VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka

LAB REPORT
On

Database Management Systems (23CS3PCDBM)

Submitted by

Prajwal Vasantha Kumar(1BM24CS210)

in partial fulfillment 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-2025 to Jan-2026

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 **Prajwal Vasantha Kumar (1BM24CS210),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment 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.

Prof.Surabhi S Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

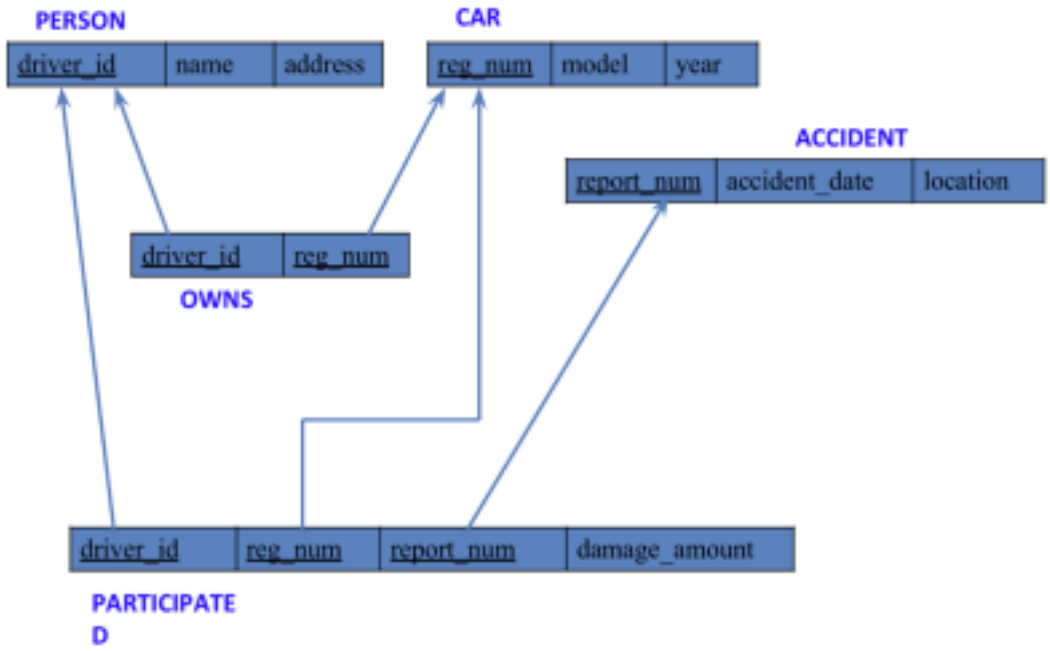
Insurance Database

Question

(Week 1)

- 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. - Enter at least five tuples for each relation
- Display Accident date and location
- 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.
- Add a new accident to the database.
- To Do
- Display Accident date and location
- Display driver id who did accident with damage amount greater than or equal to Rs.25000

Schema Diagram



Create database

```
create database insurance_dhiksha;

use insurance_dhiksha;
```

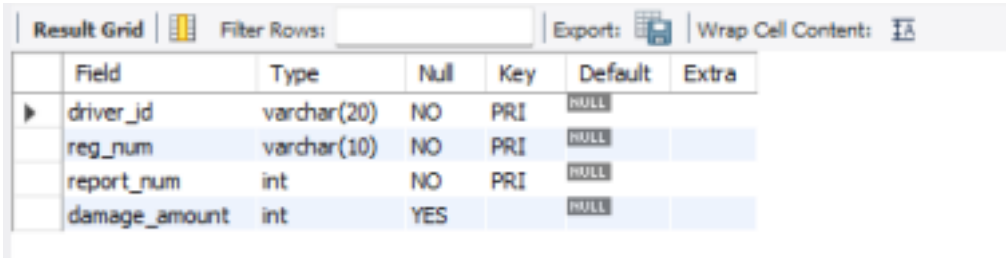
Create table

```
create table insurance_dhiksha.person(

driver_id varchar(20),

name varchar(30),

address varchar(50),

PRIMARY KEY(driver_id)

);

create table insurance_dhiksha.car(

reg_num varchar(15),

model varchar(10),
year int,

PRIMARY KEY(reg_num)

);
```

```
create table insurance_dhiksha.owns(
driver_id varchar(20),
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 insurance_dhiksha.accident(
report_num int,
accident_date date,
location varchar(50),
PRIMARY KEY(report_num)
);
create table insurance_dhiksha.participated(
driver_id varchar(20),
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)
);
```

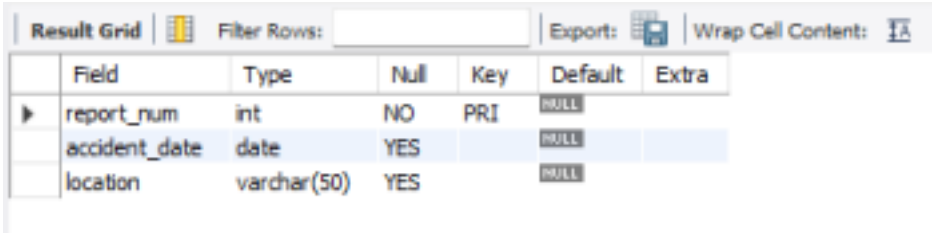
Structure of the table

desc person;



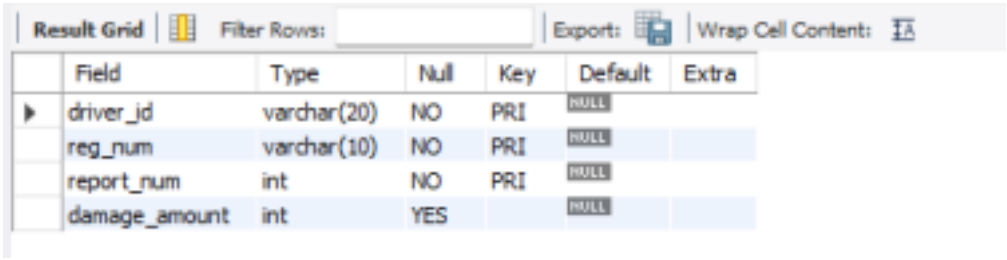
Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

desc accident;



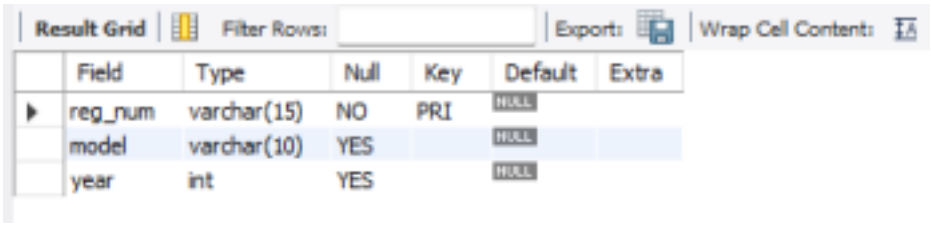
Field	Type	Null	Key	Default	Extra
report_num	int	NO	PRI	NULL	
accident_date	date	YES		NULL	
location	varchar(50)	YES		NULL	

desc participated;



Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

desc car;



Field	Type	Null	Key	Default	Extra
reg_num	varchar(15)	NO	PRI	NULL	
model	varchar(10)	YES		NULL	
year	int	YES		NULL	

desc owns;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	

Inserting Values to the table

insert into person values("A01","Richard", "Srinivas nagar");

insert into person values("A02","Pradeep", "Rajaji nagar");

insert into person values("A03","Smith", "Ashok nagar");

insert into person values("A04","Venu", "N R Colony");

insert into person values("A05","John", "Hanumanth nagar");

select * from person;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

person 19

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");

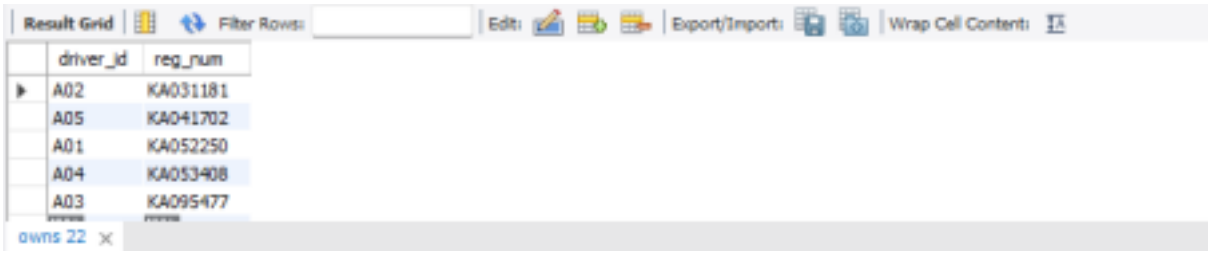
select * from car;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

car 20

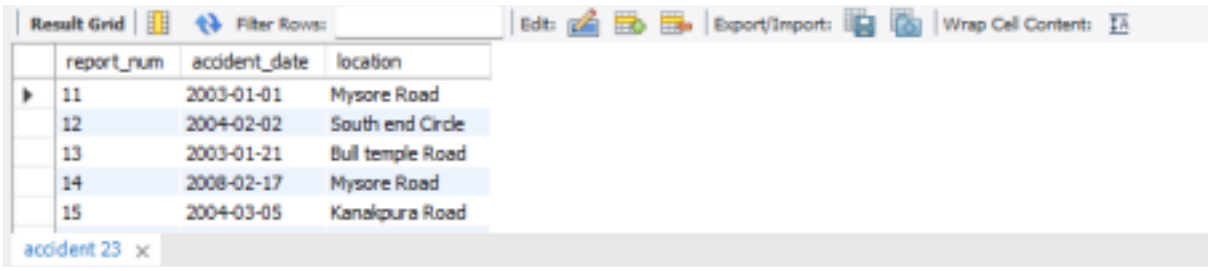
```
insert into owns values("A01","KA052250");
insert into owns values("A02","KA031181");
insert into owns values("A03","KA095477");
insert into owns values("A04","KA053408");
insert into owns values("A05","KA041702");
select * from owns;
```



