VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

Mohammed Moinuddin A (1BM24CS170)

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 **Mohammed Moinuddin A (1BM24CS170),** 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.

Rashmi H Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Experiment 1: 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.

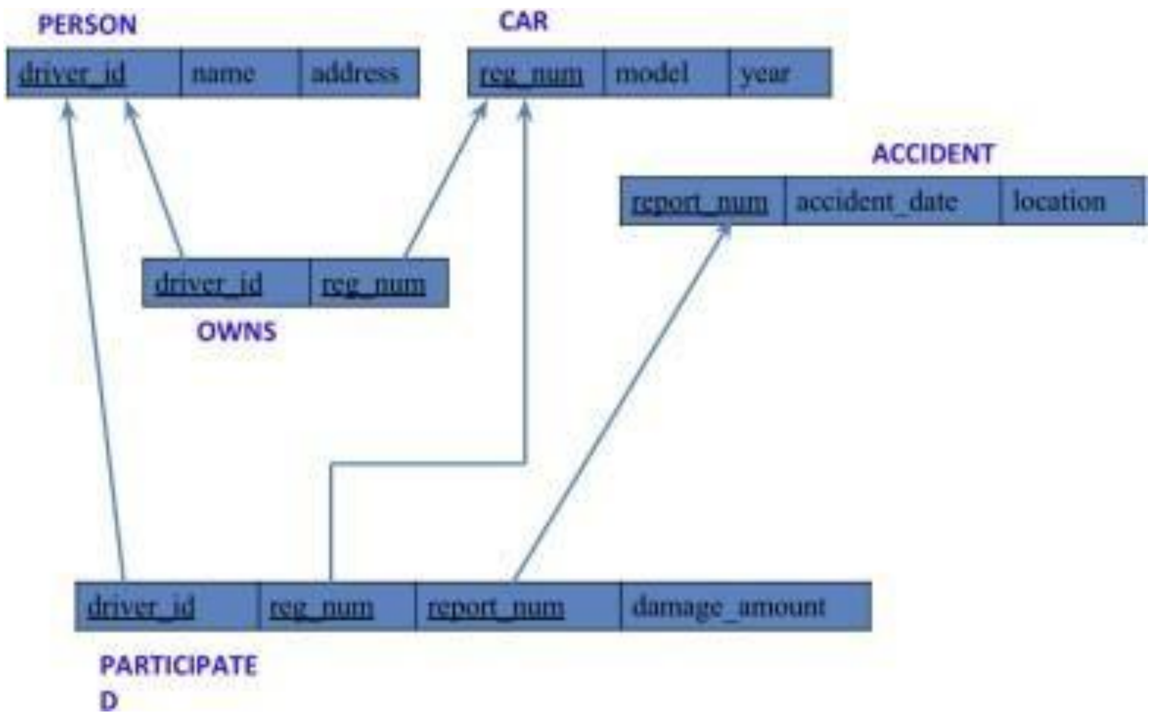
- Add a new accident to the database.

- To Do

- Display Accident date and location

-Display driver id who did accident with damage amount greater than or equal to Rs.2500

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(20),

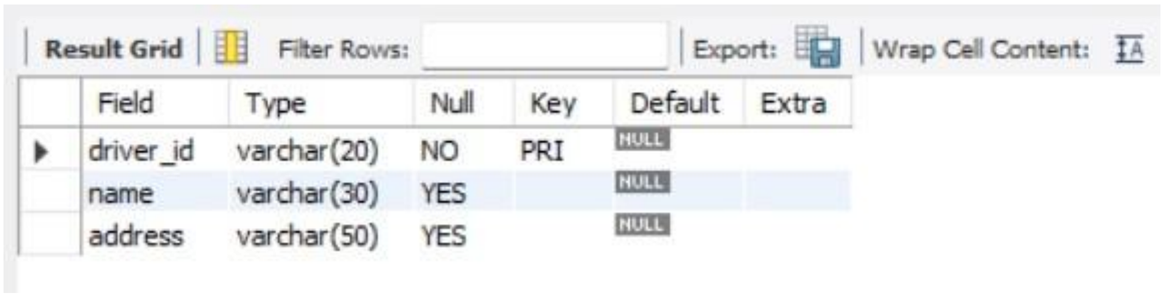
model varchar(20), year
int,

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)
);
```

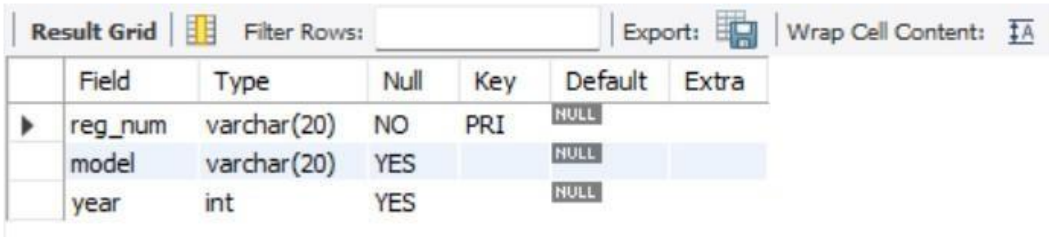
Structure of the table

desc person;


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

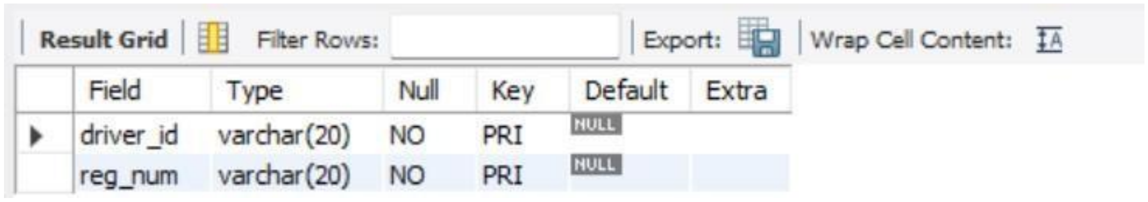
Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
address	varchar(50)	YES		NULL	

desc car;


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

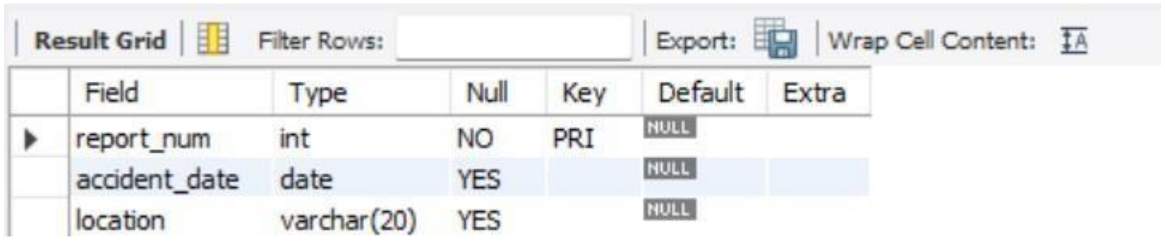
Field	Type	Null	Key	Default	Extra
reg_num	varchar(20)	NO	PRI	NULL	
model	varchar(20)	YES		NULL	
year	int	YES		NULL	

desc owns;


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

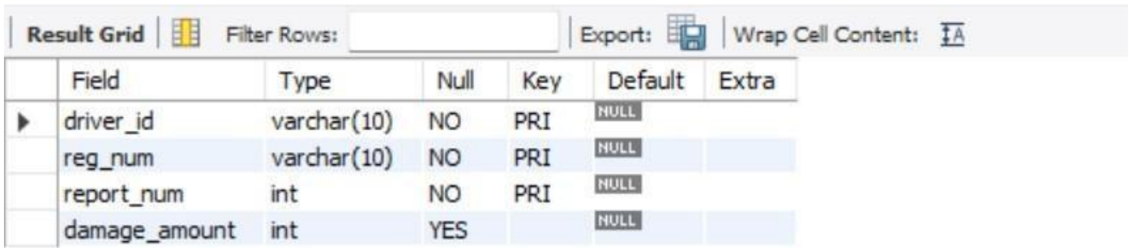
Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(20)	NO	PRI	NULL	

desc accident;


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

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

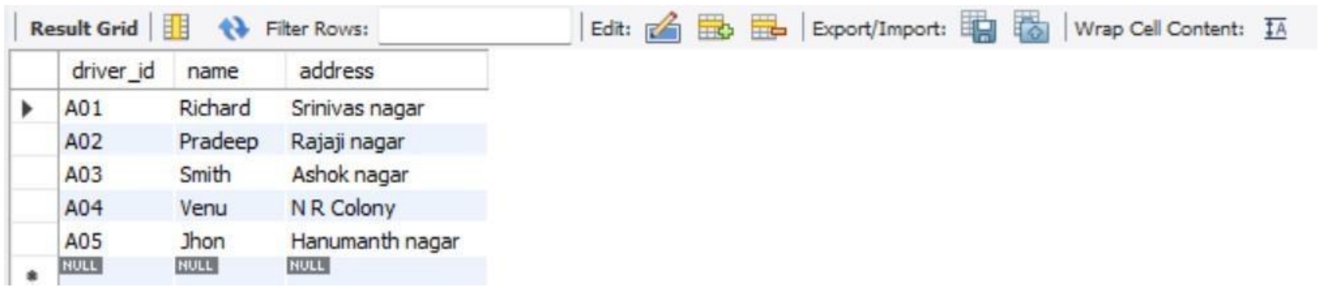
desc participated;


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

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

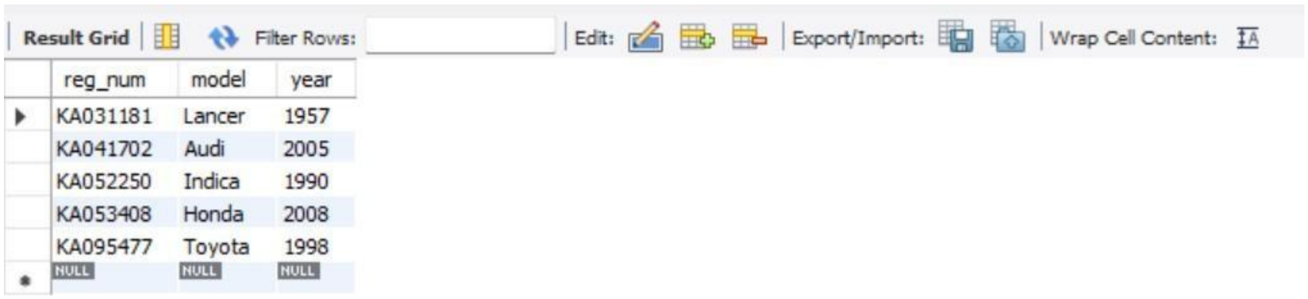
Inserting Values to the table

