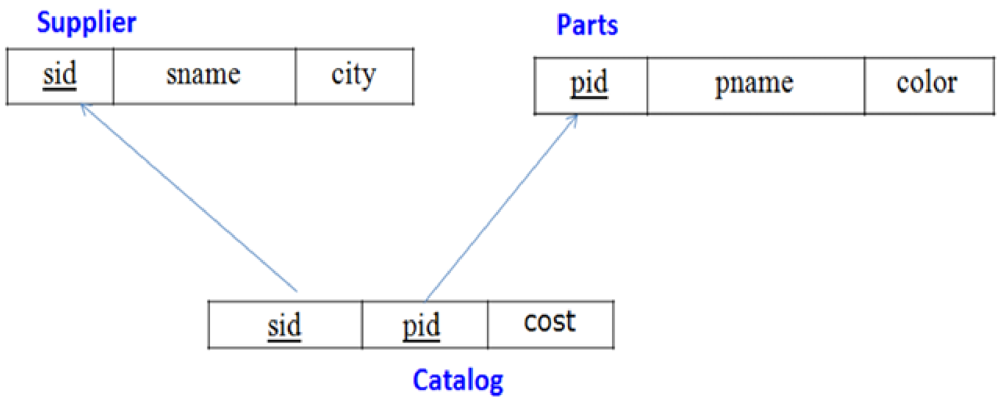
(v) **Display those employees who are working in the same department where his manager is working**

```
select e.empno, e.ename

from employee e

join employee m on e.mgrno = m.empno

where e.deptno = m.deptno;
```



empno	ename

PROGRAM 7: Supplier Database



Using Scheme diagram,

i. Create tables by properly specifying the primary keys and the foreign keys.
ii. Insert appropriate records in each table.
iii. Find the pnames of parts for which there is some supplier.
iv. Find the snames of suppliers who supply every part.
v. Find the snames of suppliers who supply every red part.
vi. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
vii. 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).
viii. For each part, find the sname of the supplier who charges the most for that part.

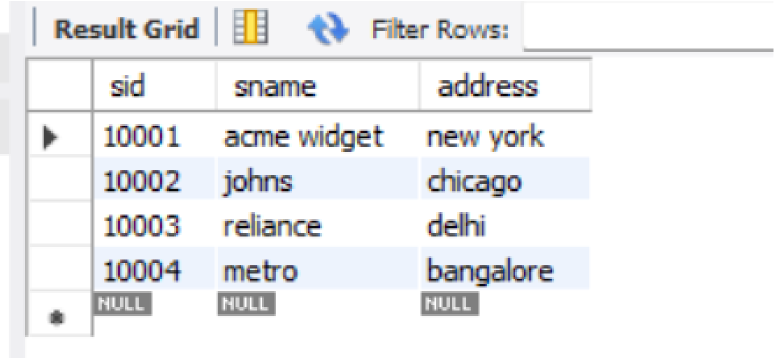
1. Create Tables

```
-- create database
create database supplierdb;
use supplierdb;

-- create tables
create table supplier (
   sid int primary key,
   sname varchar(50),
   address varchar(100)
);
```

```
          create table part (
              pid int primary key,
    pname varchar(50),
    color varchar(20)
);

create table catalog (
    sid int,
    pid int,
    cost decimal(10,2),
    primary key (sid, pid),
    foreign key (sid) references supplier(sid),
    foreign key (pid) references part(pid)
);
```

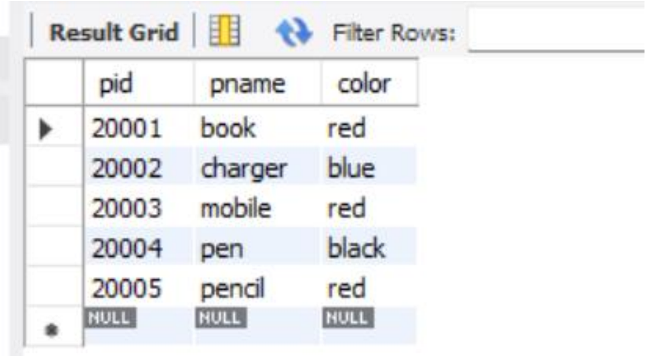
2. Insert Values

```
-- insert data into supplier
insert into supplier values
(10001, 'acme widget', 'new york'),
(10002, 'johns', 'chicago'),
(10003, 'reliance', 'delhi'),
(10004, 'metro', 'bangalore');
```



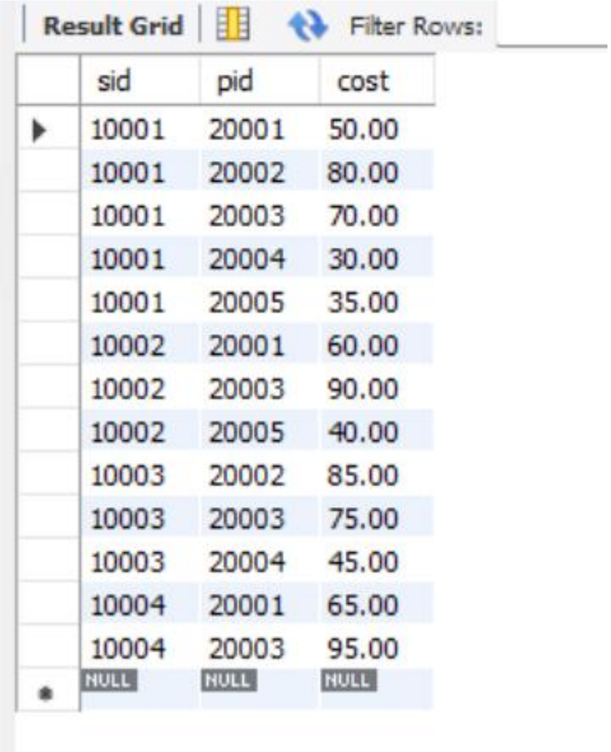
	sid	sname	address
▶	10001	acme widget	new york
	10002	johns	chicago
	10003	reliance	delhi
	10004	metro	bangalore
*	NULL	NULL	NULL

```
-- insert data into part
insert into part values
(20001, 'book', 'red'),
(20002, 'charger', 'blue'),
(20003, 'mobile', 'red'),
(20004, 'pen', 'black'),
(20005, 'pencil', 'red');
```



pid	pname	color
20001	book	red
20002	charger	blue
20003	mobile	red
20004	pen	black
20005	pencil	red
NULL	NULL	NULL

```
-- insert data into catalog
insert into catalog values
(10001, 20001, 50.00),
(10001, 20002, 80.00),
(10001, 20003, 70.00),
(10001, 20004, 30.00),
(10002, 20001, 60.00),
(10002, 20003, 90.00),
(10002, 20005, 40.00),
(10003, 20003, 75.00),
(10003, 20002, 85.00),
(10003, 20004, 45.00),
(10004, 20001, 65.00),
(10004, 20003, 95.00);
insert into catalog values (10001, 20005, 35.00);
```