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

owns 22

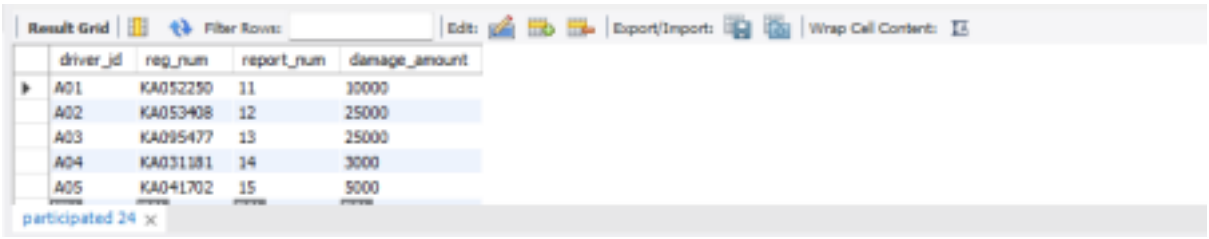
```
insert into accident values(11,'2003-01-01',"Mysore Road");
insert into accident values(12,'2004-02-02',"South end Circle");
insert into accident values(13,'2003-01-21',"Bull temple Road");
insert into accident values(14,'2008-02-17',"Mysore Road");
insert into accident values(15,'2004-03-05',"Kanakpura Road");
select * from accident;
```



report_num	accident_date	location
11	2003-01-01	Mysore Road
12	2004-02-02	South end Circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore Road
15	2004-03-05	Kanakpura Road

accident 23

```
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);
select * from participated;
```

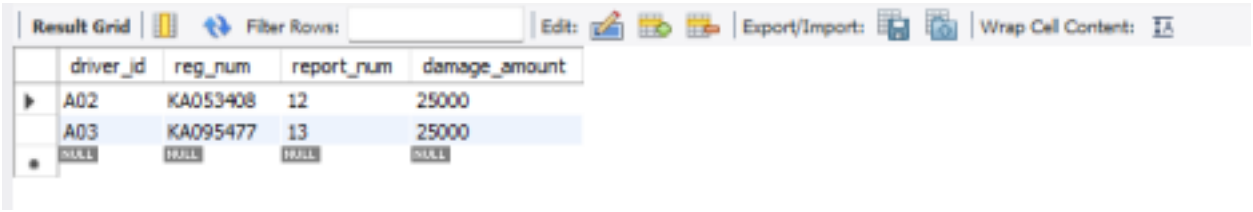


driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A02	KA053408	12	25000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000

Queries

- Update the damage amount to 25000 for the car with a specific reg-num (example 'KA053408') for which the accident report number was 12.

```
update participated
set damage_amount=25000
where reg_num='KA053408' and report_num=12;
```



driver_id	reg_num	report_num	damage_amount
A02	KA053408	12	25000
A03	KA095477	13	25000
NULL	NULL	NULL	NULL

- **Find the total number of people who owned cars that were involved in accidents in 2008.**

```
select count(distinct driver_id) CNT
from participated a, accident b
where a.report_num=b.report_num and b.accident_date like '2008%';
```

- **Add a new accident to the database.**

```
insert into accident values(16,'2008-03-08',"Domlur"); select * from accident;
```

More Queries on Insurance Database

Question
(Week 2)

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 previous week's lab and Enter at least five tuples for each relationEnter at least five tuples for each relation

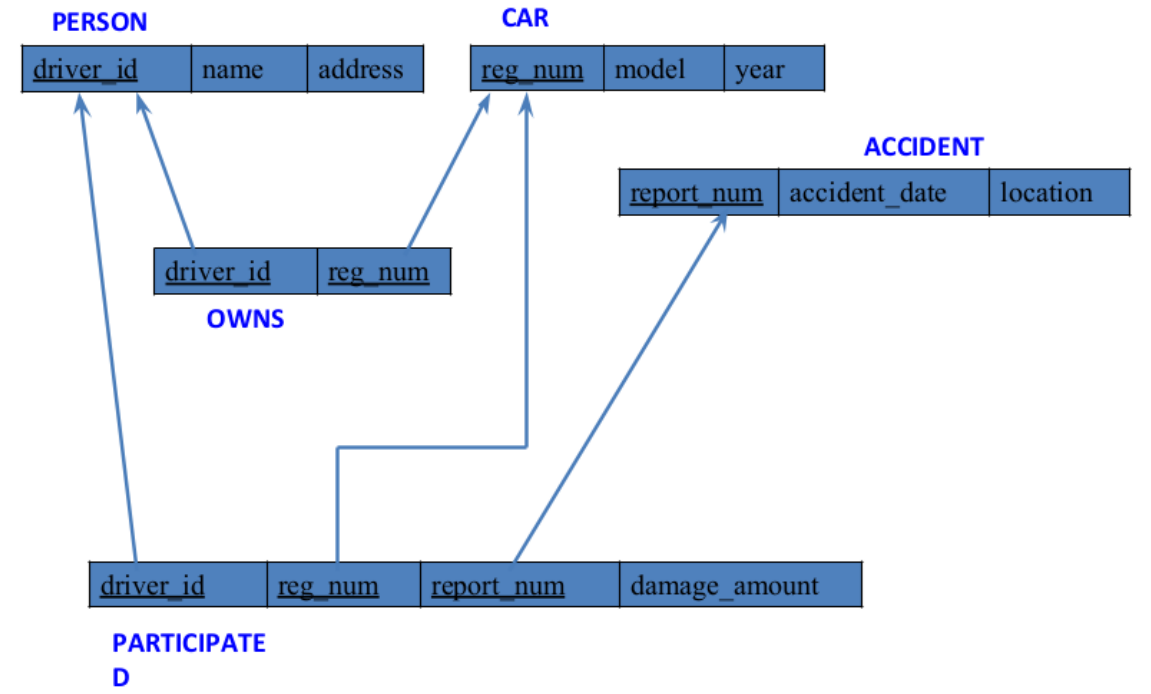
Enter at least five tuples for each relation

Display the entire CAR relation in the ascending order of manufacturing year.

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

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

Schema diagram



Queries

- **Display the entire CAR relation in the ascending order of manufacturing year.**

select * from car order by year asc;

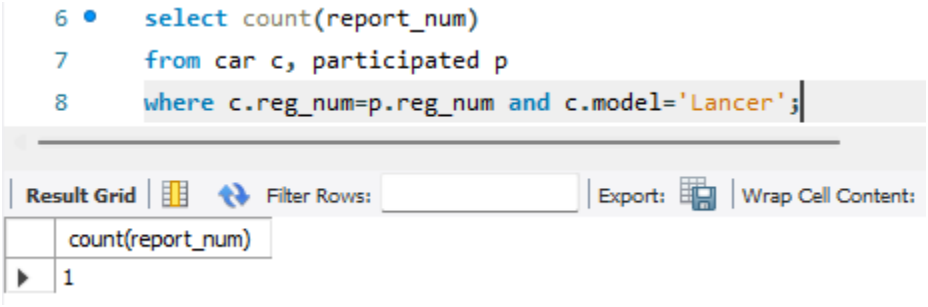
```
5 •  select * from car order by year asc;
```

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

reg_num	model	year
KA031181	LANCER	1957
KA052250	INDICA	1990
KA095477	TOYOTA	1998
KA041702	AUDI	2005
KA053408	HONDA	2008
NULL	NULL	NULL

- **Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.**

```
select count(report_num)
from car c, participated p
where c.reg_num=p.reg_num and c.model='Lancer';
```



Bank Database

Question
(Week 3)

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)

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

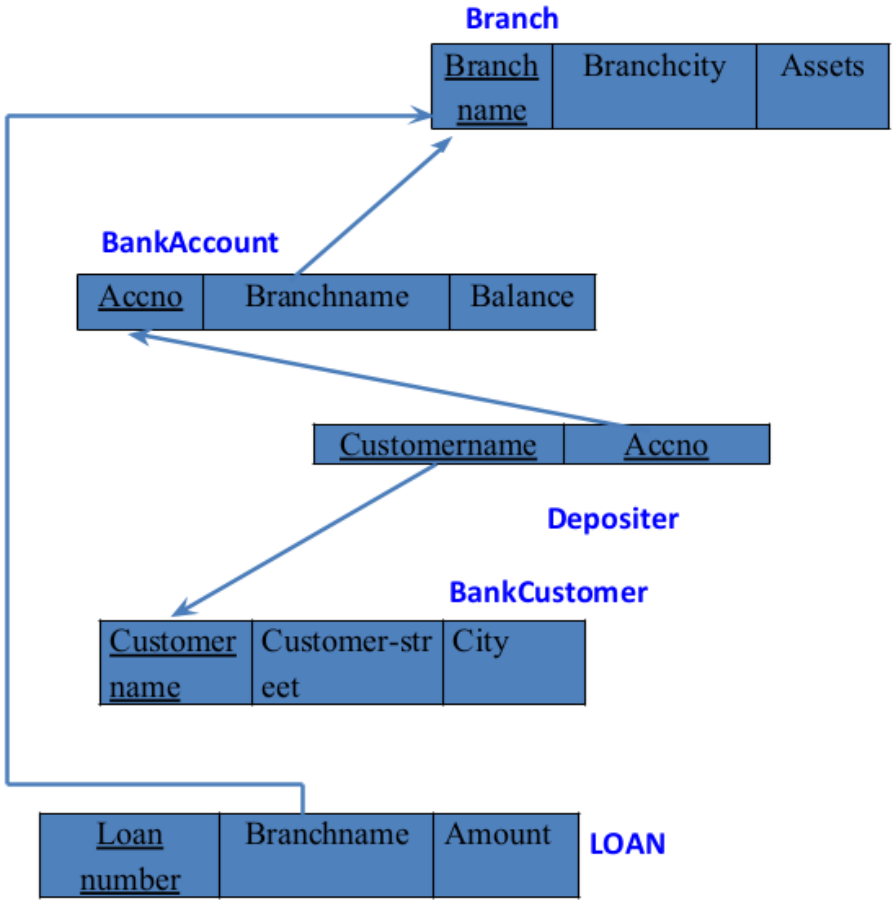
Enter at least five tuples for each relation.

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

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

CREATE A VIEW WHICH GIVES EACH BRANCH THE SUM OF THE AMOUNT OF ALL THE LOANS AT THE BRANCH.