```
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;
```



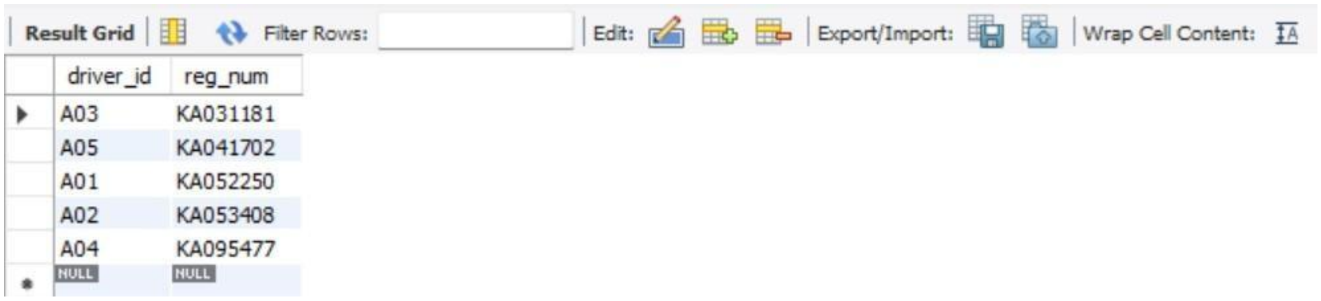
driver_id	name	address
A01	Richard	Srinivas nagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	N R Colony
A05	Jhon	Hanumanth nagar
NULL	NULL	NULL

```
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;
```



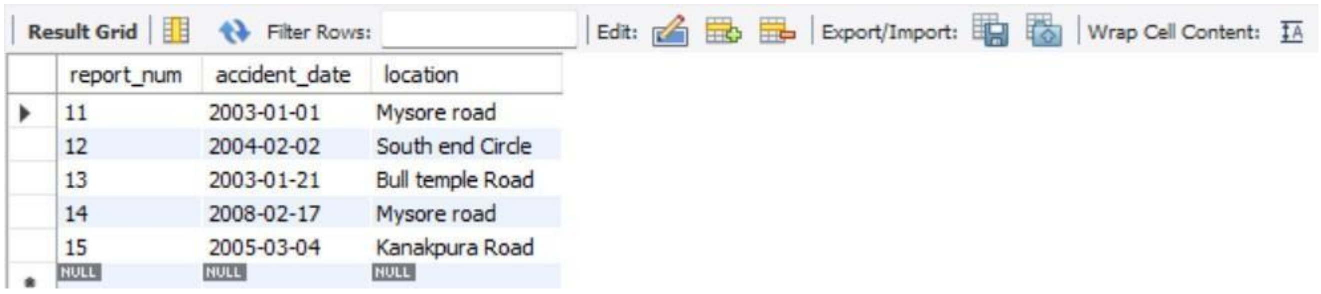
reg_num	model	year
KA031181	Lancer	1957
KA041702	Audi	2005
KA052250	Indica	1990
KA053408	Honda	2008
KA095477	Toyota	1998
NULL	NULL	NULL

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



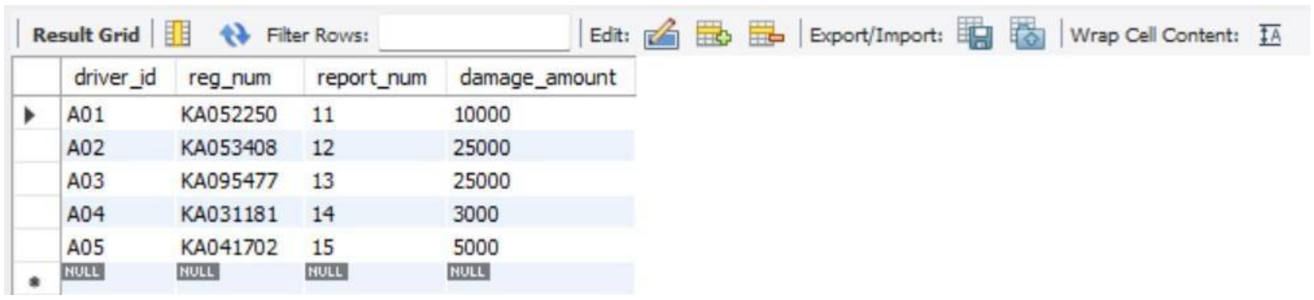
driver_id	reg_num
A03	KA031181
A05	KA041702
A01	KA052250
A02	KA053408
A04	KA095477
NULL	NULL

```
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, "2005-03-04", "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	2005-03-04	Kanakpura Road
NULL	NULL	NULL