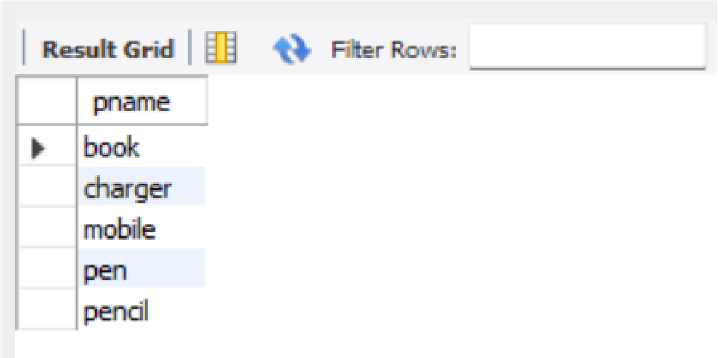


sid	pid	cost
10001	20001	50.00
10001	20002	80.00
10001	20003	70.00
10001	20004	30.00
10001	20005	35.00
10002	20001	60.00
10002	20003	90.00
10002	20005	40.00
10003	20002	85.00
10003	20003	75.00
10003	20004	45.00
10004	20001	65.00
10004	20003	95.00
NULL	NULL	NULL

3. Queries

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

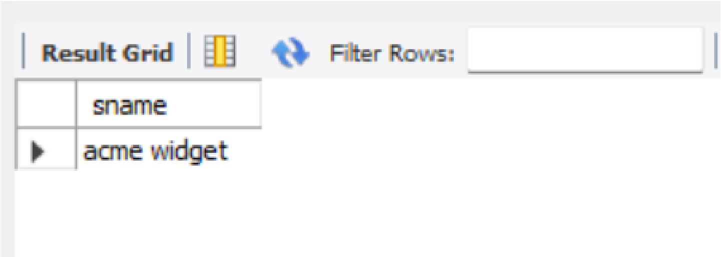
```
select distinct p.pname
from part p
join catalog c on p.pid = c.pid;
```



pname
book
charger
mobile
pen
pencil

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

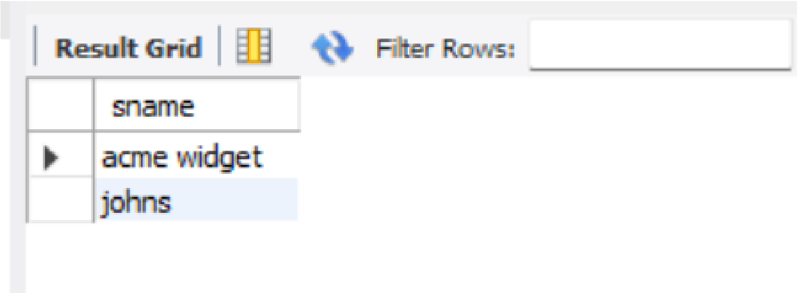
```
select s.sname
from supplier s
where not exists (
    select p.pid
    from part p
    where p.pid not in (
        select c.pid
        from catalog c
        where c.sid = s.sid
    )
);
```



sname
acme widget

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

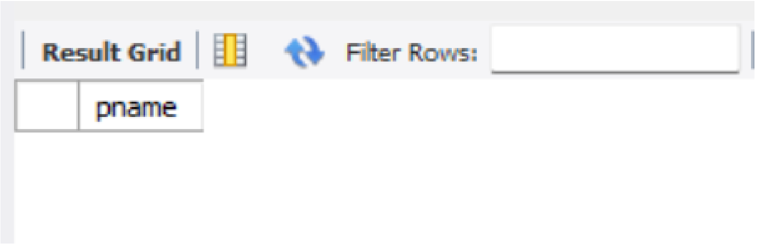
```
select s.sname
from supplier s
where not exists (
    select p.pid
    from part p
    where p.color = 'red'
    and p.pid not in (
        select c.pid
        from catalog c
        where c.sid = s.sid
    )
);
```



sname
acme widget
johns

(iv) Find the pnames of parts supplied by acme widget and by no one else

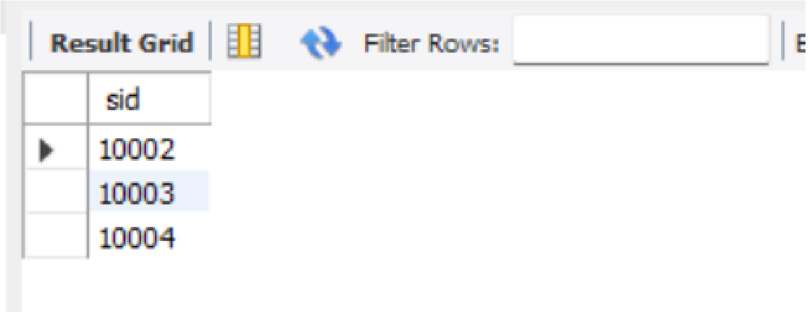
```
select p.pname
from part p
join catalog c on p.pid = c.pid
join supplier s on s.sid = c.sid
where s.sname = 'acme widget'
and p.pid not in (
    select c2.pid
    from catalog c2
    join supplier s2 on s2.sid = c2.sid
    where s2.sname <> 'acme widget'
);
```



pname

(v) Find the sids of suppliers who charge more for some part than the average cost of that part

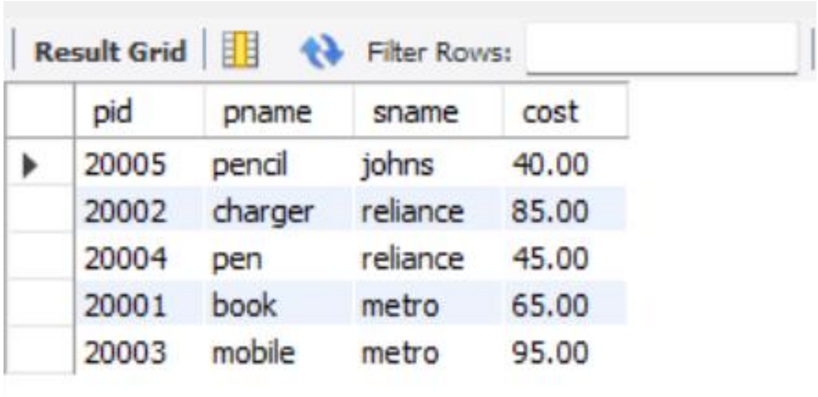
```
select distinct c.sid
from catalog c
where c.cost > (
   select avg(c2.cost)
   from catalog c2
   where c2.pid = c.pid
);
```



sid
10002
10003
10004

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

```
select p.pid, p.pname, s.sname, c.cost
from part p
join catalog c on p.pid = c.pid
join supplier s on s.sid = c.sid
where c.cost = (
   select max(c2.cost)
   from catalog c2
   where c2.pid = p.pid
);
```

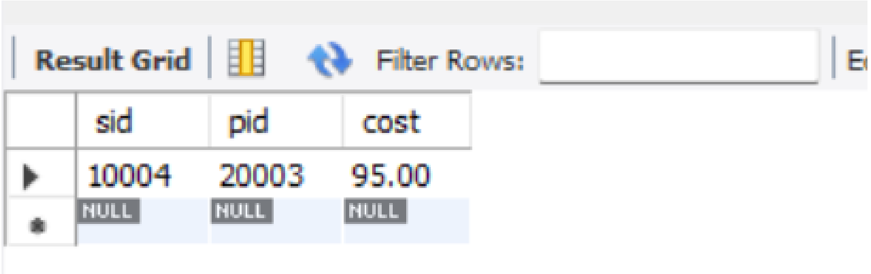


pid	pname	sname	cost
20005	pencil	johns	40.00
20002	charger	reliance	85.00
20004	pen	reliance	45.00
20001	book	metro	65.00
20003	mobile	metro	95.00