Schema Diagram



Create database

create database if not exists bank;
use bank;

Create table

create table BRANCH(Branch_name varchar(20), branch_city varchar(20), Assets real, PRIMARY KEY(Branch_name));

create table BANKACCOUNT(accno int, Branch_name varchar(20), balance real, primary key(accno,Branch_name), foreign key(Branch_name) references BRANCH(Branch_name));

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

create table DEPOSITER(customer_name varchar(20), accno int, primary key(customer_name,accno), foreign key(accno) references BANKACCOUNT(accno),foreign key (customer_name) references BANKCUSTOMER(customer_name));

CREATE TABLE LOAN(LOAN_number int, Branch_name varchar(20), ammount real, primary key(Branch_name), foreign key(Branch_name) references BRANCH(Branch_name));

Structure of the table

desc branch;

```
10 •  desc branch;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
Branch_name	varchar(20)	NO	PRI	NULL	
branch_city	varchar(20)	YES		NULL	
Assets	double	YES		NULL	

desc bankaccount;

```
12 •  desc bankaccount;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
accno	int	NO	PRI	NULL	
Branch_name	varchar(20)	NO	PRI	NULL	
balance	double	YES		NULL	

desc bankcustomer;

```
14 •  desc bankcustomer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
customer_street	varchar(20)	YES		NULL	
customer_city	varchar(20)	YES		NULL	

desc depositer;

```
16 •  desc depositer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
accno	int	NO	PRI	NULL	

desc loan;

```
18 •  desc loan;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
LOAN_number	int	YES		NULL	
Branch_name	varchar(20)	NO	PRI	NULL	
ammount	double	YES		NULL	

Inserting Values into the table

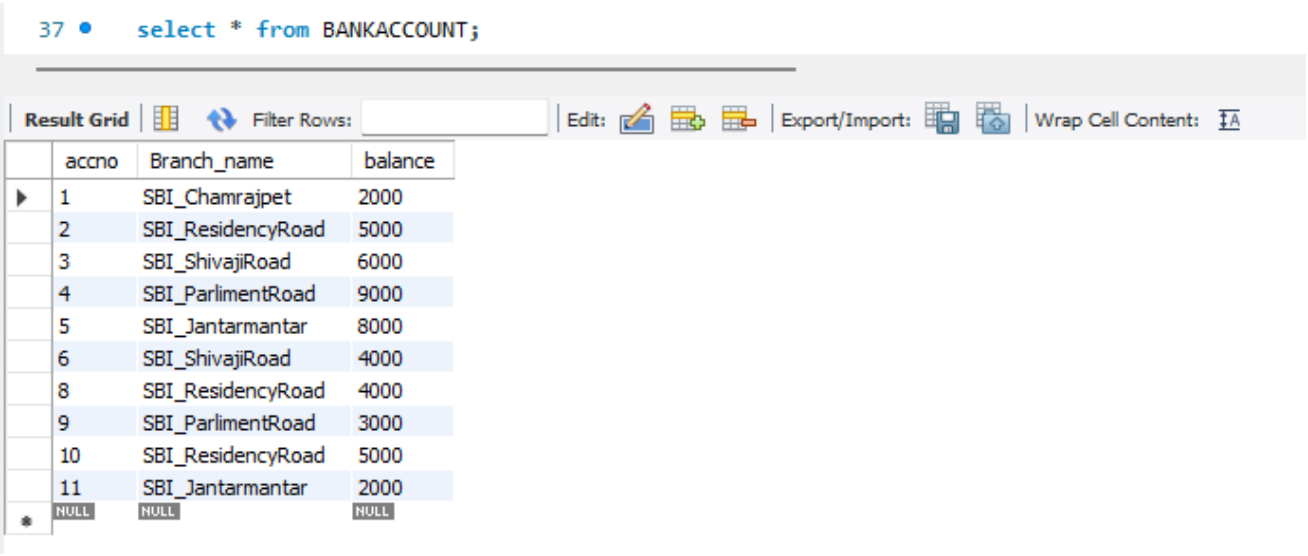
insert into BRANCH VALUES("SBI_Chamrajpet","Bangalore",50000);
insert into BRANCH VALUES("SBI_ResidencyRoad","Bangalore",10000);
insert into BRANCH VALUES("SBI_ShivajiRoad","Bombay",20000);
insert into BRANCH VALUES("SBI_ParlimentRoad","Delhi",10000);
insert into BRANCH VALUES("SBI_Jantarmantar","Delhi",20000);
select * from BRANCH;

```
25 •  select * from BRANCH;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Branch_name	branch_city	Assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

```
insert into BANKACCOUNT values(1,"SBI_Chamrajpet",2000);
insert into BANKACCOUNT values(2,"SBI_ResidencyRoad",5000);
insert into BANKACCOUNT values(3,"SBI_ShivajiRoad",6000);
insert into BANKACCOUNT values(4,"SBI_ParlimentRoad",9000);
insert into BANKACCOUNT values(5,"SBI_Jantarmantar",8000);
insert into BANKACCOUNT values(6,"SBI_ShivajiRoad",4000);
insert into BANKACCOUNT values(8,"SBI_ResidencyRoad",4000);
insert into BANKACCOUNT values(9,"SBI_ParlimentRoad",3000);
insert into BANKACCOUNT values(10,"SBI_ResidencyRoad",5000);
insert into BANKACCOUNT values(11,"SBI_Jantarmantar",2000);
select * from BANKACCOUNT;
```



accno	Branch_name	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
NULL	NULL	NULL

```
insert into BANKCUSTOMER values("Avinash","Bull_Temple_Road","Bangalore");
insert into BANKCUSTOMER values("Dinesh","Bannergatta_Road","Bangalore");
insert into BANKCUSTOMER values("Mohan","NationalCollege_Road","Bangalore");
insert into BANKCUSTOMER values("Nikil","Akbar_Road","Delhi");
insert into BANKCUSTOMER values("Ravi","Prithviraj_Road","Delhi");
select * from BANKCUSTOMER;
```

```
44 •  select * from BANKCUSTOMER;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_name	customer_street	customer_city
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);
insert into DEPOSITER values("Dinesh",2);
insert into DEPOSITER values("Nikil",4);
insert into DEPOSITER values("Ravi",5);
insert into DEPOSITER values("Avinash",8);
insert into DEPOSITER values("Nikil",9);
insert into DEPOSITER values("Dinesh",10);
insert into DEPOSITER values("Nikil",11);
select * from DEPOSITER;
```

```
54 •  select * from DEPOSITER;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_name	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);
insert into LOAN values(2,"SBI_ResidencyRoad",2000);
insert into LOAN values(3,"SBI_ShivajiRoad",3000);
insert into LOAN values(4,"SBI_ParlimentRoad",4000);
insert into LOAN values(5,"SBI_Jantarmantar",5000);
select * from LOAN;
```

```
61 •   select * from LOAN;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

LOAN_number	Branch_name	ammount
1	SBI_Chamrajpet	1000
5	SBI_Jantarmantar	5000
4	SBI_ParlimentRoad	4000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
NULL	NULL	NULL

Queries

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

select branch_name, CONCAT(assets/100000, 'Lakhs')assets_in_lakhs from branch;

```
56
57      -- Queries to do --
58
59 •    select branch_name, CONCAT(assets/100000, 'Lakhs')assets_in_lakhs   from branch;
60
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

branch_name	assets_in_lakhs
SBI_Chamrajpet	0.5Lakhs
SBI_Jantarmantar	0.2Lakhs
SBI_ParlimentRoad	0.1Lakhs
SBI_ResidencyRoad	0.1Lakhs
SBI_ShivajiRoad	0.2Lakhs

- **Find all the customers who have at least two accounts at the Same branch (ex.SBI_ResidencyRoad)**

select d.customer_name from DEPOSITER d, BANKACCOUNT b where b.branch_name="SBI_ResidencyRoad" and d.accno=b.accno group by d.customer_name having count(d.accno)>=2;

```
61 •   select d.customer_name from DEPOSITER d, BANKACCOUNT b where b.branch_name="SBI_ResidencyRoad" and
62     d.accno=b.accno group by d.customer_name having count(d.accno)>=2;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name
Dinesh

- **Create a view which gives each branch the sum of the amount of all the**

loans at the branch.

create view sum_of_loan as select Branch_name, SUM(balance) from BANKACCOUNT group by branch_name;

select * from sum_of_loan;

```
63
64 •    create view sum_of_loan as select Branch_name, SUM(balance) from BANKACCOUNT group by branch_name;
65 •    select * from sum_of_loan;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Branch_name	SUM(balance)
SBI_Chamrajpet	2000
SBI_Jantarmantar	10000
SBI_ParlimentRoad	12000
SBI_ResidencyRoad	14000
SBI_ShivajiRoad	10000

More Queries on Bank Database

Question
(Week 4)

-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)
-Borrower (customer-name: String, loan-number: int)

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

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

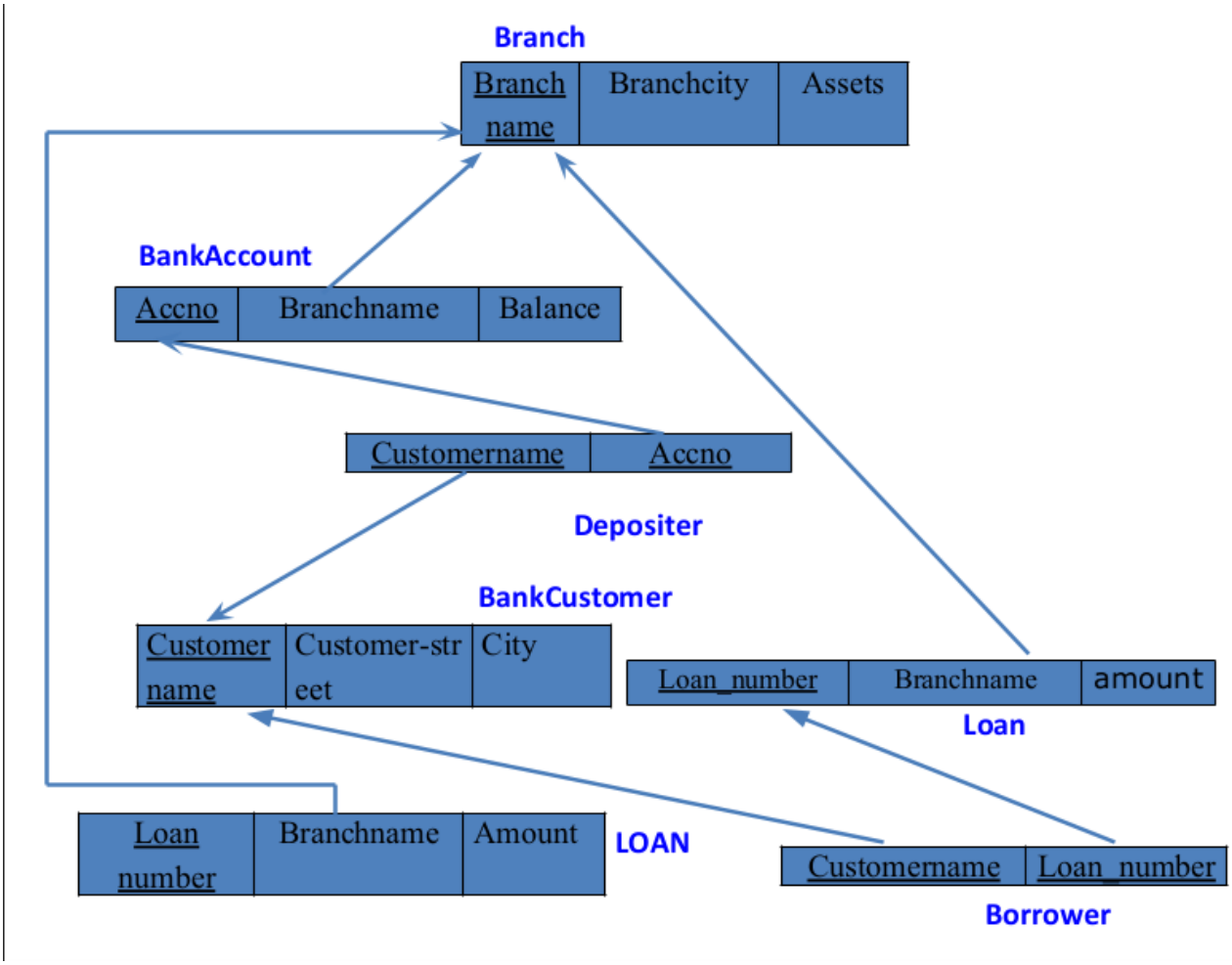
-Find all customers who have both an account and a loan at the Bangalore branch