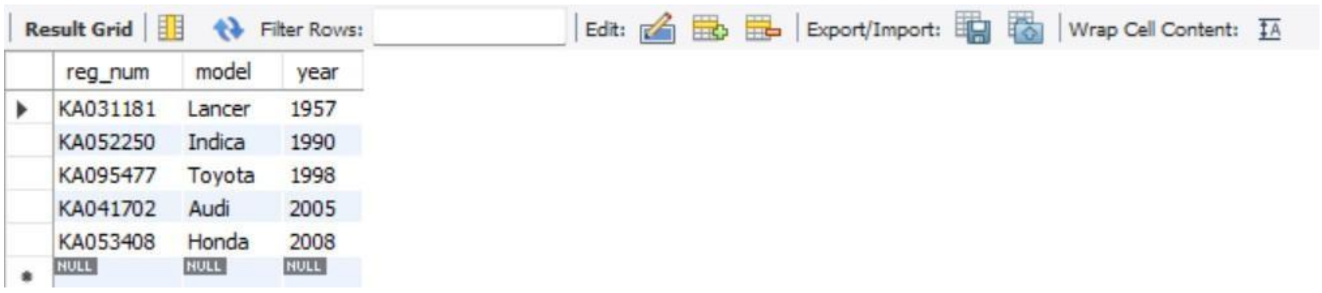
```
insert into participated values("A01","KA052250",11,10000);
insert into participated values("A02","KA053408",12,50000);
insert into participated values("A03","KA095477",13,25000);
insert into participated values("A04","KA031181",14,3000);
insert into participated values("A05","KA041702",15,5000);
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

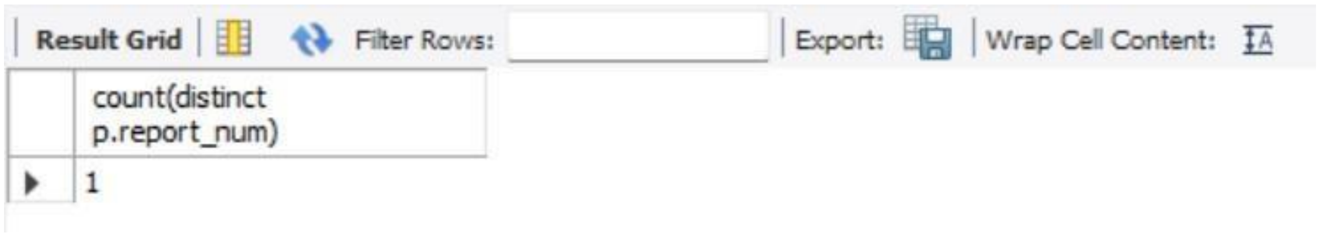
Queries:

-Display the entire CAR relation in the ascending order of manufacturing year.
select* from CAR order by year asc;



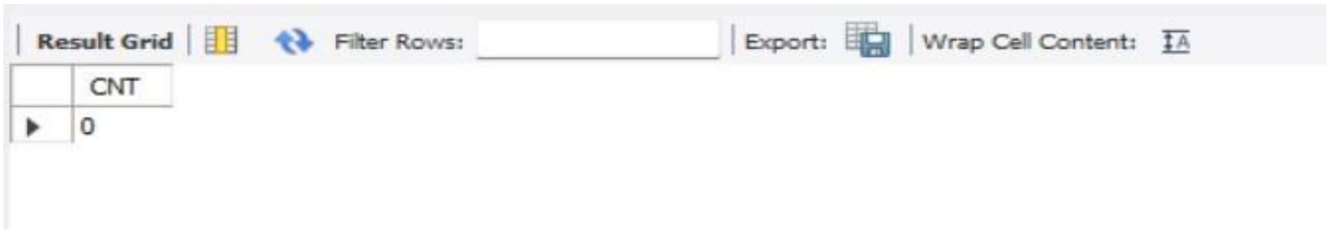
reg_num	model	year
KA031181	Lancer	1957
KA052250	Indica	1990
KA095477	Toyota	1998
KA041702	Audi	2005
KA053408	Honda	2008
NULL	NULL	NULL

-Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
select count(distinct p.report_num) from participated p join car c on p.reg_num = c.reg_num where c.model = 'lancer';



count(distinct p.report_num)
1

-Find the total number of people who owned cars that involved in accidents in 2008.
select count(distinct driver_id) CNT from participated p, accident a where p.report_num=a.report_num and a.accident_date like "_08%";



CNT
0

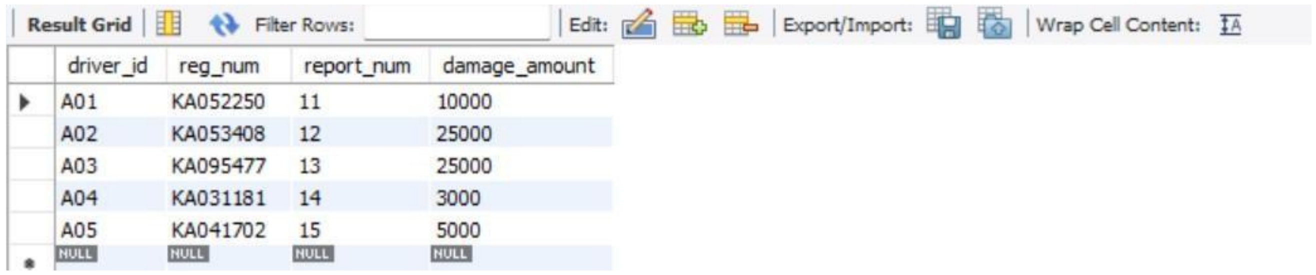
Experiment 2: More Queries on Insurance Database

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;

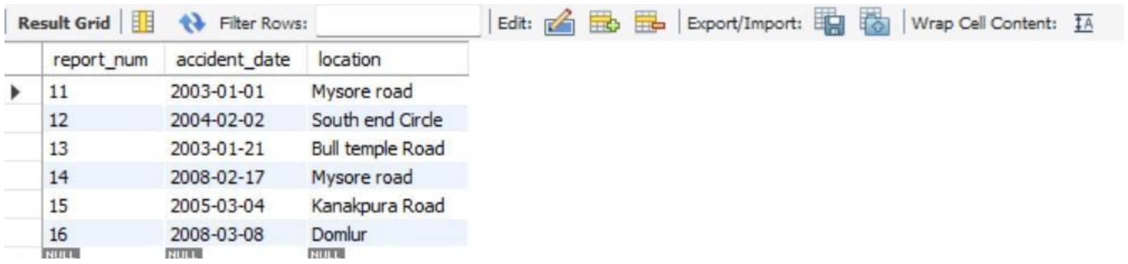
select * from participated;



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

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

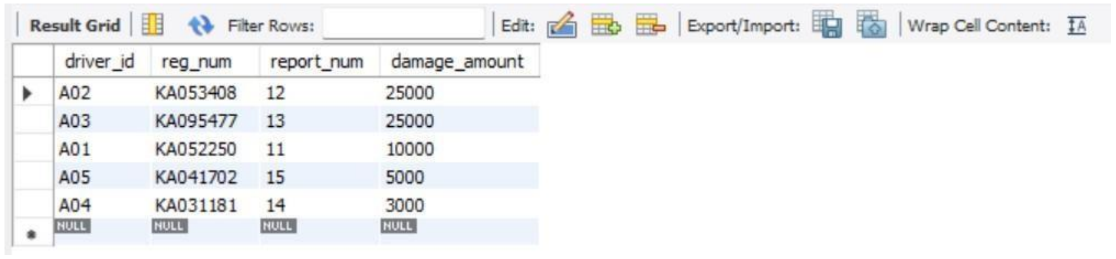
-Add a new accident to the database.
insert into accident values(16,'2008-03-08',"Domlur");
select * from accident;



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

report_num	accident_date	location
11	2003-01-01	Mysore road
12	2004-02-02	South end Circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore road
15	2005-03-04	Kanakpura Road
16	2008-03-08	Domlur
NULL	NULL	NULL

- List the entire participated relation in the descending order of damage amount.
select * from participated order by damage_amount desc;

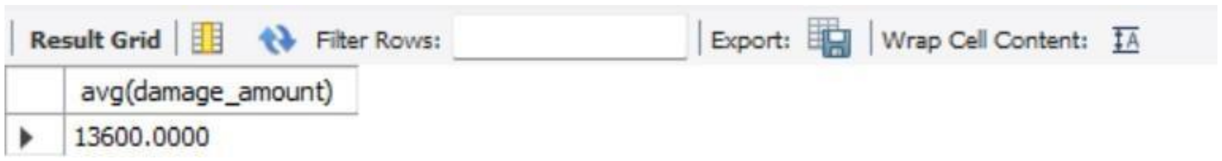


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

-Find the average damage amount.

select avg(damage_amount) from participated;

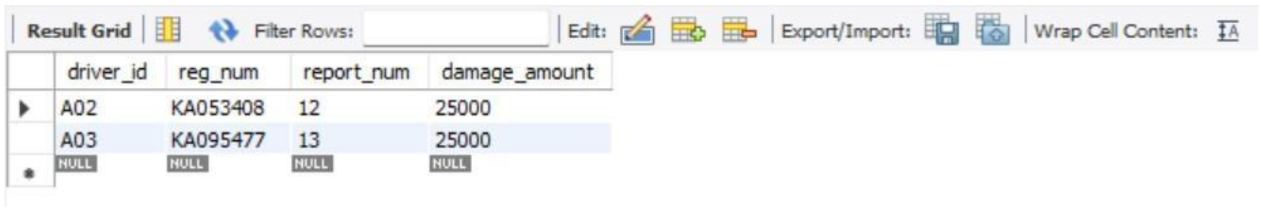


	avg(damage_amount)
▶	13600.0000

-Delete the tuple whose damage amount is below the average damage amount

delete from participated where damage_amount < (select avg_damage from (select avg(damage_amount) as avg_damage from participated)as t);

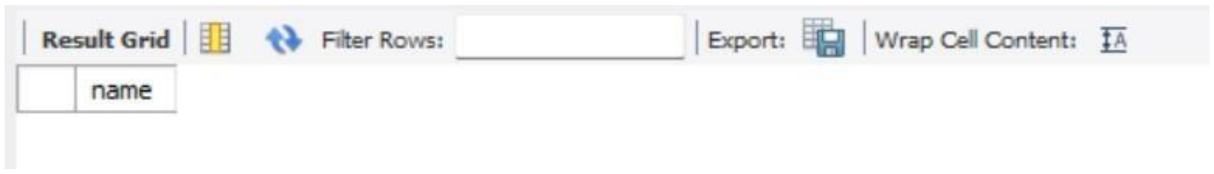
select * from participated;



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

-List the name of drivers whose damage is greater than the average damage amount

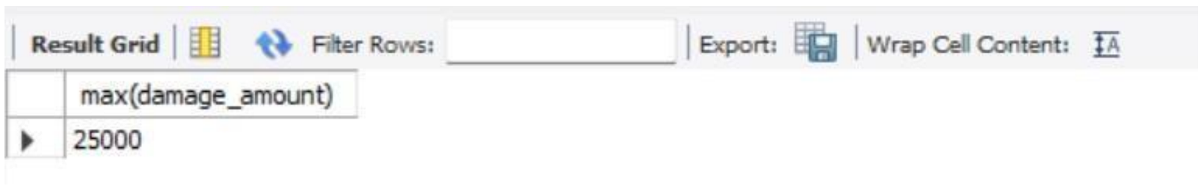
select name from person a, participated b where a.driver_id = b.driver_id and damage_amount > (select avg(damage_amount) from participated);



	name

-Find maximum damage amount.

select max(damage_amount) from participated;



	max(damage_amount)
▶	25000

Experiment 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)

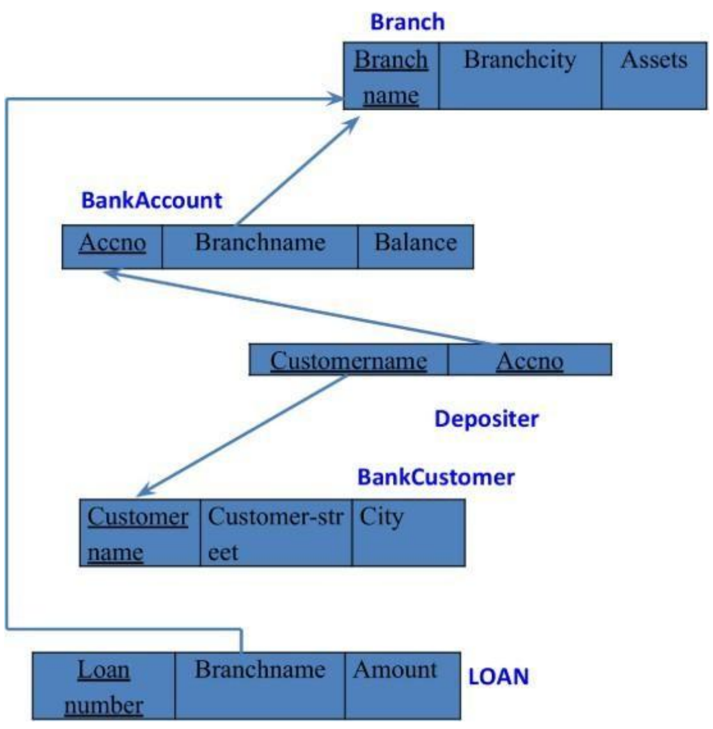
-Depositer(customer-name: String, accno: int)

-Loan(loan-number: int, branch-name: String, amount: real)

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

-Enter at least five tuples for each relation.

Schema Diagram



Create database

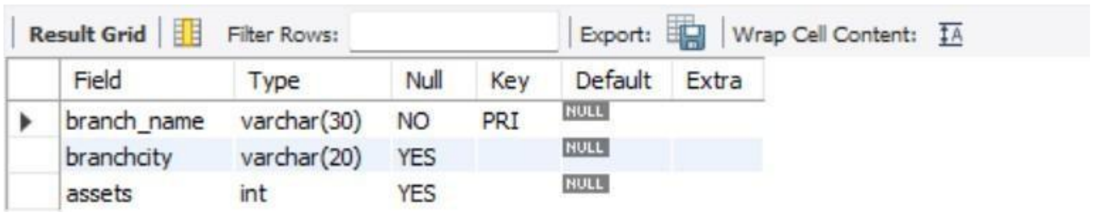
```
create database IF NOT exists bank_database;

use bank_database;
```

Create table

```
create table branch(

branch_name varchar(30),
branchcity varchar(20),
assets int,
 primary key (branch_name));

create table bankaccount(

accno int,
 branch_name varchar(30),
balance int
, primary key (accno),
 foreign key (branch_name) references branch(branch_name));

create table bankcustomer(

customer_name varchar(20),
customer_street varchar(30),
customer_city varchar(30),
primary key(customer_name));

create table depositer(

customer_name varchar(20),
accno int,
primary key(accno),
foreign key(customer_name) references bankcustomer(customer_name),
foreign key(accno) references bankaccount(accno));


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

Structure of the table

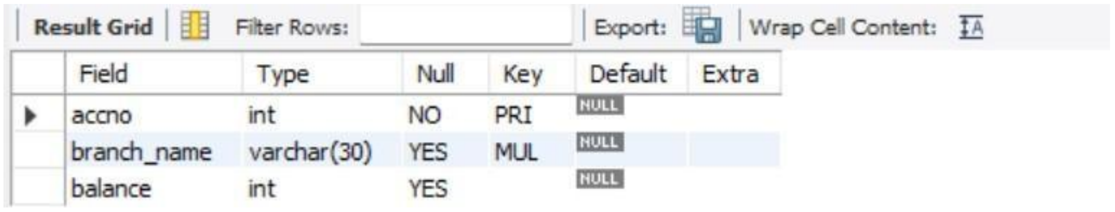
desc branch;



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

Field	Type	Null	Key	Default	Extra
branch_name	varchar(30)	NO	PRI	NULL	
branchcity	varchar(20)	YES		NULL	
assets	int	YES		NULL	

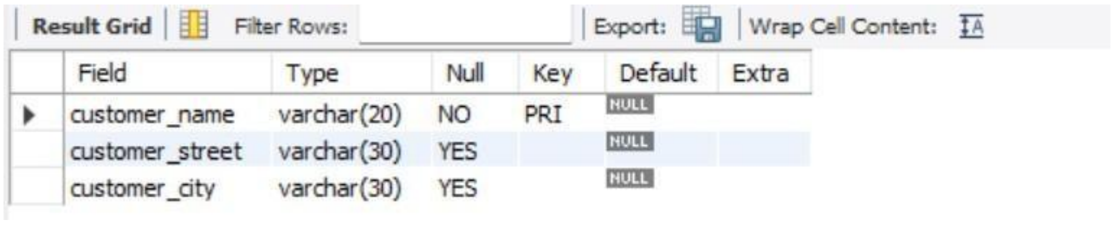
desc bankaccount;