4. Queries

(i) Most expensive part and the supplier who supplies it

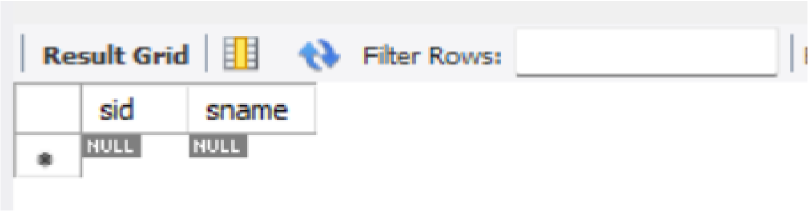
```
select c.sid, c.pid, c.cost
from catalog c
where c.cost = (select max(cost) from catalog);
```



sid	pid	cost
10004	20003	95.00
NULL	NULL	NULL

(ii) Suppliers who do NOT supply any red parts

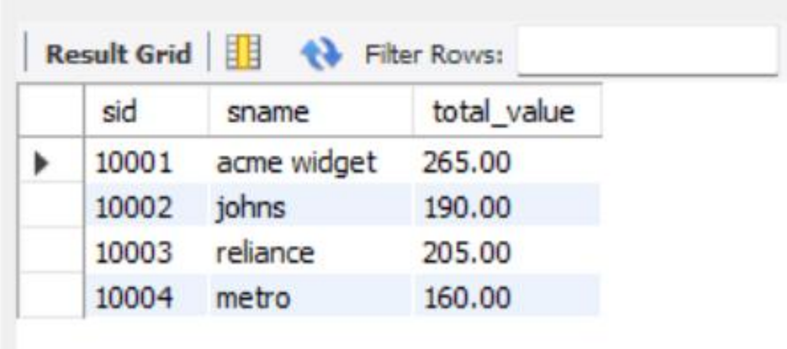
```
select s.sid, s.sname
from supplier s
where s.sid not in (
   select c.sid
   from catalog c
   join part p on c.pid = p.pid
   where p.color = 'red'
);
```



	sid	sname
*	NULL	NULL

(iii) Each supplier and total value of all parts they supply

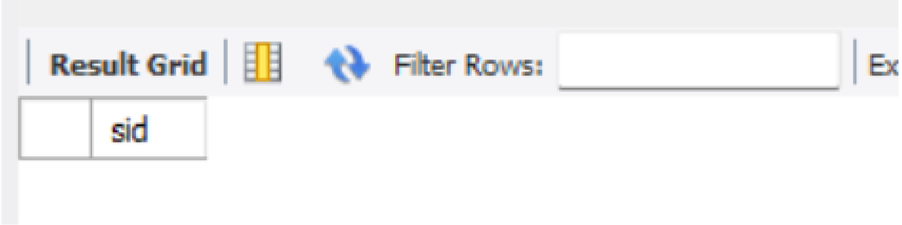
```
select s.sid, s.sname, sum(c.cost) as total_value
from supplier s
join catalog c on s.sid = c.sid
group by s.sid, s.sname;
```



	sid	sname	total_value
▶	10001	acme widget	265.00
	10002	johns	190.00
	10003	reliance	205.00
	10004	metro	160.00

(iv) Suppliers who supply at least 2 parts cheaper than ₹20

```
select c.sid
from catalog c
where c.cost < 20
group by c.sid
having count(*) >= 2;
```

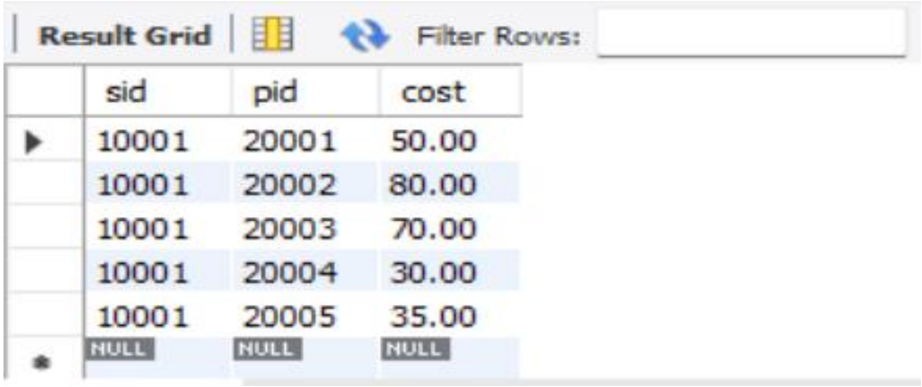


	sid

(v) Suppliers who offer the cheapest cost for each part

```
select c.sid, c.pid, c.cost
from catalog c
where c.cost = (
   select min(cost)
```

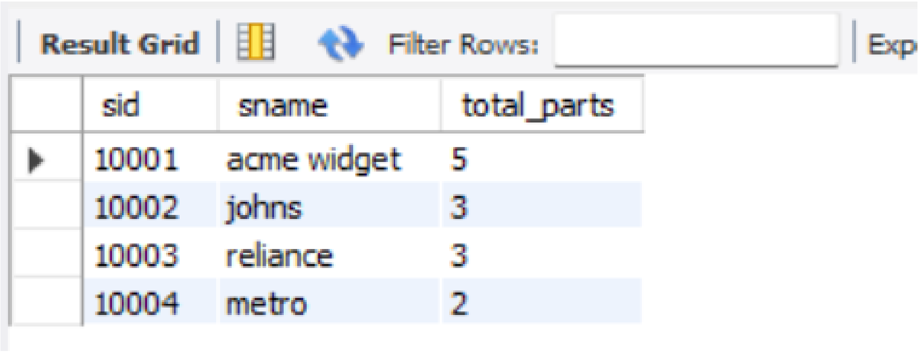
```
    from catalog
    where pid = c.pid
);
```



sid	pid	cost
10001	20001	50.00
10001	20002	80.00
10001	20003	70.00
10001	20004	30.00
10001	20005	35.00
NULL	NULL	NULL

(vi) View: suppliers and total number of parts they supply

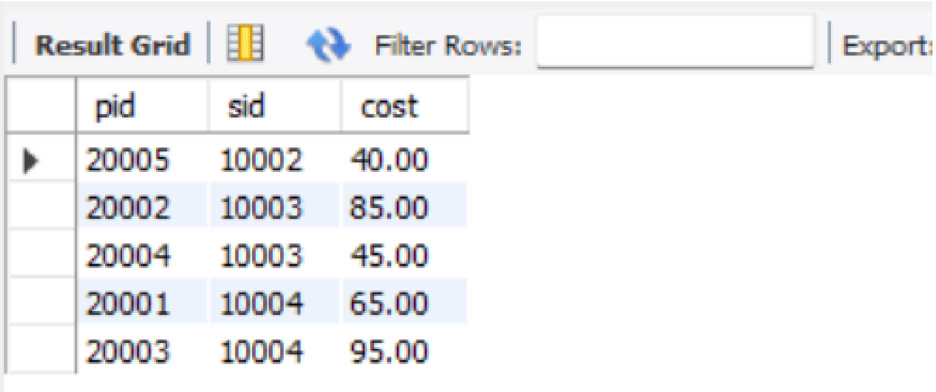
```
create view supplier_parts_count as
select s.sid, s.sname, count(c.pid) as total_parts
from supplier s
left join catalog c on s.sid = c.sid
group by s.sid, s.sname;
```



sid	sname	total_parts
10001	acme widget	5
10002	johns	3
10003	reliance	3
10004	metro	2