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

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

-Update the Balance of all accounts by 5%

Schema Diagram



Creating table

create table Borrower (customer_name varchar(20), loan_number int,
foreign key (customer_name) references bankcustomer(customer_name),
foreign key (loan_number) references LOAN(loan_number));

Structure of the table

desc Borrower;

```
3 •  desc Borrower;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	YES	MUL	NULL	
loan_number	int	YES	MUL	NULL	

Inserting Values into the table

insert into Borrower values('Avinash',1);
insert into Borrower values('Dinesh',2);
insert into Borrower values('Mohan',3);
insert into Borrower values('Nikil',4);
insert into Borrower values('Ravi',5);
select * from Borrower;

```
5 •  select * from Borrower;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	loan_number
Avinash	1
Dinesh	2
Mohan	3
Nikil	4
Ravi	5

Queries

- **Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).**

  SELECT d.customer_name From bankaccount a,branch b, depositer d WHERE b.branch_name=a.branch_name AND a.accno=d.accno AND b.branch_city='Delhi' GROUP BY d.customer_name HAVING COUNT(distinct b.branch_name)=(SELECT COUNT(branch_name)FROM branch WHERE branch_city='Delhi');

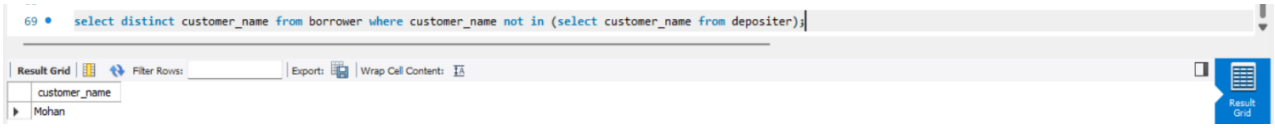
```
65 •  SELECT d.customer_name From bankaccount a,branch b, depositer d WHERE b.branch_name=a.branch_name AND
66    a.accno=d.accno AND b.branch_city='Delhi' GROUP BY d.customer_name
67    HAVING COUNT(distinct b.branch_name)=(SELECT COUNT(branch_name)FROM branch WHERE branch_city='Delhi');
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name
Nikil

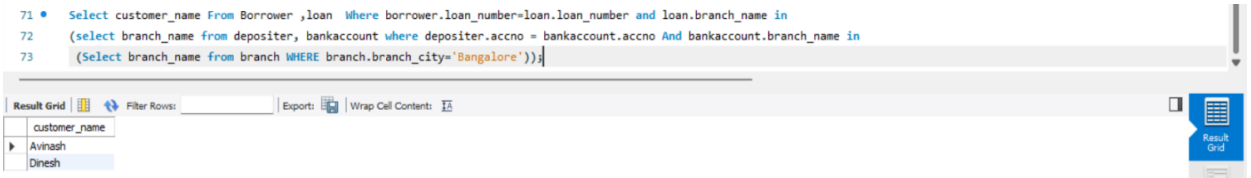
- **Find all customers who have a loan at the bank but do not have an account.**

select distinct customer_name from borrower where customer_name not in (select customer_name from depositer);



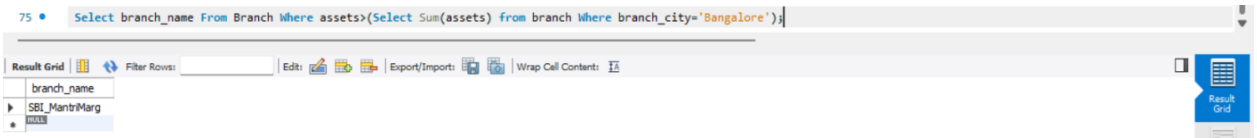
- **Find all customers who have both an account and a loan at the Bangalore branch**

  Select customer_name From Borrower ,loan Where borrower.loan_number=loan.loan_number and loan.branch_name in (select branch_name from depositer, bankaccount where depositer.accno = bankaccount.accno And bankaccount.branch_name in (Select branch_name from branch WHERE branch.branch_city='Bangalore'));



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

  Select branch_name From Branch Where assets>(Select Sum(assets) from branch Where branch_city='Bangalore');



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

DELETE FROM bankaccount WHERE branch_name IN(SELECT branch_name FROM branch WHERE branch_city='Bombay');

delete from depositer where accno in
(select accno from branch, bankaccount where branch_city = 'Bombay' and branch.branch_name = bankaccount.branch_name);

- **Update the Balance of all accounts by 5%**

update bankaccount
set balance = balance * 1.05

Employee Database

Question
(Week 5)

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.

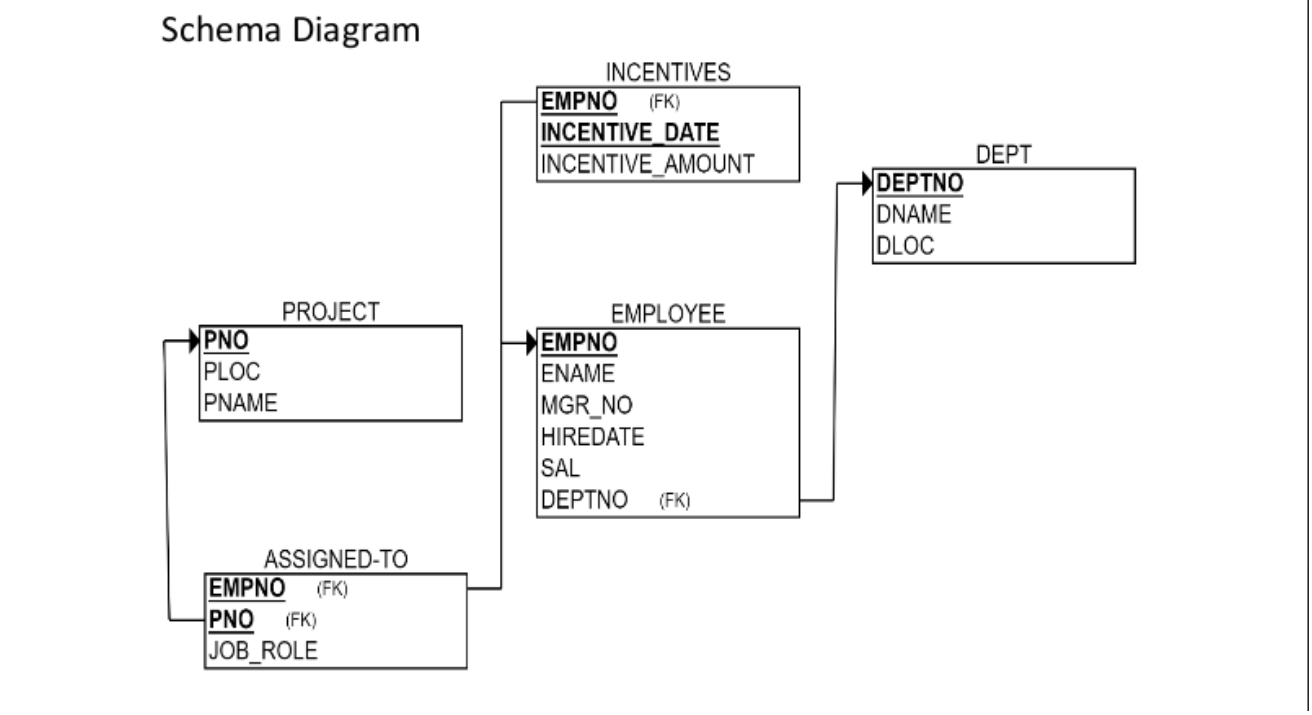
2. Enter greater than five tuples for each table.

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.

Schema Diagram



Create database

create database employee;
use employee;

Create table

create table DEPT(DEPTNO int, DNAME varchar(20),DLOC varchar(20),primary key(DEPTNO));

create table EMPLOYEE(EMPNO int,ENAME varchar(20),MGR_NO int, HIREDATE date, SAL int, DEPTNO int, primary key(EMPNO,DEPTNO),foreign key(DEPTNO) references DEPT(DEPTNO));

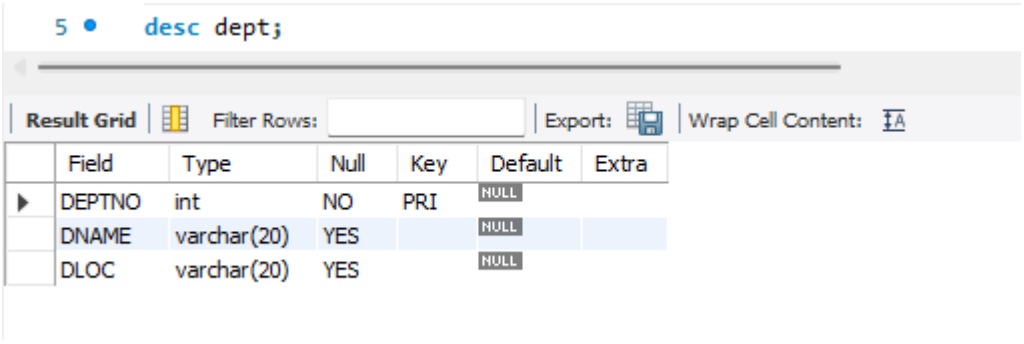
create table PROJECT(PNO int,PNAME varchar(20),PLOC varchar(20),primary key(PNO));

create table INCENTIVES(EMPNO int,INCENTIVE_DATE date,INCENTIVE_AMOUNT int, primary key(EMPNO),foreign key(EMPNO) references EMPLOYEE(EMPNO));

create table ASSIGNED_TO(EMPNO int,PNO int,JOB_ROLE varchar(20),primary key(EMPNO,PNO),FOREIGN KEY(EMPNO) references EMPLOYEE(EMPNO),FOREIGN KEY(PNO) REFERENCES PROJECT(PNO));

Structure of the table

desc dept;



Field	Type	Null	Key	Default	Extra
DEPTNO	int	NO	PRI	NULL	
DNAME	varchar(20)	YES		NULL	
DLOC	varchar(20)	YES		NULL	

desc employee;