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

Field	Type	Null	Key	Default	Extra
accno	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
balance	int	YES		NULL	

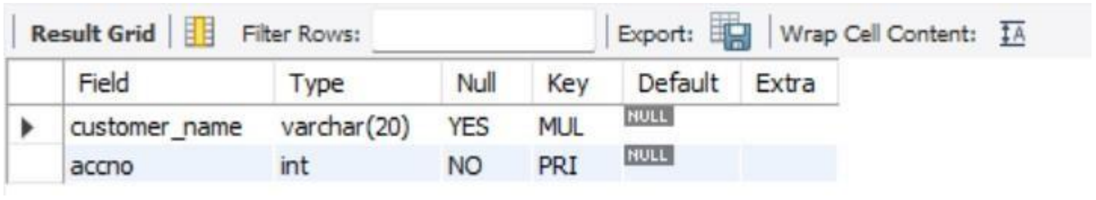
desc bankcustomer;



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

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
customer_street	varchar(30)	YES		NULL	
customer_city	varchar(30)	YES		NULL	

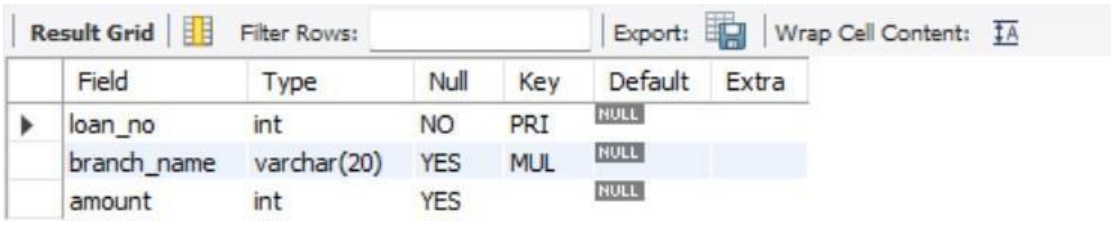
desc depositer;



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

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

desc loan;



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

Field	Type	Null	Key	Default	Extra
loan_no	int	NO	PRI	NULL	
branch_name	varchar(20)	YES	MUL	NULL	
amount	int	YES		NULL	

Inserting Values to the table

insert into branch values("SBI_Chamrajpet","Banglore", 50000);

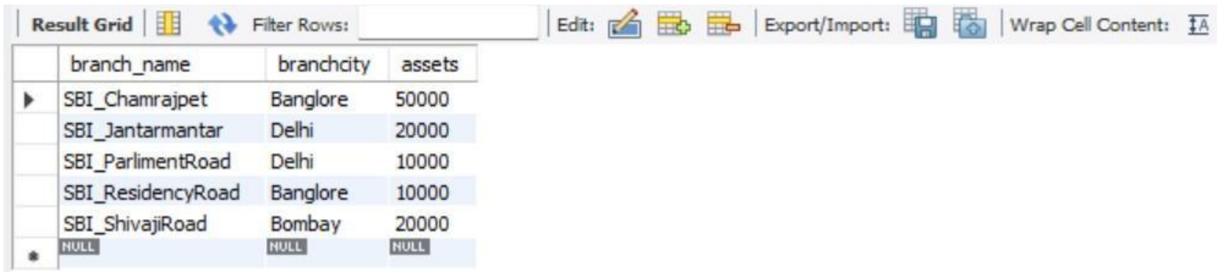
insert into branch values("SBI_ResidencyRoad","Banglore", 10000);

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;



branch_name	branchcity	assets
SBI_Chamrajpet	Banglore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Banglore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

inset 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);

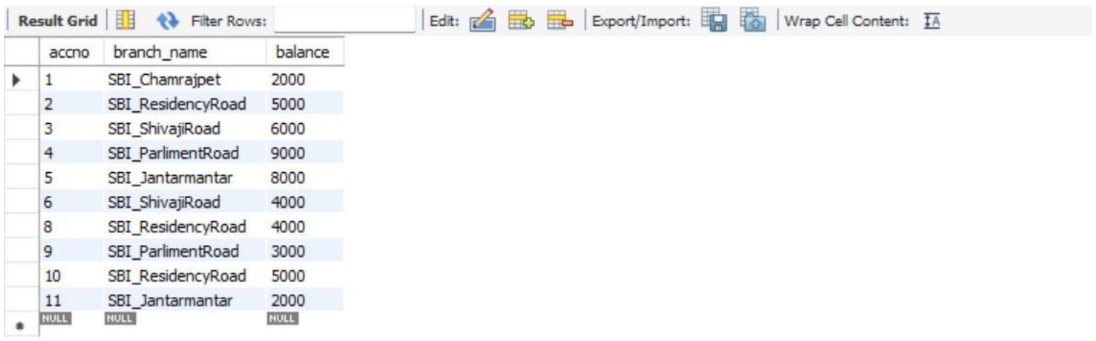
insert into bankaccount values(8, "SBI_ResidencyRoad", 4000);

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", "Banglore");

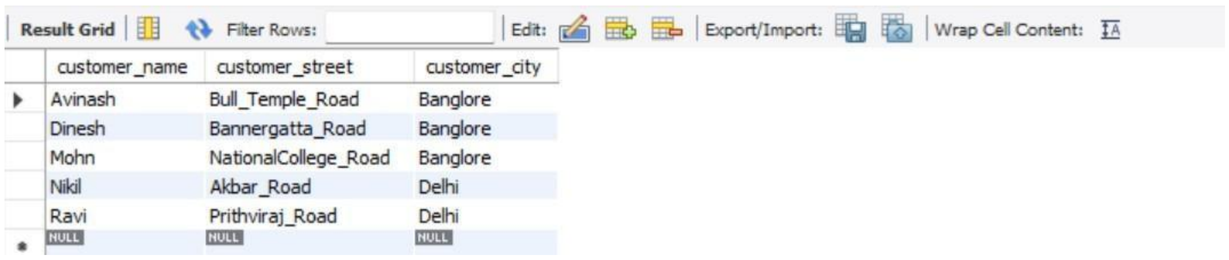
insert into bankcustomer values("Dinesh", "Bannergatta_Road", "Banglore");

insert into bankcustomer values("Mohn", "NationalCollege_Road", "Banglore");

insert into bankcustomer values("Nikil", "Akbar_Road", "Delhi");

insert into bankcustomer values("Ravi", "Prithviraj_Road", "Delhi");

select * from bankcustomer;



customer_name	customer_street	customer_city
Avinash	Bull_Temple_Road	Banglore
Dinesh	Bannergatta_Road	Banglore
Mohn	NationalCollege_Road	Banglore
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);

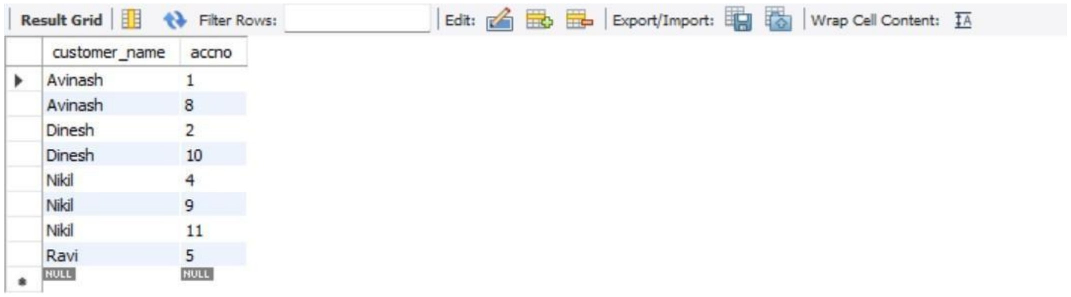
insert into depositer values("Avinash",8);

insert into depositer values("Nikil",9);

insert into depositer values("Dinesh",10);

insert into depositer values("Nikil",11);

select * from depositer;



customer_name	accno
Avinash	1
Avinash	8
Dinesh	2
Dinesh	10
Nikil	4
Nikil	9
Nikil	11
Ravi	5
NULL	NULL

insert into loan values(1,"SBI_Chamrajpet",1000);

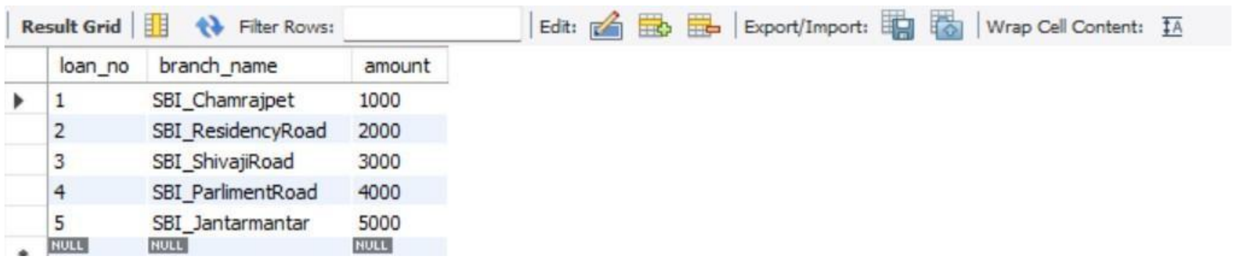
insert into loan values(2,"SBI_ResidencyRoad",2000);

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;

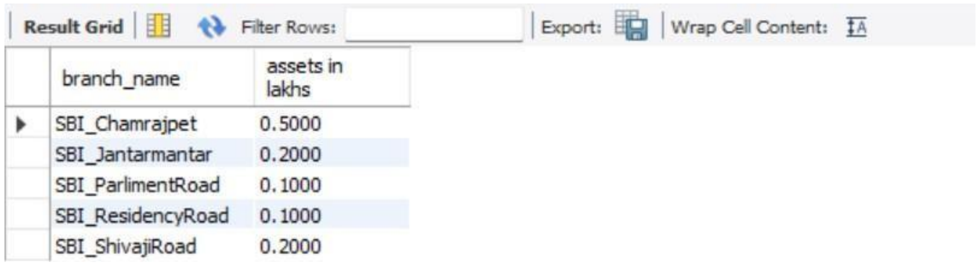


loan_no	branch_name	amount
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_ParlimentRoad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

Queries:

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

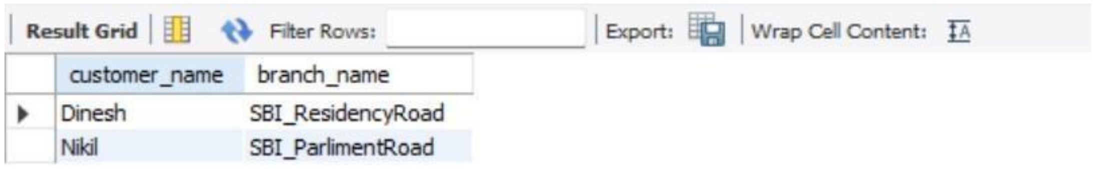
select branch_name,assets/100000 as "assets in lakhs" from branch;



branch_name	assets in lakhs
SBI_Chamrajpet	0.5000
SBI_Jantarmantar	0.2000
SBI_ParlimentRoad	0.1000
SBI_ResidencyRoad	0.1000
SBI_ShivajiRoad	0.2000

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

select d.customer_name, b.branch_name from depositer d join bankaccount b on d.accno = b.accno group by d.customer_name,b.branch_name having count(b.accno) >= 2;

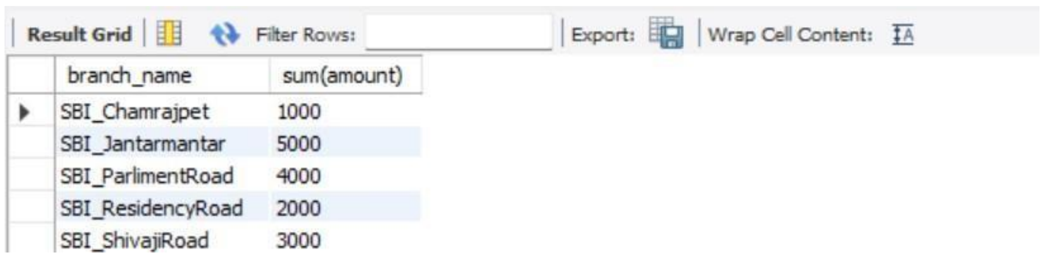


customer_name	branch_name
Dinesh	SBI_ResidencyRoad
Nikil	SBI_ParlimentRoad

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

create view loan_sum as select branch_name,sum(amount) from loan group by branch_name;

select * from loan_sum;



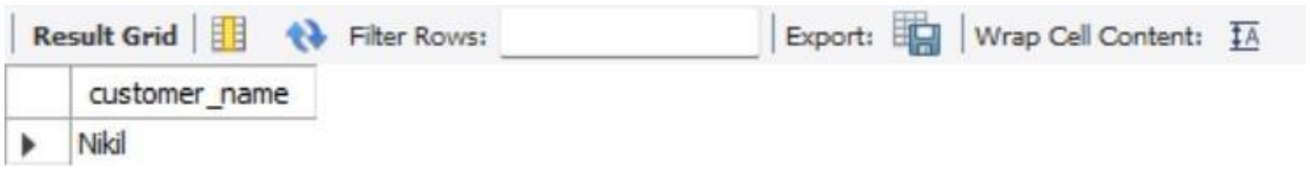
branch_name	sum(amount)
SBI_Chamrajpet	1000
SBI_Jantarmantar	5000
SBI_ParlimentRoad	4000
SBI_ResidencyRoad	2000
SBI_ShivajiRoad	3000

Experiment 4: More Queries on Bank Database

Queries:

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

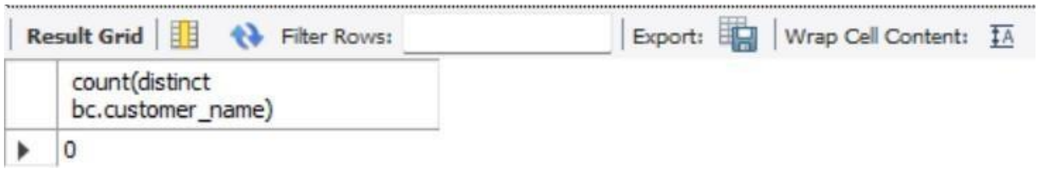
select d.customer_name from depositer d join bankaccount a on d.accno = a.accno join branch b on a.branch_name = b.branch_name where b.branchcity = "Delhi" group by d.customer_name having count(distinct b.branch_name) = (select count(*) from branch where branchcity = "Delhi");



	customer_name
▶	Nikil

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

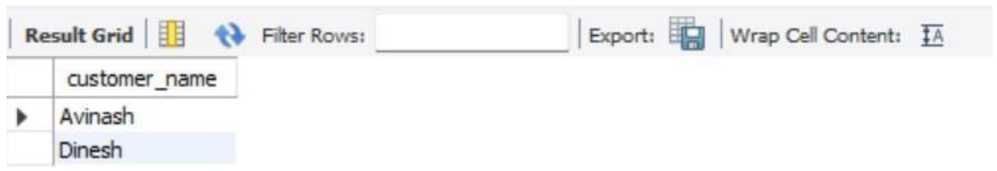
select count(distinct bc.customer_name) from bankcustomer bc join depositer d on bc.customer_name = d.customer_name join bankaccount ba on d.accno = ba.accno join branch b on ba.branch_name = b.branch_name join loan l on b.branch_name = l.branch_name where bc.customer_name not in (select customer_name from depositer);



	count(distinct bc.customer_name)
▶	0

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

select distinct d.customer_name from depositer d join bankaccount a on d.accno = a.accno join branch b on a.branch_name = b.branch_name join loan l on b.branch_name = l.branch_name where b.branchcity = 'Banglore';

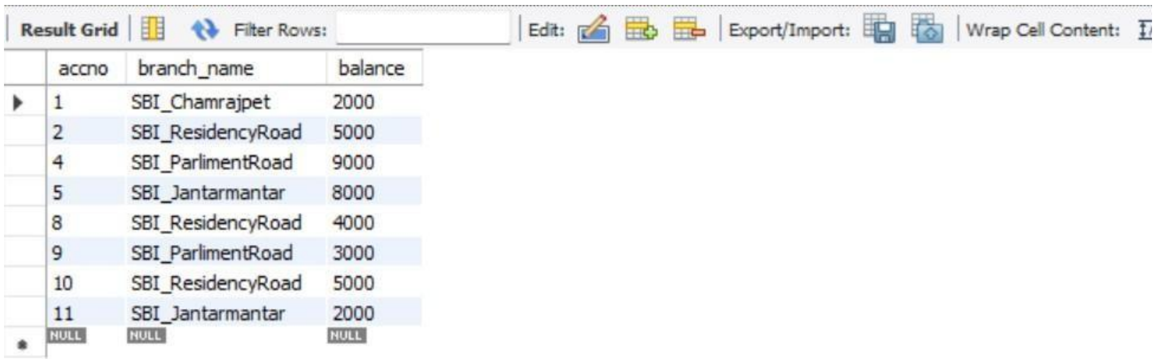


	customer_name
▶	Avinash
	Dinesh

-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 branchcity = 'bombay');

select * from bankaccount;

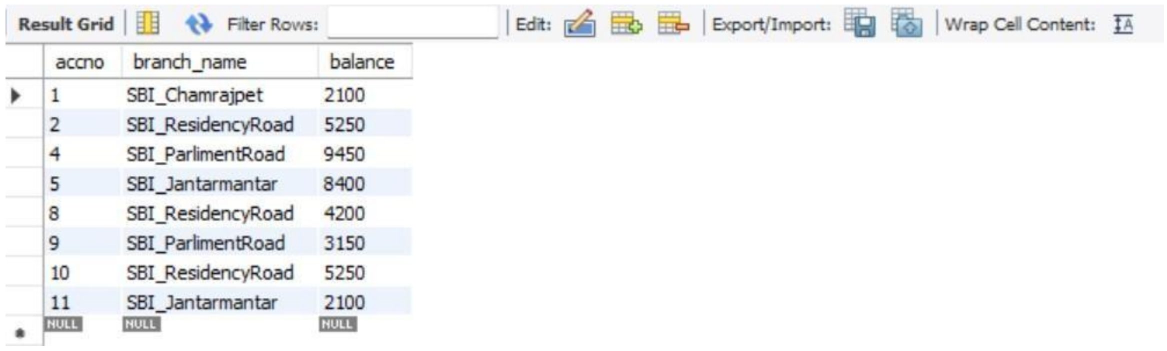


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

-Update the Balance of all accounts by 5%

update bankaccount set balance = balance * 1.05;
select * from bankaccount;

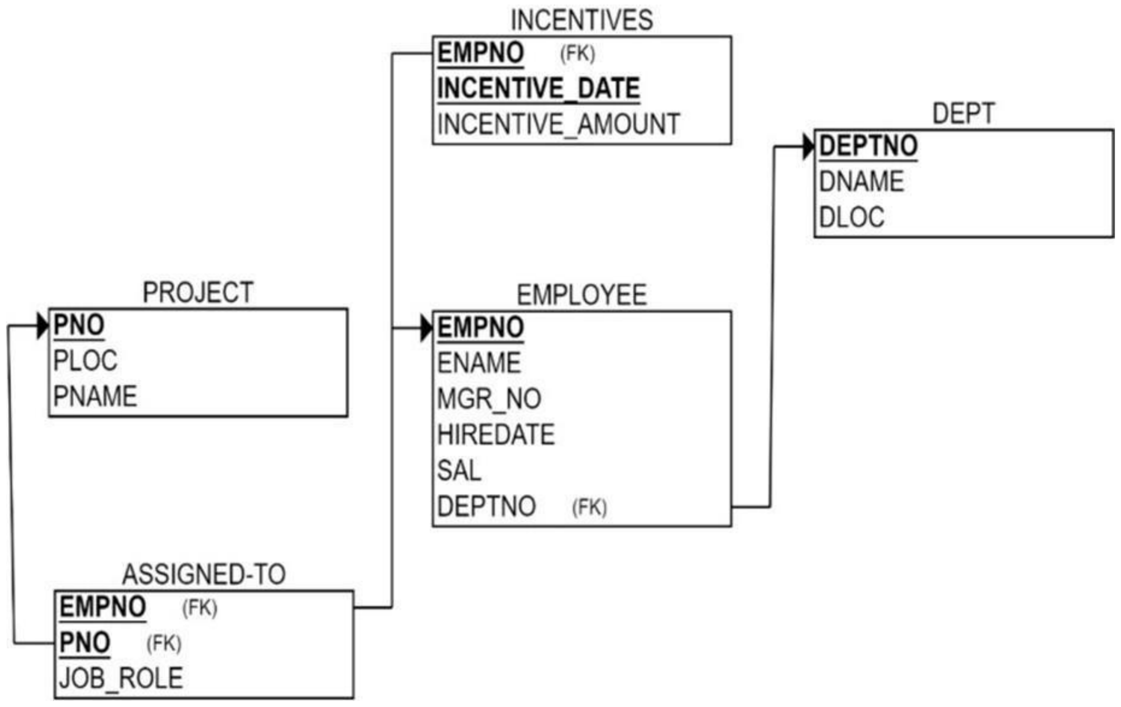


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

accno	branch_name	balance
1	SBI_Chamrajpet	2100
2	SBI_ResidencyRoad	5250
4	SBI_ParlimentRoad	9450
5	SBI_Jantarmantar	8400
8	SBI_ResidencyRoad	4200
9	SBI_ParlimentRoad	3150
10	SBI_ResidencyRoad	5250
11	SBI_Jantarmantar	2100
NULL	NULL	NULL