(vii) View: most expensive supplier for each part

```
create view most_expensive_supplier as
select c.pid, c.sid, c.cost
from catalog c
where c.cost = (
    select max(cost)
    from catalog
    where pid = c.pid
);
```



pid	sid	cost
20005	10002	40.00
20002	10003	85.00
20004	10003	45.00
20001	10004	65.00
20003	10004	95.00

(viii) Trigger prevent inserting catalog cost below 1

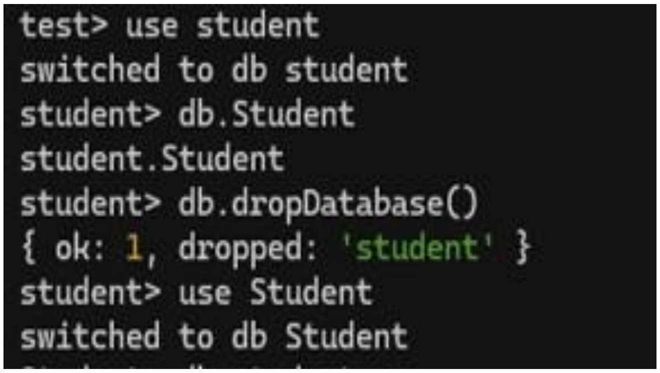
```
delimiter //
create trigger check_cost_before_insert
before insert on catalog
for each row
begin
   if new.cost < 1 then
      signal sqlstate '45000'
      set message_text = 'cost cannot be less than 1';
   end if;
end;
//
delimiter ;
```

(ix) Trigger set default cost if not provided (default = 50)

```
delimiter //
create trigger set_default_cost
before insert on catalog
for each row
begin
   if new.cost is null then
      set new.cost = 50;
   end if;
end;
//
delimiter ;
```

Program 8: NoSQL Student Database

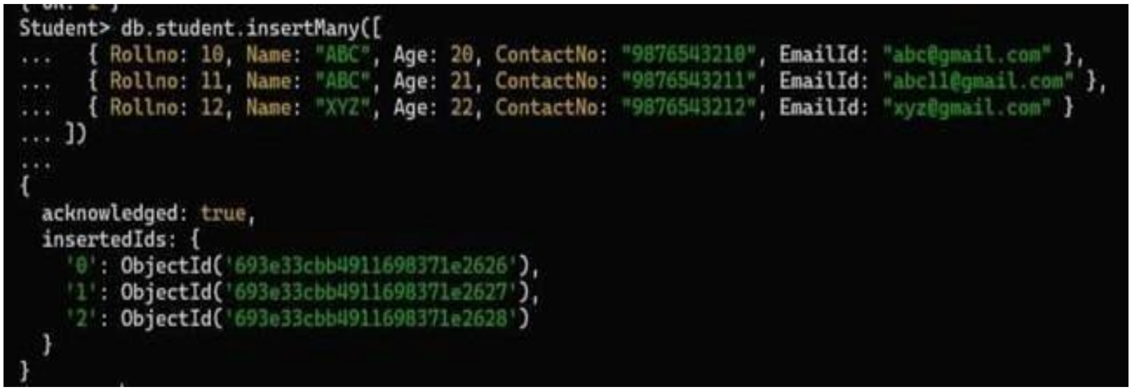
i. **Create a database “Student” with the following attributes Rollno, Age, ContactNo, EmailId.**



```
test> use student
switched to db student
student> db.Student
student.Student
student> db.dropDatabase()
{ ok: 1, dropped: 'student' }
student> use Student
switched to db Student
```

ii. **Insert appropriate values**

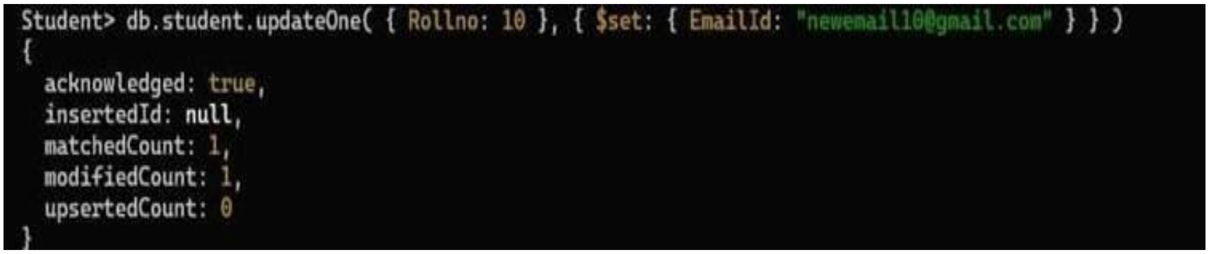
db.student.insertMany ([{ Rollno: 10, Name: "ABC", Age: 20, ContactNo: "9876543210", EmailId: "abc@gmail.com" },{ Rollno: 11, Name: "ABC", Age: 21, ContactNo: "9876543211", EmailId: "abc11@gmail.com" }, { Rollno: 12, Name: "XYZ", Age: 22, ContactNo: "9876543212", EmailId: "xyz@gmail.com" }])



```
Student> db.student.insertMany([
...   { Rollno: 10, Name: "ABC", Age: 20, ContactNo: "9876543210", EmailId: "abc@gmail.com" },
...   { Rollno: 11, Name: "ABC", Age: 21, ContactNo: "9876543211", EmailId: "abc11@gmail.com" },
...   { Rollno: 12, Name: "XYZ", Age: 22, ContactNo: "9876543212", EmailId: "xyz@gmail.com" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693e33cbb4911698371e2626'),
    '1': ObjectId('693e33cbb4911698371e2627'),
    '2': ObjectId('693e33cbb4911698371e2628')
  }
}
```