```
6 •  desc employee;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
ENAME	varchar(20)	YES		NULL	
MGR_NO	int	YES		NULL	
HIREDATE	date	YES		NULL	
SAL	int	YES		NULL	
DEPTNO	int	NO	PRI	NULL	

desc project;

```
7 •  desc project;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
PNO	int	NO	PRI	NULL	
PLOC	varchar(20)	YES		NULL	
PNAME	varchar(20)	YES		NULL	

desc incentives;

```
8 •  desc incentives;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
INCENTIVE_DATE	date	NO	PRI	NULL	
INCENTIVE_AMOUNT	int	YES		NULL	

desc assigned_to;

```
9 •  desc assigned_to;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
PNO	int	NO	PRI	NULL	
JOB_ROLE	varchar(20)	YES		NULL	

Inserting Values into the table

INSERT INTO DEPT (DEPTNO, DNAME, DLOC) VALUES (10, 'SALES', 'Bengaluru');
INSERT INTO DEPT (DEPTNO, DNAME, DLOC) VALUES (20, 'IT', 'Hyderabad');
INSERT INTO DEPT (DEPTNO, DNAME, DLOC) VALUES (30, 'HR', 'Mysuru');
INSERT INTO DEPT (DEPTNO, DNAME, DLOC) VALUES (40, 'R&D', 'Bengaluru');
INSERT INTO DEPT (DEPTNO, DNAME, DLOC) VALUES (50, 'MARKETING', 'Mumbai');
INSERT INTO DEPT (DEPTNO, DNAME, DLOC) VALUES (60, 'FINANCE', 'Hyderabad');
select * from DEPT;

```
5 •  select * from  dept;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

DEPTNO	DNAME	DLOC
10	SALES	Bengaluru
20	IT	Hyderabad
30	HR	Mysuru
40	R&D	Bengaluru
50	MARKETING	Mumbai
60	FINANCE	Hyderabad
NULL	NULL	NULL

INSERT INTO PROJECT (PNO, PNAME, PLOC) VALUES (100, 'Alpha', 'Bengaluru');
INSERT INTO PROJECT (PNO, PNAME, PLOC) VALUES (200, 'Beta', 'Hyderabad');
INSERT INTO PROJECT (PNO, PNAME, PLOC) VALUES (300, 'Gamma', 'Mysuru');
INSERT INTO PROJECT (PNO, PNAME, PLOC) VALUES (400, 'Delta', 'Mumbai');
INSERT INTO PROJECT (PNO, PNAME, PLOC) VALUES (500, 'Omega', 'Bengaluru');
INSERT INTO PROJECT (PNO, PNAME, PLOC) VALUES (600, 'Zeta', 'Delhi');
SELECT * FROM PROJECT;

```
7 •  select * from  project;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

PNO	PLOC	PNAME
100	Bengaluru	Alpha
200	Hyderabad	Beta
300	Mysuru	Gamma
400	Mumbai	Delta
500	Bengaluru	Omega
600	Delhi	Zeta
NULL	NULL	NULL

INSERT INTO EMPLOYEE (EMPNO, ENAME, MGR_NO, HIREDATE, SAL, DEPTNO)
VALUES (7001, 'Alice', 7004,'2020-01-15', 80000, 10);
INSERT INTO EMPLOYEE (EMPNO, ENAME, MGR_NO, HIREDATE, SAL, DEPTNO)
VALUES (7002, 'Bob', 7005,'2019-03-22', 95000, 20);
INSERT INTO EMPLOYEE (EMPNO, ENAME, MGR_NO, HIREDATE, SAL, DEPTNO)
VALUES (7003, 'Charlie', 7004,'2021-05-10', 60000, 30);
INSERT INTO EMPLOYEE (EMPNO, ENAME, MGR_NO, HIREDATE, SAL, DEPTNO)
VALUES (7004, 'David', NULL, '2018-06-01', 120000, 10);
INSERT INTO EMPLOYEE (EMPNO, ENAME, MGR_NO, HIREDATE, SAL, DEPTNO)
VALUES (7005, 'Eve', NULL, '2017-11-30', 110000, 20);
INSERT INTO EMPLOYEE (EMPNO, ENAME, MGR_NO, HIREDATE, SAL, DEPTNO)
VALUES (7006, 'Frank', 7005, '2022-02-18', 75000, 50);
SELECT * FROM EMPLOYEE;

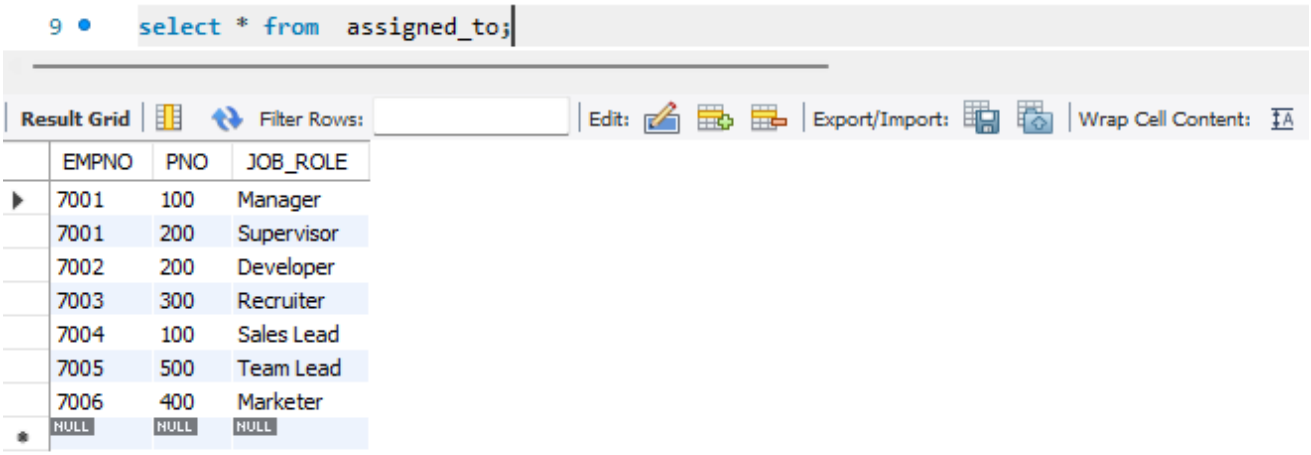
```
6 •  select * from  employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EMPNO	ENAME	MGR_NO	HIREDATE	SAL	DEPTNO
7001	Alice	7004	2020-01-15	80000	10
7002	Bob	7005	2019-03-22	95000	20
7003	Charlie	7004	2021-05-10	60000	30
7004	David	NULL	2018-06-01	120000	10
7005	Eve	NULL	2017-11-30	110000	20
7006	Frank	7005	2022-02-18	75000	50
NULL	NULL	NULL	NULL	NULL	NULL

INSERT INTO ASSIGNED_TO (EMPNO, PNO, JOB_ROLE) VALUES (7001, 100,
'Manager');

```
INSERT INTO ASSIGNED_TO (EMPNO, PNO, JOB_ROLE) VALUES (7002, 200,
'Developer');
INSERT INTO ASSIGNED_TO (EMPNO, PNO, JOB_ROLE) VALUES (7003, 300,
'Recruiter');
INSERT INTO ASSIGNED_TO (EMPNO, PNO, JOB_ROLE) VALUES (7004, 100, 'Sales
Lead');
INSERT INTO ASSIGNED_TO (EMPNO, PNO, JOB_ROLE) VALUES (7005, 500, 'Team
Lead');
INSERT INTO ASSIGNED_TO (EMPNO, PNO, JOB_ROLE) VALUES (7001, 200,
'Supervisor');
INSERT INTO ASSIGNED_TO (EMPNO, PNO, JOB_ROLE) VALUES (7006, 400,
'Marketer');
SELECT * FROM ASSIGNED_TO;
```



```
9 • select * from assigned_to;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EMPNO	PNO	JOB_ROLE
7001	100	Manager
7001	200	Supervisor
7002	200	Developer
7003	300	Recruiter
7004	100	Sales Lead
7005	500	Team Lead
7006	400	Marketer
NULL	NULL	NULL

```
INSERT INTO INCENTIVES (EMPNO, INCENTIVE_DATE, INCENTIVE_AMOUNT)
VALUES (7001, '2023-01-01', 5000);
INSERT INTO INCENTIVES (EMPNO, INCENTIVE_DATE, INCENTIVE_AMOUNT)
VALUES (7002, '2023-02-01', 3000);
INSERT INTO INCENTIVES (EMPNO, INCENTIVE_DATE, INCENTIVE_AMOUNT)
VALUES (7004, '2023-01-15', 5000);
SELECT * FROM INCENTIVES;
```

```
8 •  select * from  incentives;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

EMPNO	INCENTIVE_DATE	INCENTIVE_AMOUNT
7001	2023-01-01	5000
7001	2023-04-01	2500
7002	2023-02-01	3000
7002	2023-05-01	3000
7004	2023-01-15	5000
NULL	NULL	NULL

Queries

- **Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru**

```
SELECT DISTINCT T1.EMPNO
FROM ASSIGNED_TO T1
JOIN PROJECT T2 ON T1.PNO = T2.PNO
WHERE T2.PLOC IN ('Bengaluru', 'Hyderabad', 'Mysuru');
```

- **Get Employee ID's of those employees who didn't receive incentives**

```
SELECT E.EMPNO
FROM EMPLOYEE E
WHERE NOT EXISTS (
   SELECT 1
   FROM INCENTIVES I
   WHERE I.EMPNO = E.EMPNO
);
```

```
SELECT E.EMPNO
FROM EMPLOYEE E
WHERE NOT EXISTS (
    SELECT 1
    FROM INCENTIVES I
    WHERE I.EMPNO = E.EMPNO
);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

EMPNO
7005
7003
7006

- **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.**

```
SELECT
   E.ENAME,
   E.EMPNO,
   D.DNAME AS DEPARTMENT_NAME,
   A.JOB_ROLE,
   D.DLOC AS DEPARTMENT_LOCATION,
   P.PLOC AS PROJECT_LOCATION
FROM
   EMPLOYEE E
JOIN
   DEPT D ON E.DEPTNO = D.DEPTNO
JOIN
   ASSIGNED_TO A ON E.EMPNO = A.EMPNO
JOIN
   PROJECT P ON A.PNO = P.PNO
WHERE
   D.DLOC = P.PLOC;
```

```
SELECT
    E.ENAME,
    E.EMPNO,
    D.DNAME AS DEPARTMENT_NAME,
    A.JOB_ROLE,
    D.DLOC AS DEPARTMENT_LOCATION,
    P.PLOC AS PROJECT_LOCATION
FROM
    EMPLOYEE E
JOIN
    DEPT D ON E.DEPTNO = D.DEPTNO
JOIN
    ASSIGNED_TO A ON E.EMPNO = A.EMPNO
JOIN
    PROJECT P ON A.PNO = P.PNO
WHERE
    D.DLOC = P.PLOC;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ENAME	EMPNO	DEPARTMENT_NAME	JOB_ROLE	DEPARTMENT_LOCATION	PROJECT_LOCATION
Alice	7001	SALES	Manager	Bengaluru	Bengaluru
Bob	7002	IT	Developer	Hyderabad	Hyderabad
Charlie	7003	HR	Recruiter	Mysuru	Mysuru
David	7004	SALES	Sales Lead	Bengaluru	Bengaluru
Frank	7006	MARKETING	Marketer	Mumbai	Mumbai

More Queries on Employee Database

Question
(Week 6)

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.