Experiment 5: Employee Database

Schema Diagram



Create database

create database IF NOT exists employee_database;

use employee_database;

Create table

create table project(pno int, ploc varchar(20), pname varchar(20), primary key(pno));

create table dept(deptno int, dname varchar(20), dloc varchar(20), primary key(deptno));

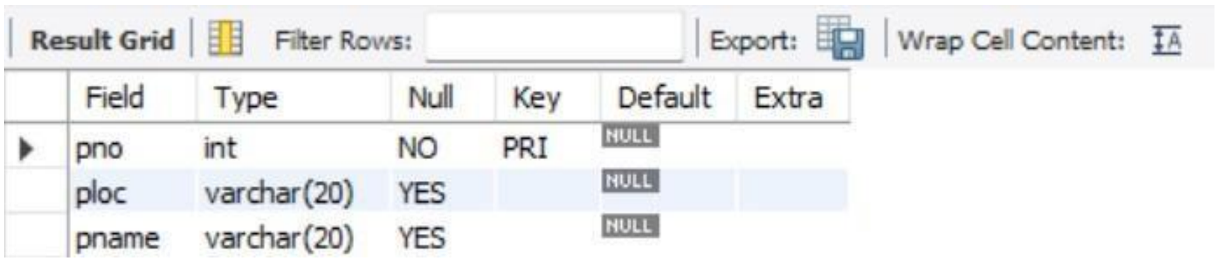
create table employee(empno int, ename varchar(20), mgr_no int, hiredate date, sal float, deptno int, primary key (empno),foreign key(deptno) references dept(deptno));

create table incentives(empno int,incentive_date date, incentive_amount int, primary key(incentive_date), foreign key(empno) references employee(empno));

create table assignedto(empno int,pno int, job_role varchar(20),foreign key (empno) references employee(empno),foreign key (pno) references project(pno));

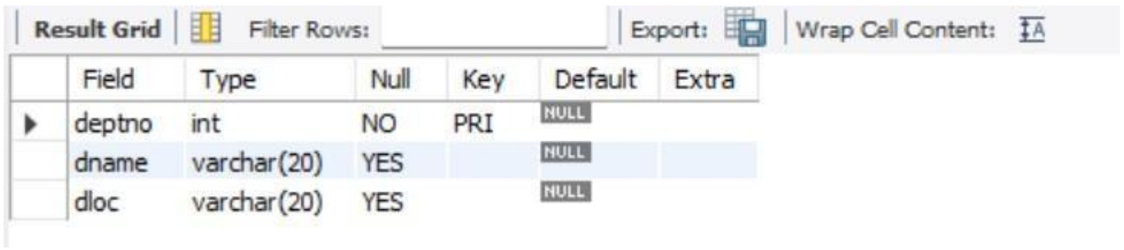
Structure of the table

desc project;



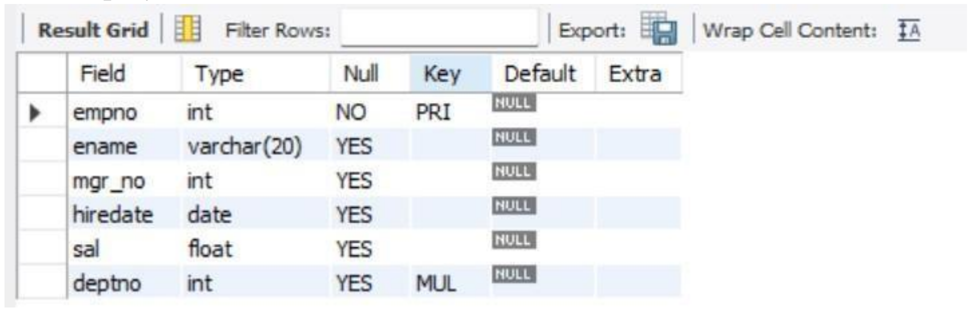
Field	Type	Null	Key	Default	Extra
pno	int	NO	PRI	NULL	
ploc	varchar(20)	YES		NULL	
pname	varchar(20)	YES		NULL	

desc dept;



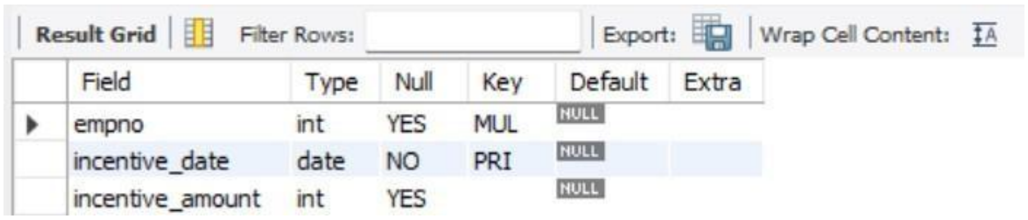
Field	Type	Null	Key	Default	Extra
deptno	int	NO	PRI	NULL	
dname	varchar(20)	YES		NULL	
dloc	varchar(20)	YES		NULL	

desc employee;



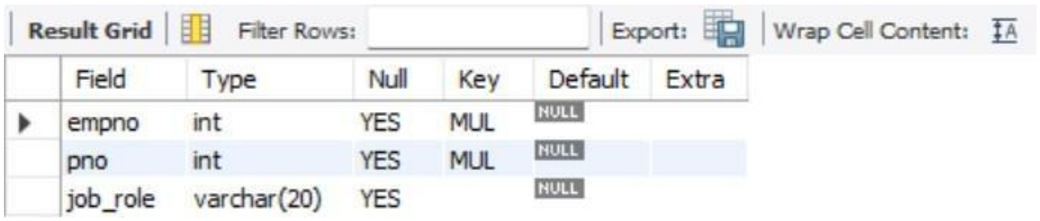
	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	ename	varchar(20)	YES		NULL	
	mgr_no	int	YES		NULL	
	hiredate	date	YES		NULL	
	sal	float	YES		NULL	
	deptno	int	YES	MUL	NULL	

desc incentives;



	Field	Type	Null	Key	Default	Extra
▶	empno	int	YES	MUL	NULL	
	incentive_date	date	NO	PRI	NULL	
	incentive_amount	int	YES		NULL	

desc assignedto;



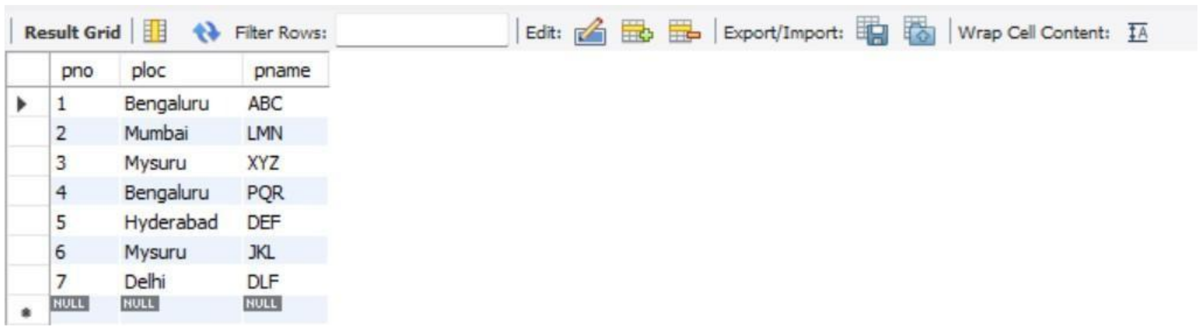
	Field	Type	Null	Key	Default	Extra
▶	empno	int	YES	MUL	NULL	
	pno	int	YES	MUL	NULL	
	job_role	varchar(20)	YES		NULL	

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

-Enter greater than five tuples for each table.

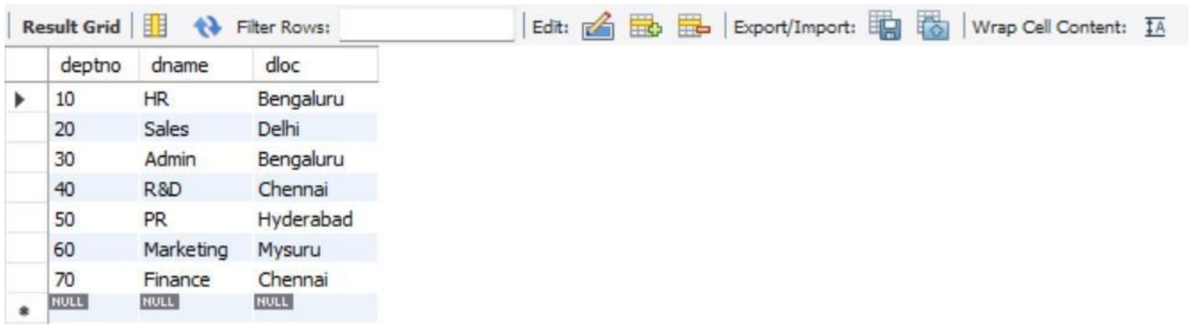
Inserting Values to the table

```
insert into project values(1,"Bengaluru", "ABC");
insert into project values(2,"Mumbai", "LMN");
insert into project values(3,"Mysuru", "XYZ");
insert into project values(4,"Bengaluru", "PQR");
insert into project values(5,"Hyderabad", "DEF");
insert into project values(6,"Mysuru", "JKL");
insert into project values(7,"Delhi", "DLF");
select * from project;
```