iii. **Write query to update Email-Id of a student with rollno 10.**

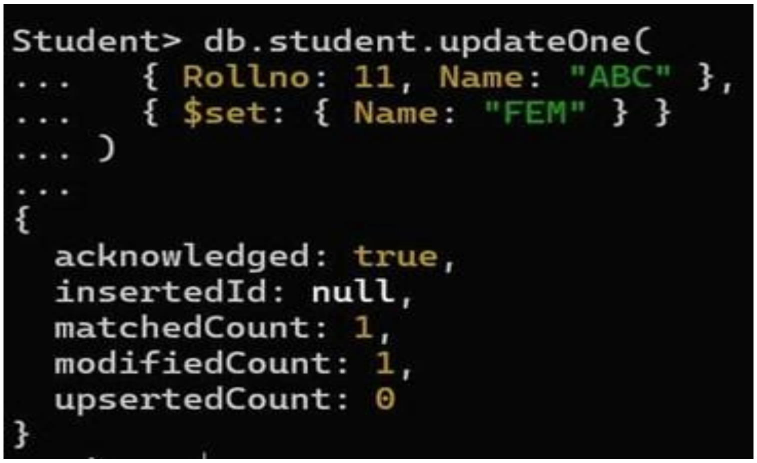
db.student.updateOne({ Rollno: 10 }, { $set: { EmailId: "newemail10@gmail.com" } })



```
Student> db.student.updateOne( { Rollno: 10 }, { $set: { EmailId: "newemail10@gmail.com" } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

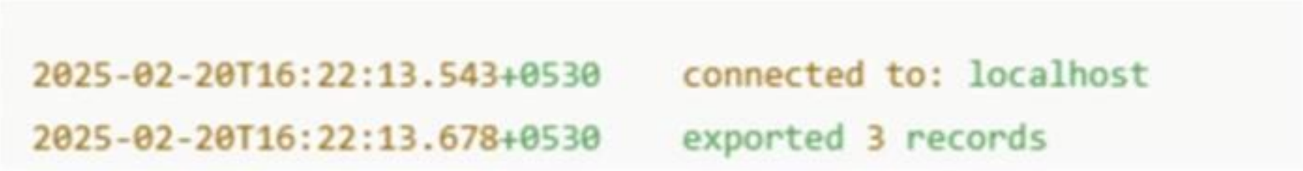
iv. Replace the student name from “ABC” to “FEM” of rollno 11.

db.student.updateOne({ Rollno: 11, Name: "ABC" }, { $set: { Name: "FEM" } }



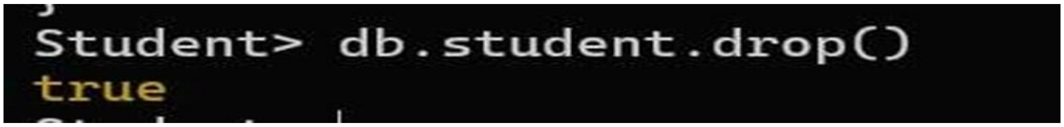
```
Student> db.student.updateOne(
...    { Rollno: 11, Name: "ABC" },
...    { $set: { Name: "FEM" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

v. Export the created table into local file system



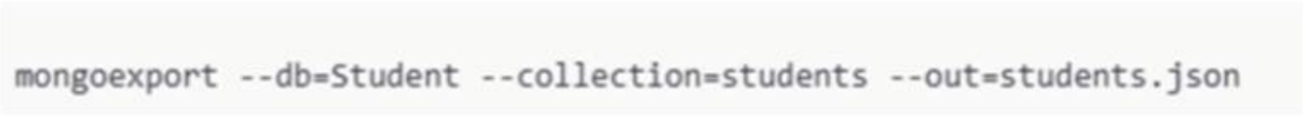
```
2025-02-20T16:22:13.543+0530    connected to: localhost
2025-02-20T16:22:13.678+0530    exported 3 records
```

vi. Drop the table.



```
Student> db.student.drop()
true
```

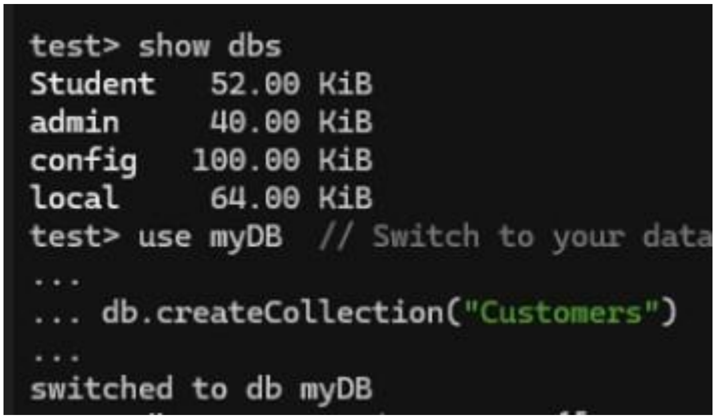
vii. Import a given csv dataset from local file system into mongodb collection.



```
mongoexport --db=Student --collection=students --out=students.json
```

Program 9: NoSQL Customer Database

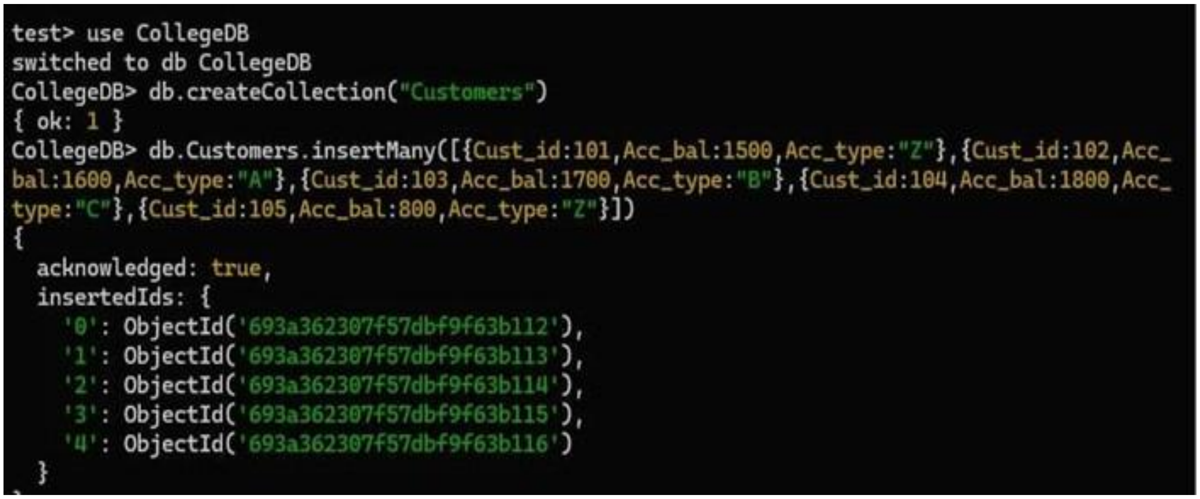
i. Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type



```
test> show dbs
Student   52.00 KiB
admin     40.00 KiB
config   100.00 KiB
local     64.00 KiB
test> use myDB  // Switch to your data
...
... db.createCollection("Customers")
...
switched to db myDB
```

ii. Insert at least 5 values into the table.

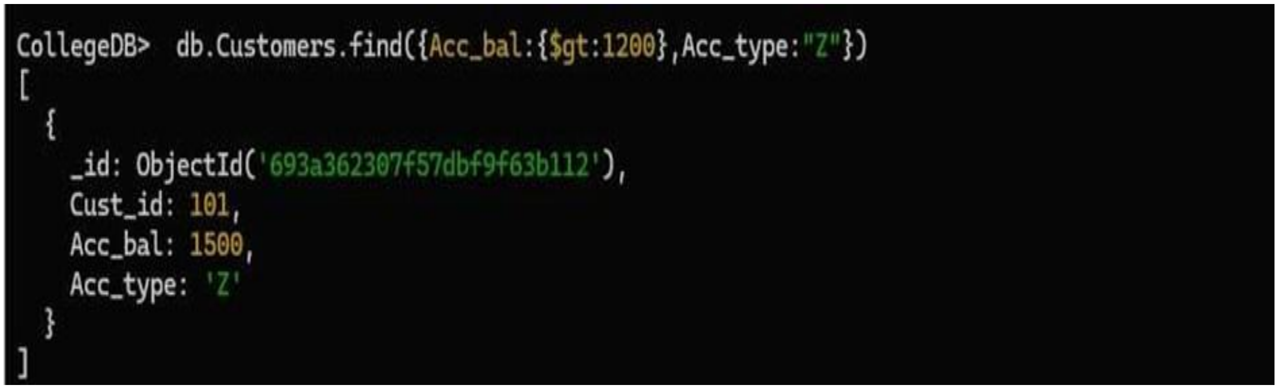
```
db.Customers.insertMany([
  { Cust_id: 101, Acc_Bal: 1500,  Acc_Type: "Z" },
  { Cust_id: 101, Acc_Bal: 1600,  Acc_Type: "A" },
  { Cust_id: 102, Acc_Bal: 1700, Acc_Type: "B" },
  { Cust_id: 103, Acc_Bal: 1800,  Acc_Type: "C" },
  { Cust_id: 103, Acc_Bal: 800,  Acc_Type: "Z" }])
```



```
test> use CollegeDB
switched to db CollegeDB
CollegeDB> db.createCollection("Customers")
{ ok: 1 }
CollegeDB> db.Customers.insertMany([{Cust_id:101,Acc_bal:1500,Acc_type:"Z"},{Cust_id:102,Acc_
bal:1600,Acc_type:"A"},{Cust_id:103,Acc_bal:1700,Acc_type:"B"},{Cust_id:104,Acc_bal:1800,Acc_
type:"C"},{Cust_id:105,Acc_bal:800,Acc_type:"Z"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693a362307f57dbf9f63b112'),
    '1': ObjectId('693a362307f57dbf9f63b113'),
    '2': ObjectId('693a362307f57dbf9f63b114'),
    '3': ObjectId('693a362307f57dbf9f63b115'),
    '4': ObjectId('693a362307f57dbf9f63b116')
  }
```