2. Enter greater than five tuples for each table.

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

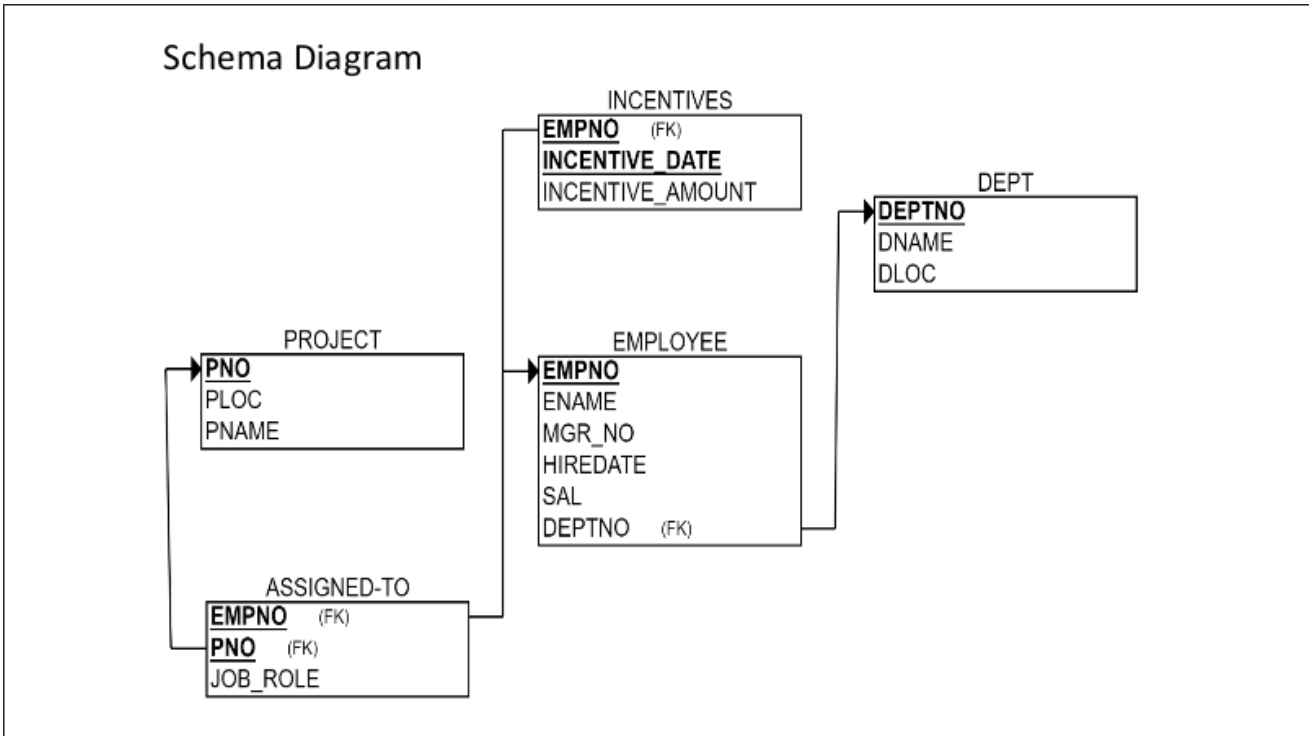
4. Display those managers name whose salary is more than average salary of his employee.

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

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

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

Schema Diagram



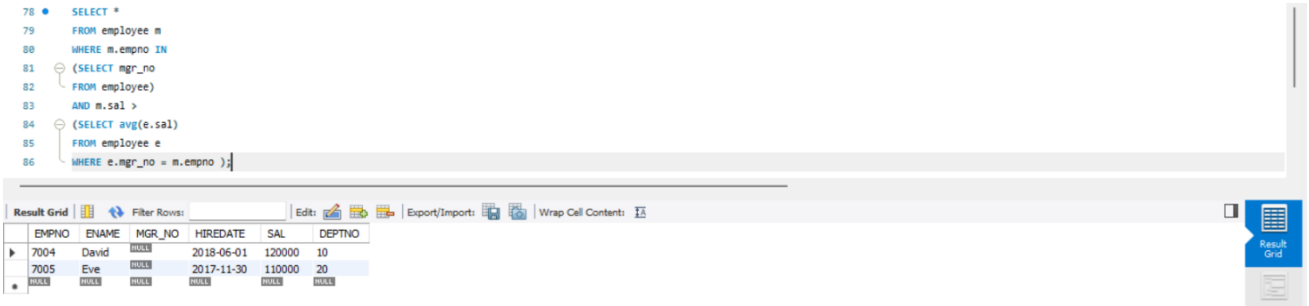
Queries

- **List the name of the managers with the maximum employee**

```
SELECT
m.ENAME AS ManagerName,COUNT(*) AS EmployeeCount FROM  EMPLOYEE e
JOIN EMPLOYEE m ON e.MGR_NO = m.EMPNO  GROUP BY m.ENAME HAVING
COUNT(*) = (SELECT MAX(mycount) FROM (SELECT COUNT(*) AS mycount FROM
EMPLOYEE WHERE MGR_NO IS NOT NULL GROUP BY MGR_NO)a);
```

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

```
SELECT *
FROM employee m
WHERE m.empno IN
(SELECT mgr_no
FROM employee)
AND m.sal >
(SELECT avg(e.sal)
FROM employee e
WHERE e.mgr_no = m.empno );
```



- **Find the name of the second top level managers of each department.**

```
select *
from employee e,incentives i
where e.empno=i.empno and 2 = ( select count(*)
from incentives j
where i.incentive_amount <= j.incentive_amount );
```

```
select *
from employee e,incentives i
where e.empno=i.empno and 2 = ( select count(*)
from incentives j
where i.incentive_amount <= j.incentive_amount );
```

EMPNO	ENAME	MGR_NO	HIREDATE	SAL	DEPTNO	EMPNO	INCENTIVE_DATE	INCENTIVE_AMOUNT
7001	Alice	7004	2020-01-15	80000	10	7001	2023-01-01	5000
7004	David	NULL	2018-06-01	120000	10	7004	2023-01-15	5000

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

```
SELECT *
FROM EMPLOYEE E
WHERE E.DEPTNO = (SELECT E1.DEPTNO
FROM EMPLOYEE E1
WHERE E1.EMPNO=E.MGR_NO);
```

```
SELECT *
FROM EMPLOYEE E
WHERE E.DEPTNO = (SELECT E1.DEPTNO
FROM EMPLOYEE E1
WHERE E1.EMPNO=E.MGR_NO);
```

EMPNO	ENAME	MGR_NO	HIREDATE	SAL	DEPTNO
7001	Alice	7004	2020-01-15	80000	10
7002	Bob	7005	2019-03-22	95000	20
NULL	NULL	NULL	NULL	NULL	NULL

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

```
SELECT distinct e.ename
from employee e, incentives i
WHERE (SELECT max(sal+incentive_amount)
FROM employee,incentives) >= ANY
(SELECT sal
FROM employee e1
where e.deptno=e1.deptno);
```

```
SELECT distinct e.ename
from employee e, incentives i
WHERE (SELECT max(sal+incentive_amount)
FROM employee,incentives) >= ANY
(SELECT sal
FROM employee e1
where e.deptno=e1.deptno);
```

ename
Alice
Bob
Charlie
David
Eve
Frank

Supplier Database

Question
(Week 7)

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.

2. Insert appropriate records in each table.

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

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

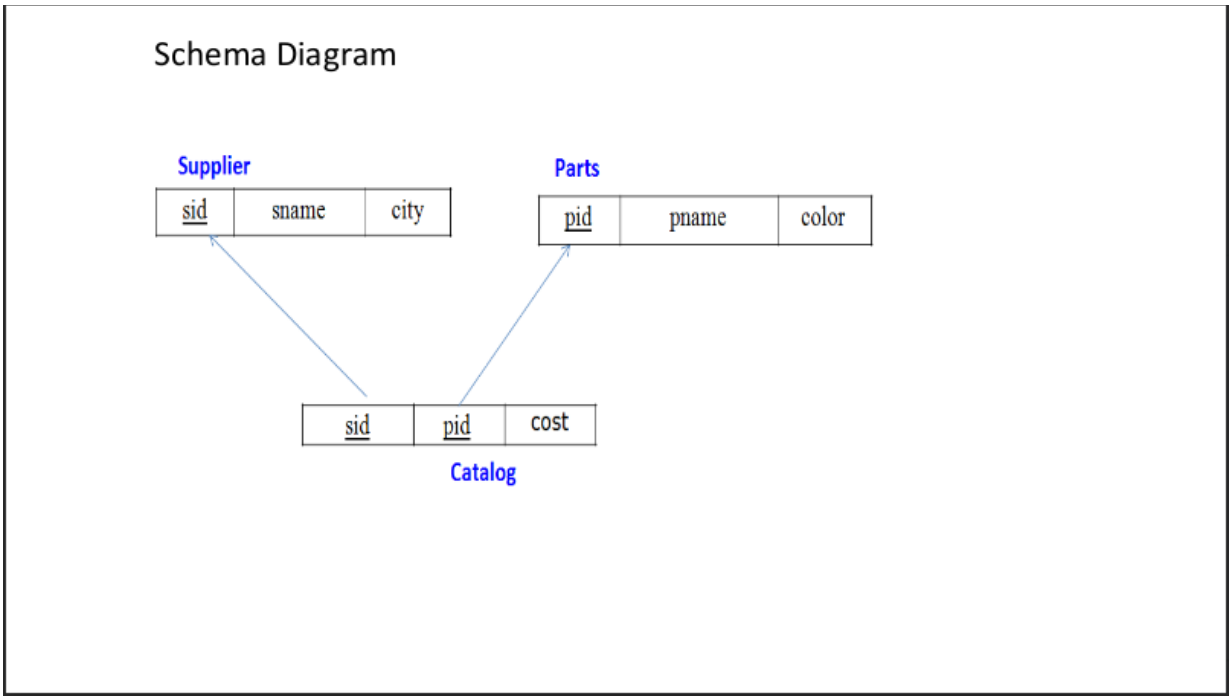
5. Find the snames of suppliers who supply every red part.

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

7. 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).

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

Schema Diagram



Create database

create database if not exists supplier;
use supplier;

Create table

create table Supplier(SID int,SNAME varchar(20),CITY varchar(20),primary key(SID));

create table Parts(PID int, PNAME varchar(20),color varchar(20),primary key(PID));

create table Catalog(SID int, PID int, cost int, primary key(SID,PID), foreign key(SID) references Supplier(SID), foreign key (PID) references Parts(PID));

Structure of the table

desc Supplier;

```
5 •   desc Supplier;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
SID	int	NO	PRI	NULL	
SNAME	varchar(20)	YES		NULL	
CITY	varchar(20)	YES		NULL	

desc Parts;

```
6 •  desc Parts;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
PID	int	NO	PRI	NULL	
PNAME	varchar(20)	YES		NULL	
color	varchar(20)	YES		NULL	

desc Catalog;

```
7 •  desc Catalog;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
SID	int	NO	PRI	NULL	
PID	int	NO	PRI	NULL	
cost	int	YES		NULL	

Insert Values into table

insert into Supplier (SID,SNAME,CITY) values
(10001,"Acme Widget","bangalore"),
(10002,"Johns","Kolkata"),
(10003,"Vimal","Mumbai"),
(10004,"Reliance","Delhi");

select * from Supplier;

```
5 •  select * from  Supplier;
```

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

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

insert into Parts(PID,PNAME,color) values
(20001,"Book","Red"),
(20002,"Pen","Red"),
(20003,"Pencil","Green"),
(20004,"Mobile","Green"),
(20005,"Charger","Black");

select * from Parts;

```
6 •  select * from  Parts;
```

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

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

insert into Catalog(SID,PID,cost) values
(10001,20001,10),
(10001,20002,10),
(10001,20003,30),
(10001,20004,10),
(10001,20005,10),
(10002,20001,10),
(10002,20002,20),
(10003,20003,30),
(10004,20003,40);
select * from Catalog;

```
7 •  select * from  Catalog;
```

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

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
NULL	NULL	NULL

Queries

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

select PNAME from Parts where PID in (select PID from Catalog);

```
31    -- Find the pnames of parts for which there is some supplier. --
32    select PNAME from Parts where PID in (select PID from Catalog);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

PNAME
Book
Pen
Pencil
Mobile
Charger

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

select distinct s.SNAME from Supplier s join catalog c1 on s.SID=c1.SID where (select count(distinct PID) from catalog c2 where c2.SID=c1.SID)= (select count(*) from Parts);

```
34    -- Find the snames of suppliers who supply every part. --
35 •  select distinct s.SNAME from Supplier s join catalog c1 on s.SID=c1.SID where (select count(distinct PID) from catalog c2 where c2.SID=c1.SID)= (select count(*) from Parts);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

SNAME
Acme Widget

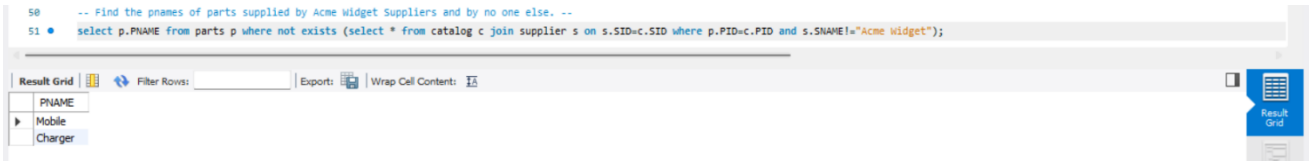
- **Find the snames of suppliers who supply every red part.**

```
SELECT s.SNAME
FROM Supplier s
JOIN Catalog c ON s.SID = c.SID
JOIN Parts p ON c.PID = p.PID
WHERE p.color = 'Red'
GROUP BY s.SID, s.SNAME  -- Group by both ID and Name
HAVING COUNT(DISTINCT p.PID) = (
   -- This subquery gets the TOTAL number of red parts
```

```
    SELECT COUNT(p2.PID)
    FROM Parts p2
    WHERE p2.color = 'Red'
);
```

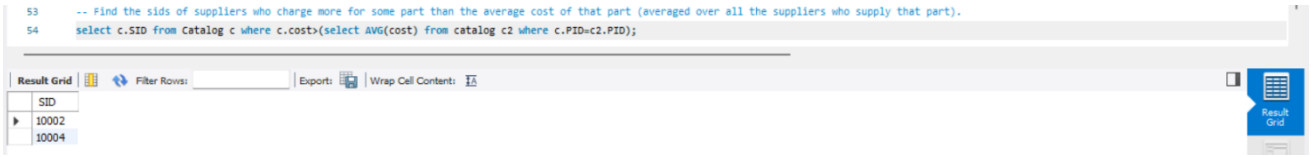
- **Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.**

select p.PNAME from parts p where not exists (select * from catalog c join supplier s on s.SID=c.SID where p.PID=c.PID and s.SNAME!="Acme Widget");



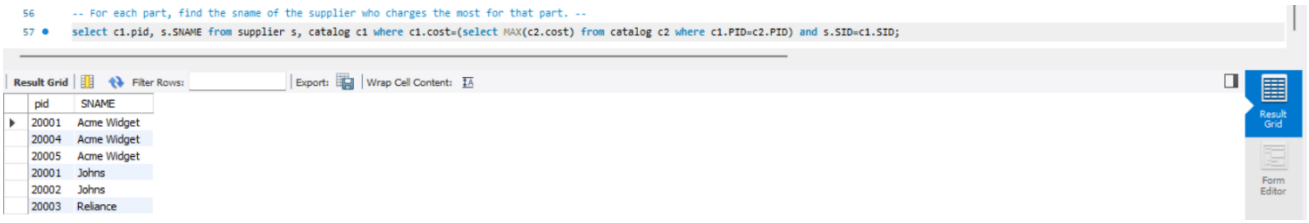
- **Find the sids of suppliers who charge more for some part than the average**
  cost of that part (averaged over all the suppliers who supply that part).

select c.SID from Catalog c where c.cost>(select AVG(cost) from catalog c2 where c.PID=c2.PID);



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

select c1.pid, s.SNAME from supplier s, catalog c1 where c1.cost=(select MAX(c2.cost) from catalog c2 where c1.PID=c2.PID) and s.SID=c1.SID;


56 -- For each part, find the sname of the supplier who charges the most for that part. --
57 select c1.pid, s.SNAME from supplier s, catalog c1 where c1.cost=(select MAX(c2.cost) from catalog c2 where c1.PID=c2.PID) and s.SID=c1.SID;
Result Grid
Filter Rows:
Export:
Wrap Cell Content:
pid SNAME
20001 Acme Widget
20004 Acme Widget
20005 Acme Widget
20001 Johns
20002 Johns
20003 Reliance
Result Grid
Form Editor

NoSql Lab 1

Question

(Week 8)

Perform the following DB operations using MongoDB.

1. Create a database "Student" with the following attributes Rollno, Age, ContactNo, Email-Id.
2. Insert appropriate values
3. Write query to update Email-Id of a student with rollno 10.
4. Replace the student name from "ABC" to "FEM" of rollno 11.
5. Export the created table into local file system
6. Drop the table
7. Import a given csv datasheet from local file system inro collection.

Create database

```
db.createCollection("Student");
```

Create table & Inserting Values into the table

```
db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
```

Structure of the table

```
db.Student.find();
```

```
Atlas atlas-13yfay-shard-0 [primary] test> db.Student.insert({Rollno:11,Age:21,Cont:3376,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746ba0f207c5c04af227804") }
}
Atlas atlas-13yfay-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("6746b8ff207c5c04af2277ff"),
    Rollno: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b941207c5c04af227800"),
    Rollno: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b980207c5c04af227801"),
    Rollno: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b999207c5c04af227802"),
    Rollno: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9b2207c5c04af227803"),
    Rollno: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  },
  {
    _id: ObjectId("6746ba0f207c5c04af227804"),
    Rollno: 11,
    Age: 21,
    Cont: 3376,
    email: 'pani.de9@gmail.com'
  }
```

Queries

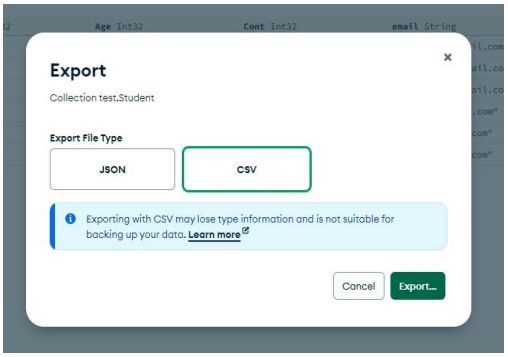
- **Write a query to update the Email-Id of a student with rollno 10.**
- **db.Student.update({rollno:5},{$set:{email:"abhinav@gmail.com"}});**

```
  },
  {
    _id: ObjectId("6746b9b2207c5c04af227803"),
    Rollno: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  },
```

- **Replace the student name from "ABC" to "FEM" of rollno 11.**
  db.Student.insert({rollno:11,age:22,name:"ABC",cont:2276,email:"madhura@gmail.com"}); db.Student.update({rollno:11,name:"ABC"},{$set:{name:"FEM"}})

```
{
  _id: ObjectId("6746ba0f207c5c04af227804"),
  Rollno: 11,
  Age: 21,
  Cont: '2276',
  email: 'rea.de9@gmail.com',
  name: 'FEM'
}
```

- **Export the created table into local file system**



	A	B	C	D	E	F
1	_id	RollNo	Age	Cont	email	Name
2	67613bdde754bbf059d14a0e	1	21	9876	antara.de9@gmail.com	
3	67613bf5e754bbf059d14a0f	2	22	9976	anushka.de9@gmail.com	
4	67613bfce754bbf059d14a10	3	21	5576	anubhav.de9@gmail.com	
5	67613c00e754bbf059d14a11	4	20	4476	pani.de9@gmail.com	
6	67613c07e754bbf059d14a12	10	23	2276	Abhinav@gmail.com	
7	67613cd2e754bbf059d14a13	11	22	2276	rea.de9@gmail.com	FEM

- **Drop the table**

db.Student.drop()