iii. Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.

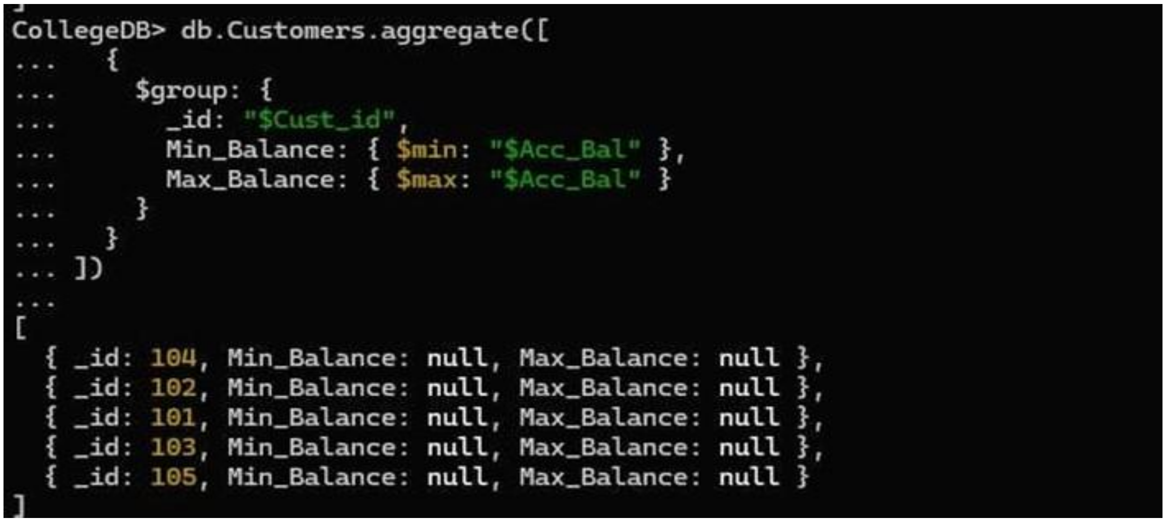
```
db.Customers.aggregate([
  { $match: { Acc_type: "Z" } },
  { $group: {   _id: "$Cust_id", TotalBalance: { $sum:
"$Acc_bal" }}},   { $match: { TotalBalance: { $gt:
1200 } } }])
```



```
CollegeDB>  db.Customers.find({Acc_bal:{$gt:1200},Acc_type:"Z"})
[
  {
    _id: ObjectId('693a362307f57dbf9f63b112'),
    Cust_id: 101,
    Acc_bal: 1500,
    Acc_type: 'Z'
  }
]
```

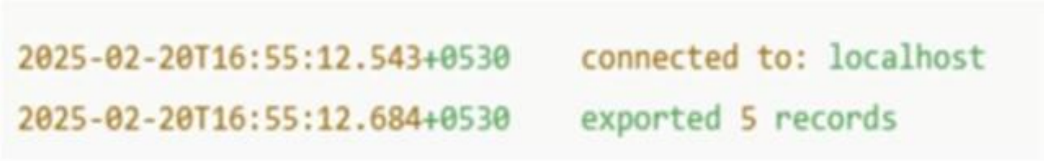
iv. Determine Minimum and Maximum account balance for each customer_id.

```
db.Customers.aggregate([{$group: { _id: "$Cust_id", MinBalance: { $min: "$Acc_Bal" },MaxBalance: { $max: "$Acc_Bal" }}}])
```



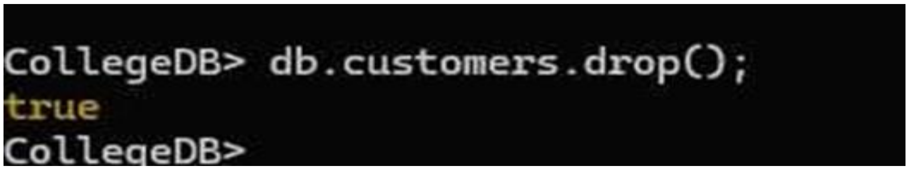
```
CollegeDB> db.Customers.aggregate([
...   {
...     $group: {
...       _id: "$Cust_id",
...       Min_Balance: { $min: "$Acc_Bal" },
...       Max_Balance: { $max: "$Acc_Bal" }
...     }
...   }
... ])
...
[
  { _id: 104, Min_Balance: null, Max_Balance: null },
  { _id: 102, Min_Balance: null, Max_Balance: null },
  { _id: 101, Min_Balance: null, Max_Balance: null },
  { _id: 103, Min_Balance: null, Max_Balance: null },
  { _id: 105, Min_Balance: null, Max_Balance: null }
]
```

v. Export the created collection into local file system.



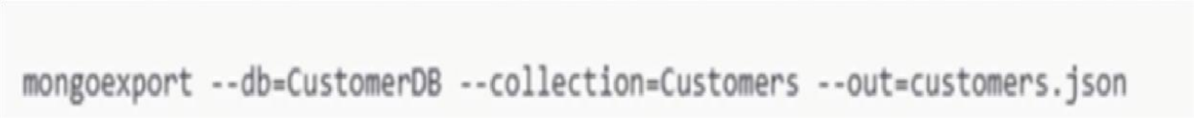
```
2025-02-20T16:55:12.543+0530    connected to: localhost
2025-02-20T16:55:12.684+0530    exported 5 records
```

vi. **Drop the table.** db.Customers.drop()



```
CollegeDB> db.customers.drop();
true
CollegeDB>
```

vii. **Import a given csv dataset from local file system into mongodb collection.**

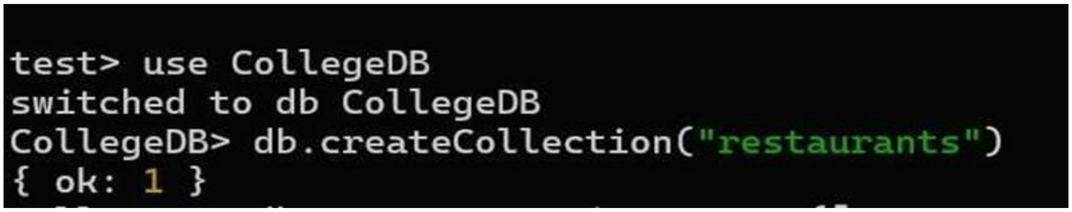


```
mongoexport --db=CustomerDB --collection=Customers --out=customers.json
```

Program 10: NoSQL Restaurant Database

i. Write NoSQL Queries on "Restaurant" collection.