```
Atlas atlas-uyucz2-shard-0 [primary] test> db.Student.drop();
true
```

- **Import a given csv datasheet from local file system inro collection.**

	A	B	C	D	E	F
1	_id	RollNo	Age	Cont	email	Name
2	67613bdde754bbf059d14a0e	1	21	9876	antara.de9@gmail.com	
3	67613bf5e754bbf059d14a0f	2	22	9976	anushka.de9@gmail.com	
4	67613bfce754bbf059d14a10	3	21	5576	anubhav.de9@gmail.com	
5	67613c00e754bbf059d14a11	4	20	4476	pani.de9@gmail.com	
6	67613c07e754bbf059d14a12	10	23	2276	Abhinav@gmail.com	
7	67613cd2e754bbf059d14a13	11	22	2276	rea.de9@gmail.com	FEM

Student

	_id ObjectId	RollNo Int32	Age Int32	Cont Int32	email String	Name String
1	ObjectId('67613bdde754bb…	1	21	9876	"antara.de9@gmail.com"	No field
2	ObjectId('67613bf5e754bb…	2	22	9976	"anushka.de9@gmail.com"	No field
3	ObjectId('67613bfce754bb…	3	21	5576	"anubhav.de9@gmail.com"	No field
4	ObjectId('67613c00e754bb…	4	20	4476	"pani.de9@gmail.com"	No field
5	ObjectId('67613c07e754bb…	10	23	2276	"Abhinav@gmail.com"	No field
6	ObjectId('67613cd2e754bb…	11	22	2276	"rea.de9@gmail.com"	"FEM"

NoSql Lab 2

Question

(Week 9)

Perform the following DB operations using MongoDB

1. Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type
2. Insert at least 5 values into the table
3. Write a query to display those records whose total account balance is greater than 1200 of account type 'Checking' for each customer_id.
4. Determine Minimum and Maximum account balance for each customer_id.
5. Export the created collection into local file system
6. Drop the table
7. Import a given csv dataset from local file system into mongodb collection.

Create table

```
db.createCollection("Customer");
```

Inserting Values into the table

```
db.Customer.insertMany([{custid: 1, acc_bal:10000,
acc_type: "Saving"}, {custid: 1, acc_bal:20000, acc_type:
"Checking"}, {custid: 3, acc_bal:50000, acc_type:
"Checking"}, {custid: 4, acc_bal:10000, acc_type:
"Saving"}, {custid: 5, acc_bal:2000, acc_type:
"Checking"}]);
```

Structure of table

```
db.Customer.find()
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.Customer.find()
[
  {
    _id: ObjectId("6751fde06a59c75535ff9949"),
    custid: 1,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994a"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994b"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994c"),
    custid: 4,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994d"),
    custid: 5,
    acc_bal: 2000,
    acc_type: 'Checking'
  }
]
```

Queries

- **Finding all checking accounts with balance greater than 12000**

db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});

```
Atlas atlas-13yfay-shard-0 [primary] test> db.customer.find({acc_bal:{$gt:12000},acc_type:"Checking"});
[
  {
    _id: ObjectId("674ff86c8df86e77109b3e37"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("674ff8b78df86e77109b3e38"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  }
]
```

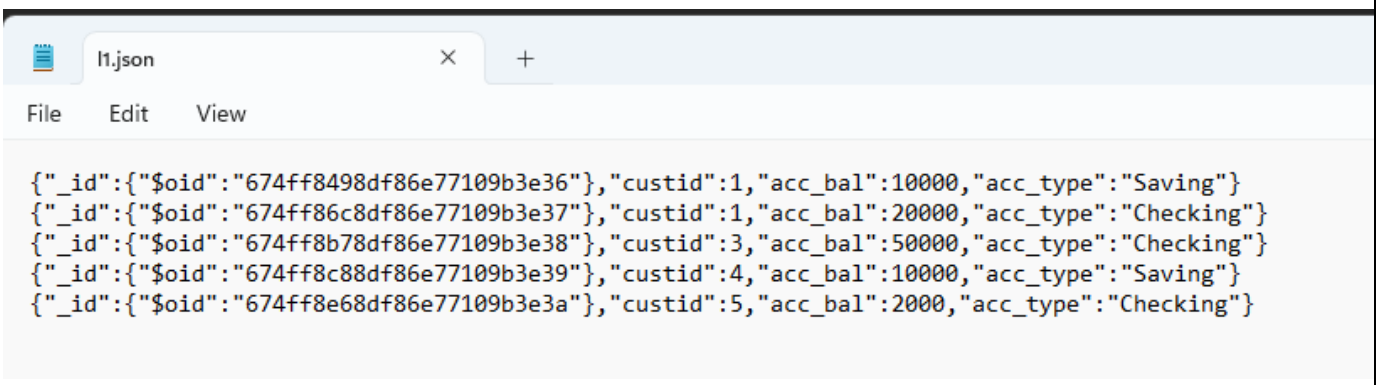
- **Finding the maximum and minimum balance of each customer**

db.Customer.aggregate([{$group:{_id:"$custid", minBal:{$min:"$acc_bal"}, maxBal: {$max:"$acc_bal"}}}]);

```
Atlas atlas-13yfay-shard-0 [primary] test> db.customer.aggregate([{$group:{_id:"$custid",minBal:{$min:"$acc_bal"},maxBal:{$max:"$acc_bal"}}}]);
[
  { _id: 4, minBal: 10000, maxBal: 10000 },
  { _id: 1, minBal: 10000, maxBal: 20000 },
  { _id: 3, minBal: 50000, maxBal: 50000 },
  { _id: 5, minBal: 2000, maxBal: 2000 }
]
Atlas atlas-13yfay-shard-0 [primary] test>
```

- **Exporting the collection to a json file**

mongoexport mongodb+srv://Likith:@cluster0.xbmgopf.mongodb.net/test --collection=Customer -- out D:\1BM23CS171\l1.json



- **Dropping collection “Customer”**

db.Customer.drop();

```
Atlas atlas-13yfay-shard-0 [primary] test> db.customer.drop();
true
Atlas atlas-13yfay-shard-0 [primary] test>
```

- **Exporting from a json file to the collection**

mongoimport mongodb+srv://Prajwal:@cluster0.xbmgopf.mongodb.net/test --collection=Customer --file D:\1BM24CS192\l1.json

db.Customer.find();

NoSql Lab 3

Question
(Week 10)

1. Write a MongoDB query to display all the documents in the collection restaurants.
2. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.
3. 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.
4. Write a MongoDB query to find the average score for each restaurant.
5. Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

Creating table

```
db.createCollection("restaurants");
```

Inserting Values into the table

```
db.restaurants.insertMany([
{ name: "Meghna Foods", town: "Jayanagar", cuisine: "Indian", score: 8, address: {
zipcode: "10001", street: "Jayanagar“
 } },
{ name: "Empire", town: "MG Road", cuisine: "Indian", score: 7, address: { zipcode:
"10100", street: "MG Road" } },
{ name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: {
zipcode: "20000", street: "Indiranagar" } },
{ name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode:
"10300", street: "Majestic" } },
{ name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: {
zipcode: "10400",
```

street: "Malleshwaram" }
}])

Structure of the table

db.Restaurant.find()

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({})
[
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'jayanagar' }
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'M G Road' }
  },
  {
    _id: ObjectId("675002dbf345f747889620bb"),
    name: 'Chinese Wok',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

Queries

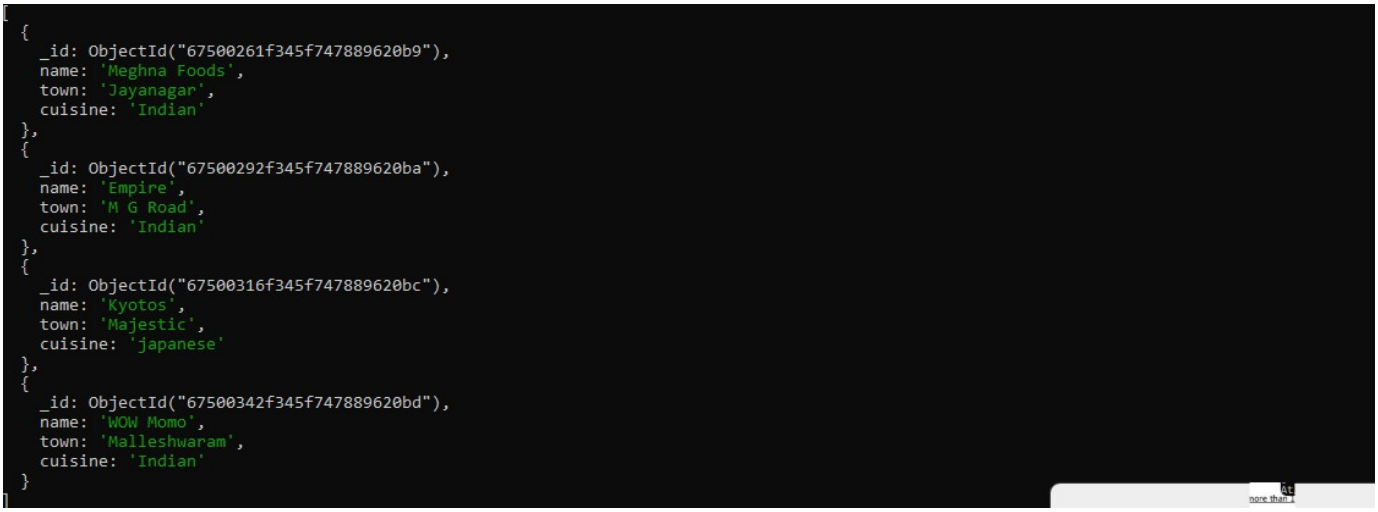
- **Arranging name of restaurants in descending order**

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

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({}).sort({name:-1})
[
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'jayanagar' }
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'M G Road' }
  },
  {
    _id: ObjectId("675002dbf345f747889620bb"),
    name: 'Chinese Wok',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

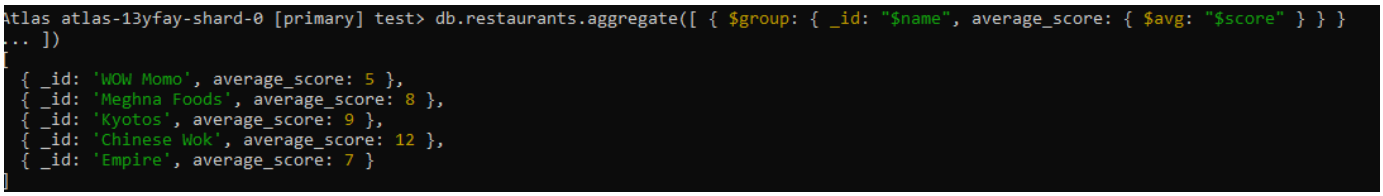
- **Finding the restaurant id, name, town and the cuisine for those restaurants which achieved a score that is not more than 10**

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



- **Finding the average score for each restaurant**

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



- **Finding name and address of the restaurants that have a zipcode that starts with 10**

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