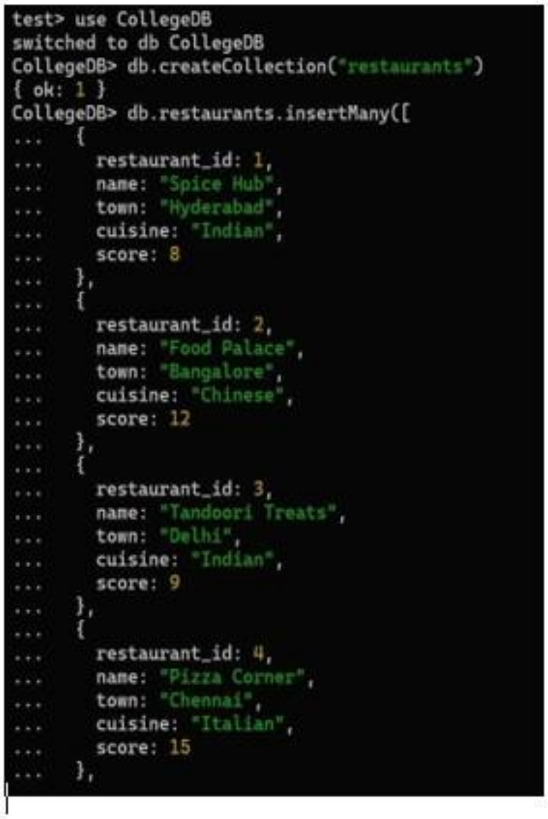
db.createCollection("restaurants")



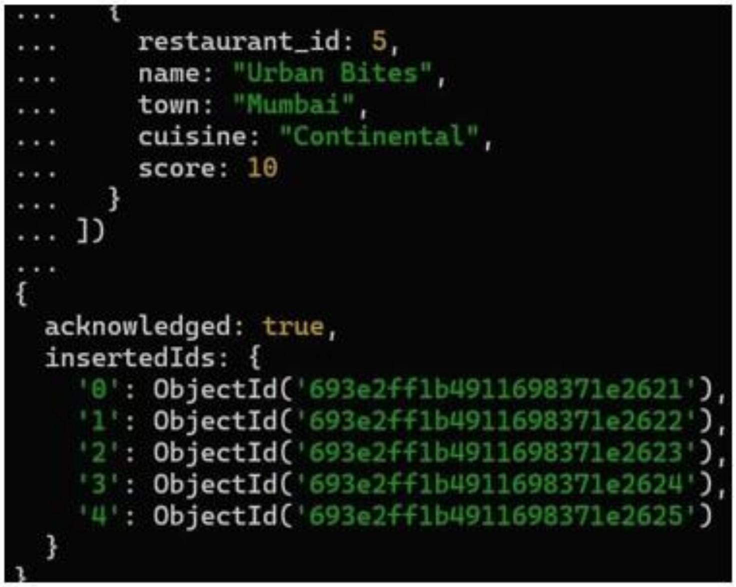
```
test> use CollegeDB
switched to db CollegeDB
CollegeDB> db.createCollection("restaurants")
{ ok: 1 }
```

ii. Write a MongoDB query to display all the documents in the collection restaurants?

db.restaurants.insertMany([{restaurant_id: 1,name: "Spice Hub",town: "Hyderabad",cuisine: "Indian",score: 8},{restaurant_id: 2,name: "Food Palace",town: "Bangalore"cuisine: "Chinese",score: 12 },{restaurant_id: 3,name: "Tandoori Treats",town: "Delhi",cuisine: "Indian",score: 9 },{restaurant_id: 4,name: "Pizza Corner",town: "Chennai",cuisine: "Italian",score: 15},{ restaurant_id: 5,name: "Urban Bites",town: "Mumbai",cuisine: "Continental", score: 10}])



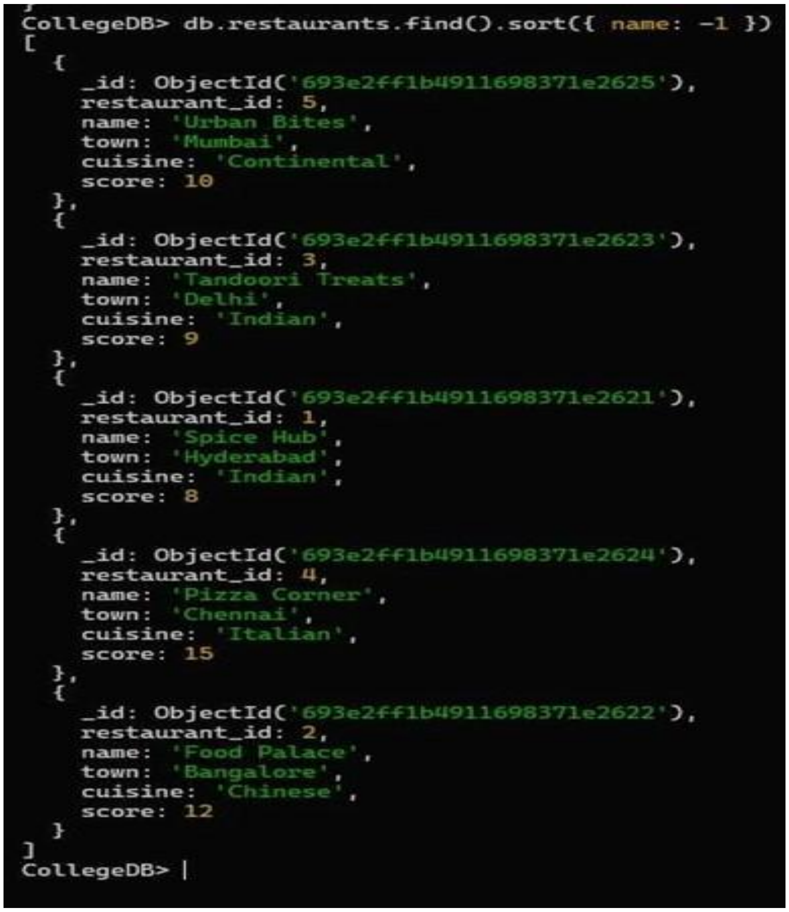
```
test> use CollegeDB
switched to db CollegeDB
CollegeDB> db.createCollection("restaurants")
{ ok: 1 }
CollegeDB> db.restaurants.insertMany([
...   {
...     restaurant_id: 1,
...     name: "Spice Hub",
...     town: "Hyderabad",
...     cuisine: "Indian",
...     score: 8
...   },
...   {
...     restaurant_id: 2,
...     name: "Food Palace",
...     town: "Bangalore",
...     cuisine: "Chinese",
...     score: 12
...   },
...   {
...     restaurant_id: 3,
...     name: "Tandoori Treats",
...     town: "Delhi",
...     cuisine: "Indian",
...     score: 9
...   },
...   {
...     restaurant_id: 4,
...     name: "Pizza Corner",
...     town: "Chennai",
...     cuisine: "Italian",
...     score: 15
...   },
```



```
...   {
...     restaurant_id: 5,
...     name: "Urban Bites",
...     town: "Mumbai",
...     cuisine: "Continental",
...     score: 10
...   }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693e2ff1b4911698371e2621'),
    '1': ObjectId('693e2ff1b4911698371e2622'),
    '2': ObjectId('693e2ff1b4911698371e2623'),
    '3': ObjectId('693e2ff1b4911698371e2624'),
    '4': ObjectId('693e2ff1b4911698371e2625')
  }
}
```

iii. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns

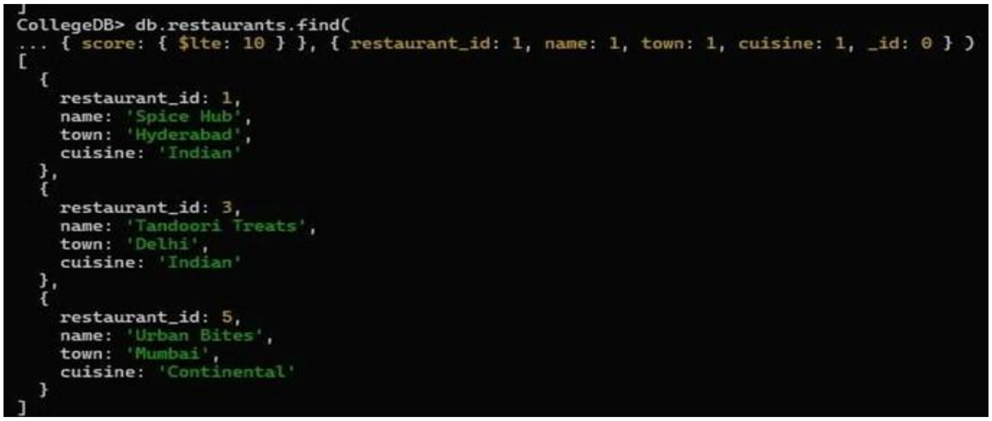
. db.restaurants.find().sort({ name: -1 })



```
}
CollegeDB> db.restaurants.find().sort({ name: -1 })
[
  {
    _id: ObjectId('693e2ff1b4911698371e2625'),
    restaurant_id: 5,
    name: 'Urban Bites',
    town: 'Mumbai',
    cuisine: 'Continental',
    score: 10
  },
  {
    _id: ObjectId('693e2ff1b4911698371e2623'),
    restaurant_id: 3,
    name: 'Tandoori Treats',
    town: 'Delhi',
    cuisine: 'Indian',
    score: 9
  },
  {
    _id: ObjectId('693e2ff1b4911698371e2621'),
    restaurant_id: 1,
    name: 'Spice Hub',
    town: 'Hyderabad',
    cuisine: 'Indian',
    score: 8
  },
  {
    _id: ObjectId('693e2ff1b4911698371e2624'),
    restaurant_id: 4,
    name: 'Pizza Corner',
    town: 'Chennai',
    cuisine: 'Italian',
    score: 15
  },
  {
    _id: ObjectId('693e2ff1b4911698371e2622'),
    restaurant_id: 2,
    name: 'Food Palace',
    town: 'Bangalore',
    cuisine: 'Chinese',
    score: 12
  }
]
CollegeDB> |
```

iv. Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10.

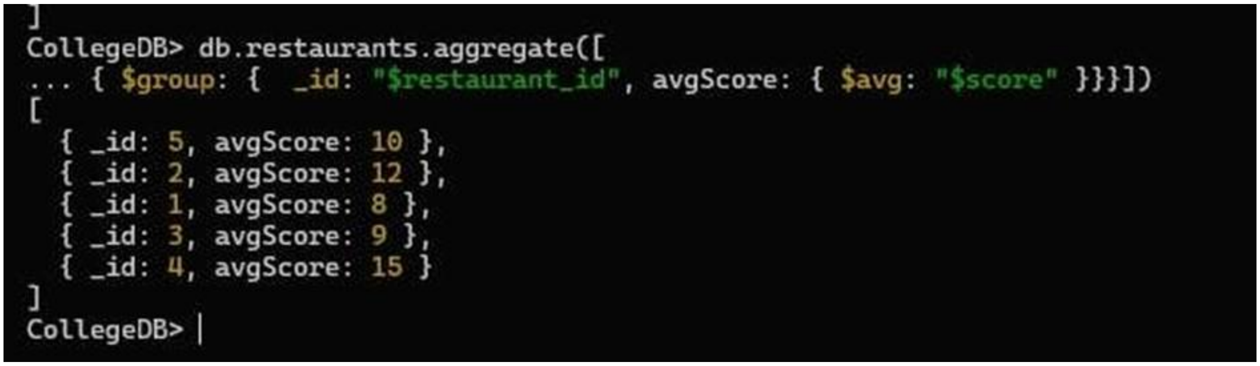
db.restaurants.find({ score: { $lte: 10 } }, { restaurant_id: 1, name: 1, town: 1, cuisine: 1, _id: 0 })



```
]
CollegeDB> db.restaurants.find(
... { score: { $lte: 10 } }, { restaurant_id: 1, name: 1, town: 1, cuisine: 1, _id: 0 } )
[
  {
    restaurant_id: 1,
    name: 'Spice Hub',
    town: 'Hyderabad',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 3,
    name: 'Tandoori Treats',
    town: 'Delhi',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 5,
    name: 'Urban Bites',
    town: 'Mumbai',
    cuisine: 'Continental'
  }
]
```

v. Write a MongoDB query to find the average score for each restaurant.

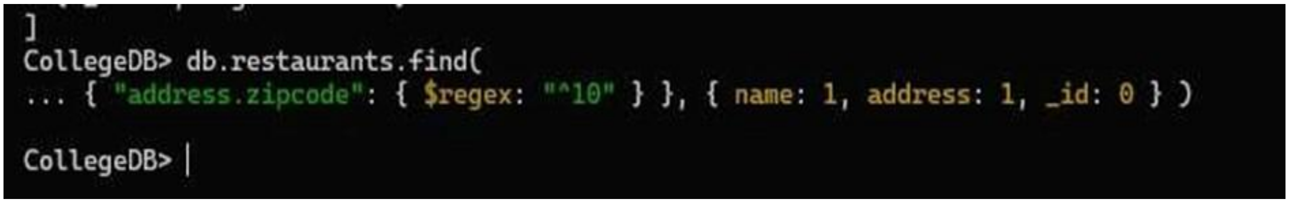
db.restaurants.aggregate([{ $group: { _id: "$restaurant_id", avgScore: { $avg: "$score" }}}])



```
]
CollegeDB> db.restaurants.aggregate([
... { $group: {  _id: "$restaurant_id", avgScore: { $avg: "$score" }}}])
[
  { _id: 5, avgScore: 10 },
  { _id: 2, avgScore: 12 },
  { _id: 1, avgScore: 8 },
  { _id: 3, avgScore: 9 },
  { _id: 4, avgScore: 15 }
]
CollegeDB> |
```

vi. Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

db.restaurants.find({ "address.zipcode": { $regex: "^10" } }, { name: 1, address: 1, _id: 0 })



```
]
CollegeDB> db.restaurants.find(
... { "address.zipcode": { $regex: "^10" } }, { name: 1, address: 1, _id: 0 } )

CollegeDB> |
```

*****