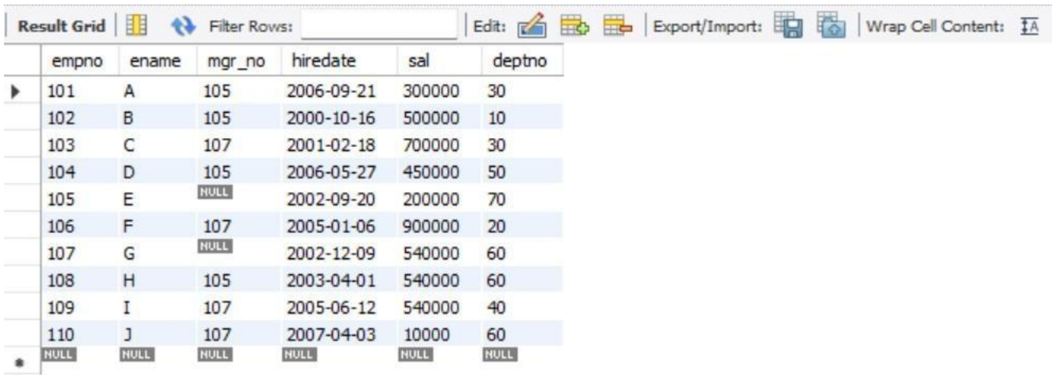
pno	ploc	pname
1	Bengaluru	ABC
2	Mumbai	LMN
3	Mysuru	XYZ
4	Bengaluru	PQR
5	Hyderabad	DEF
6	Mysuru	JKL
7	Delhi	DLF
NULL	NULL	NULL

```
insert into dept values(10,"HR","Bengaluru");
insert into dept values(20,"Sales","Delhi");
insert into dept values(30,"Admin","Bengaluru");
insert into dept values(40,"R&D","Chennai");
insert into dept values(50,"PR","Hyderabad");
insert into dept values(60,"Marketing","Mysuru");
insert into dept values(70,"Finance","Chennai");
select * from dept;
```



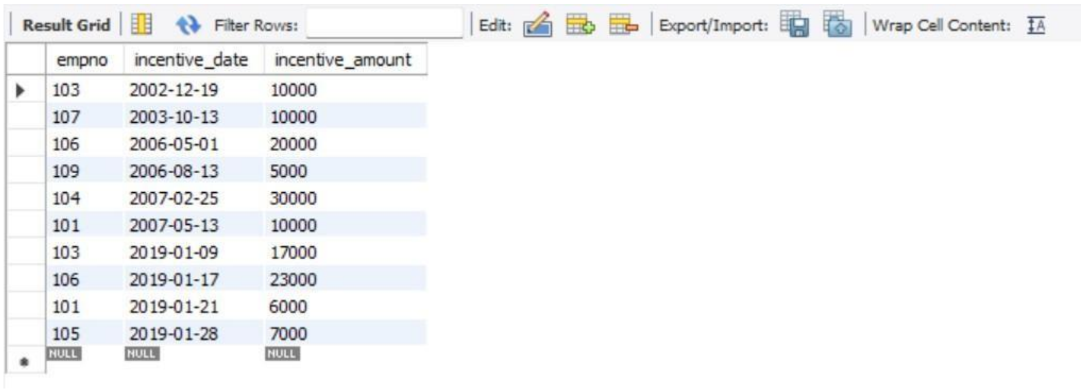
deptno	dname	dloc
10	HR	Bengaluru
20	Sales	Delhi
30	Admin	Bengaluru
40	R&D	Chennai
50	PR	Hyderabad
60	Marketing	Mysuru
70	Finance	Chennai
NULL	NULL	NULL

```
insert into employee values(101,"A",105,"2006-09-21",300000,30);
insert into employee values(102,"B",105,"2000-10-16",500000,10);
insert into employee values(103,"C",107,"2001-02-18",700000,30);
insert into employee values(104,"D",105,"2006-05-27",450000,50);
insert into employee values(105,"E",NULL,"2002-09-20",200000,70);
insert into employee values(106,"F",107,"2005-01-06",900000,20);
insert into employee values(107,"G",NULL,"2002-12-09",540000,60);
insert into employee values(108,"H",105,"2003-04-01",540000,60);
insert into employee values(109,"I",107,"2005-06-12",540000,40);
insert into employee values(110,"J",107,"2007-04-03",10000,60);
select * from employee;
```


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

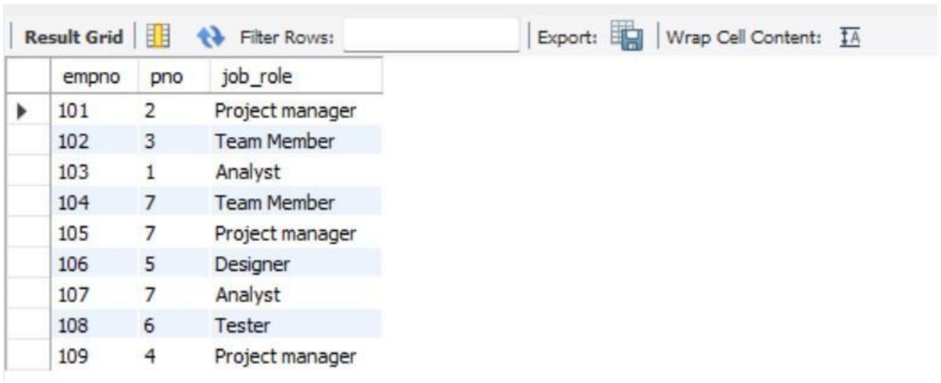
empno	ename	mgr_no	hiredate	sal	deptno
101	A	105	2006-09-21	300000	30
102	B	105	2000-10-16	500000	10
103	C	107	2001-02-18	700000	30
104	D	105	2006-05-27	450000	50
105	E	NULL	2002-09-20	200000	70
106	F	107	2005-01-06	900000	20
107	G	NULL	2002-12-09	540000	60
108	H	105	2003-04-01	540000	60
109	I	107	2005-06-12	540000	40
110	J	107	2007-04-03	10000	60
NULL	NULL	NULL	NULL	NULL	NULL

insert into incentives values(101,"2007-05-13",10000);
insert into incentives values(103,"2002-12-19",10000);
insert into incentives values(104,"2007-02-25",30000);
insert into incentives values(106,"2006-05-1",20000);
insert into incentives values(107,"2003-10-13",10000);
insert into incentives values(109,"2006-08-13",5000);
insert into incentives values(106,"2019-01-17",23000);
insert into incentives values(103,"2019-01-09",17000);
insert into incentives values(105,"2019-01-28",7000);
insert into incentives values(101,"2019-01-21",6000);
select * from incentives;


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

empno	incentive_date	incentive_amount
103	2002-12-19	10000
107	2003-10-13	10000
106	2006-05-01	20000
109	2006-08-13	5000
104	2007-02-25	30000
101	2007-05-13	10000
103	2019-01-09	17000
106	2019-01-17	23000
101	2019-01-21	6000
105	2019-01-28	7000
NULL	NULL	NULL

insert into assignedto values(101,2,"Project manager");
insert into assignedto values(102,3,"Team Member");
insert into assignedto values(103,1,"Analyst");
insert into assignedto values(104,7,"Team Member");
insert into assignedto values(105,7,"Project manager");
insert into assignedto values(106,5,"Designer");
insert into assignedto values(107,7,"Analyst");
insert into assignedto values(108,6,"Tester");
insert into assignedto values(109,4,"Project manager");
select * from assignedto;

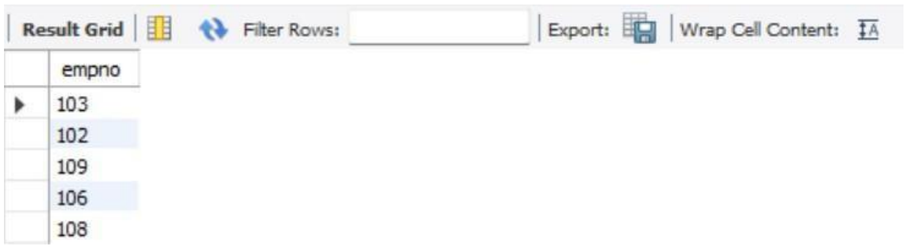


empno	pno	job_role
101	2	Project manager
102	3	Team Member
103	1	Analyst
104	7	Team Member
105	7	Project manager
106	5	Designer
107	7	Analyst
108	6	Tester
109	4	Project manager

Queries:

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

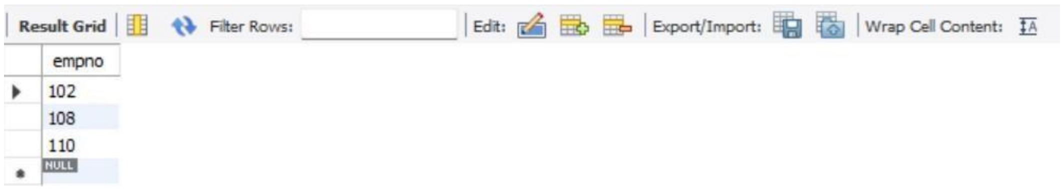
select e.empno from employee e join assignedto a on a.empno=e.empno join project p on a.pno=p.pno where p.ploc in ("Bengaluru","Mysuru","Hyderabad") group by e.empno;



empno
103
102
109
106
108

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

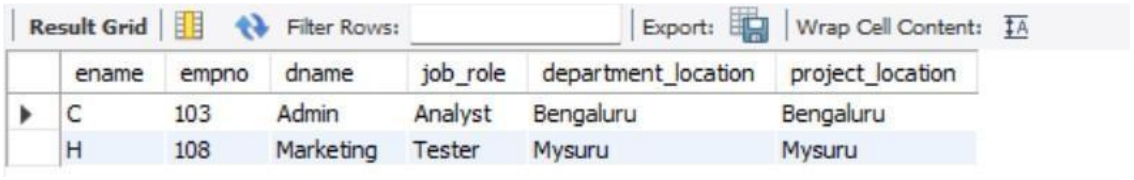
select empno from employee where empno not in(select empno from incentives);



empno
102
108
110
NULL

-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,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 assignedto a on e.empno=a.empno join project p on a.pno=p.pno where d.dloc=p.ploc;



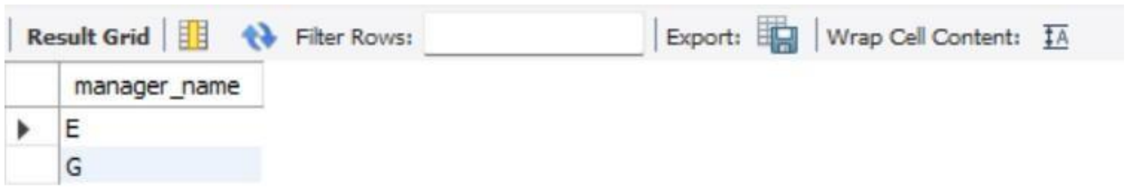
ename	empno	dname	job_role	department_location	project_location
C	103	Admin	Analyst	Bengaluru	Bengaluru
H	108	Marketing	Tester	Mysuru	Mysuru

Experiment 6: More Queries on Employee Database

Queries:

-List the name of the managers with the maximum employees

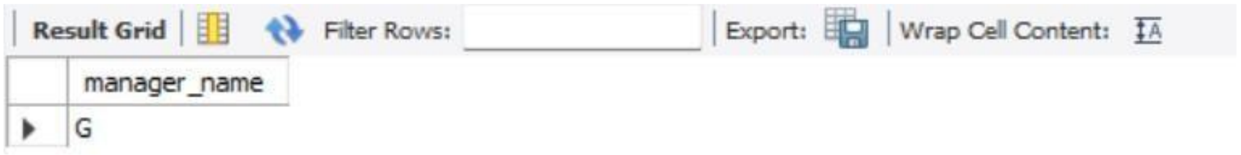
select m.ename as manager_name from employee m join employee e on e.mgr_no=m.empno group by m.empno having count(e.empno)=(select max(emp_count) from (select count(empno) as emp_count from employee where mgr_no is not null group by mgr_no)t);



manager_name
E
G

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

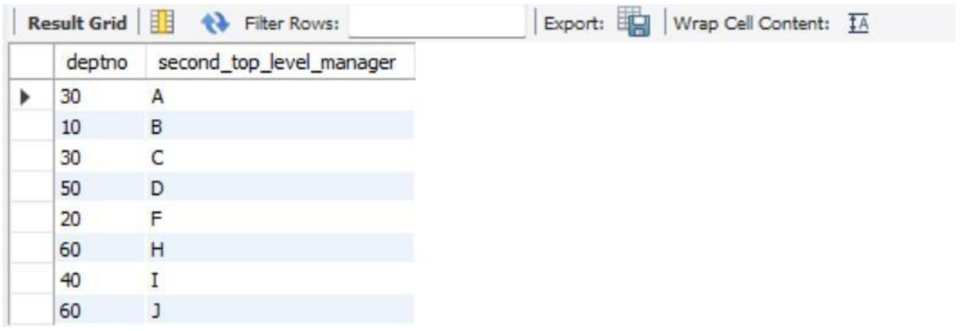
select a.ename as manager_name from employee a join employee b on b.mgr_no=a.empno group by a.empno,a.ename,a.sal having a.sal>avg(b.sal);



manager_name
G

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

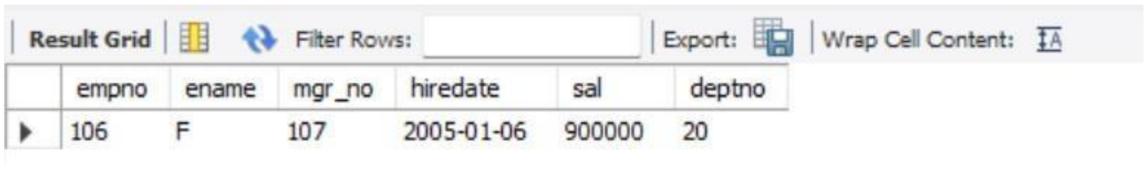
select distinct e.deptno, e.ename as second_top_level_manager from employee e join employee m on e.mgr_no = m.empno where m.mgr_no is null;



deptno	second_top_level_manager
30	A
10	B
30	C
50	D
20	F
60	H
40	I
60	J

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

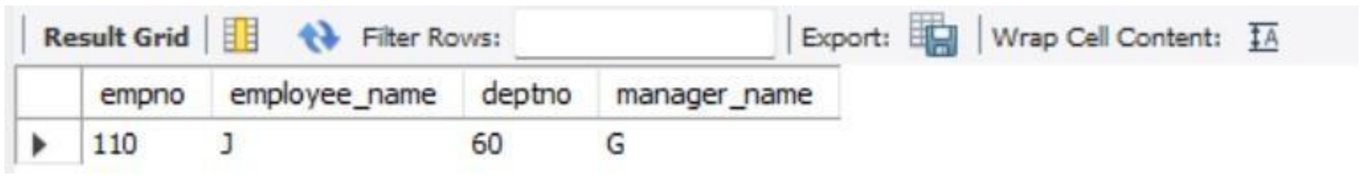
select e.* from employee e join incentives i on e.empno=i.empno where i.incentive_amount=(select incentive_amount from incentives order by incentive_amount desc limit 1 offset 1) and i.incentive_date like "2019-01%";



empno	ename	mgr_no	hiredate	sal	deptno
106	F	107	2005-01-06	900000	20

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

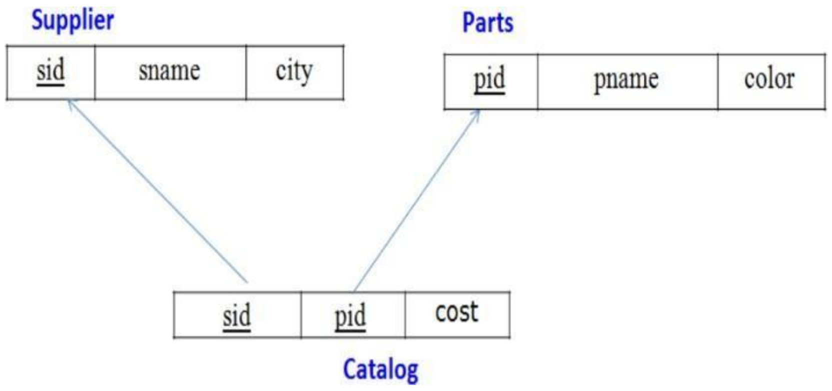
select e.empno, e.ename as employee_name, e.deptno, m.ename as manager_name from employee e join employee m on e.mgr_no = m.empno where e.deptno = m.deptno;



empno	employee_name	deptno	manager_name
110	J	60	G

Experiment 7: Supplier Database

Schema Diagram



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

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

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

Create database

create database IF NOT exists supplier_database;

use supplier_database;

Create table

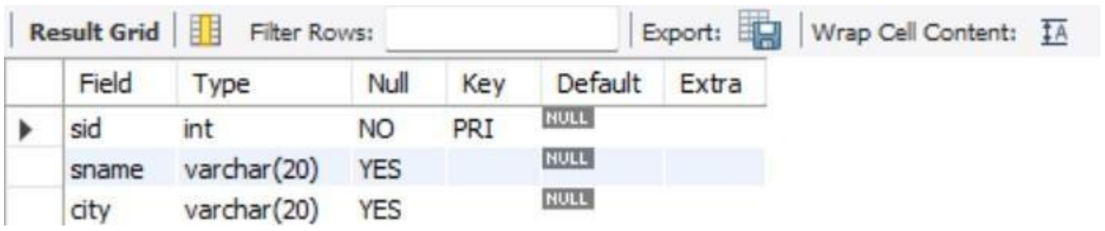
create table suppliers(sid int,sname varchar(20), city varchar(20), primary key(sid));

create table parts(pid int,pname varchar(20), color varchar(20), primary key(pid));

create table catalog(sid int,pid int,cost int,foreign key (sid) references suppliers(sid),foreign key (pid) references parts(pid));

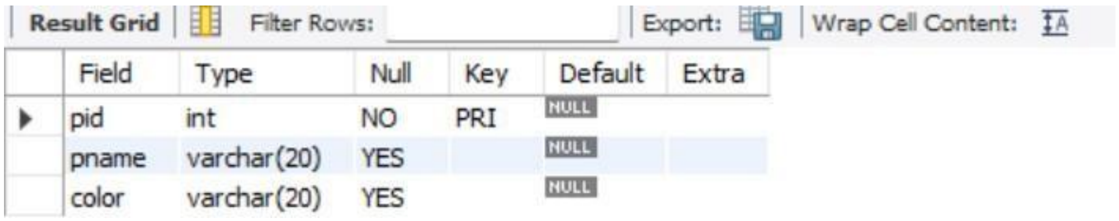
Structure of the table

desc suppliers;



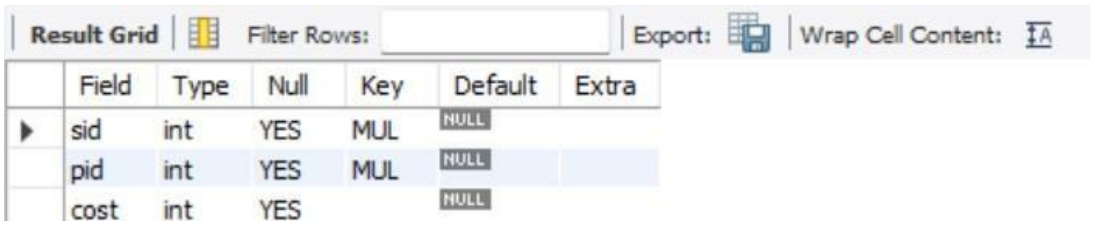
Field	Type	Null	Key	Default	Extra
sid	int	NO	PRI	NULL	
sname	varchar(20)	YES		NULL	
city	varchar(20)	YES		NULL	

desc parts;



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

desc catalog;



Field	Type	Null	Key	Default	Extra
sid	int	YES	MUL	NULL	
pid	int	YES	MUL	NULL	
cost	int	YES		NULL	

-Using Scheme diagram, create tables by properly specifying the primary keys and the foreign keys.

-Insert appropriate records in each table.

Inserting Values to the table

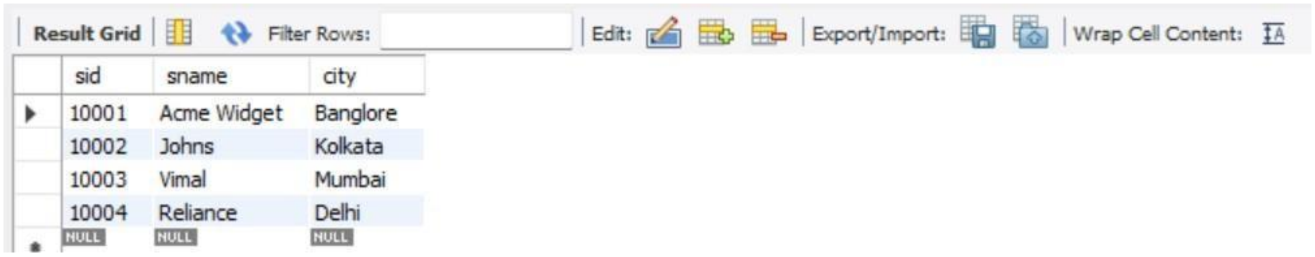
insert into suppliers value(10001,"Acme Widget","Banglore");

insert into suppliers value(10002,"Johns","Kolkata");

insert into suppliers value(10003,"Vimal","Mumbai");

insert into suppliers value(10004,"Reliance","Delhi");

select * from suppliers;



sid	sname	city
10001	Acme Widget	Banglore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi
NULL	NULL	NULL

insert into parts value(20001,"Book","Red");

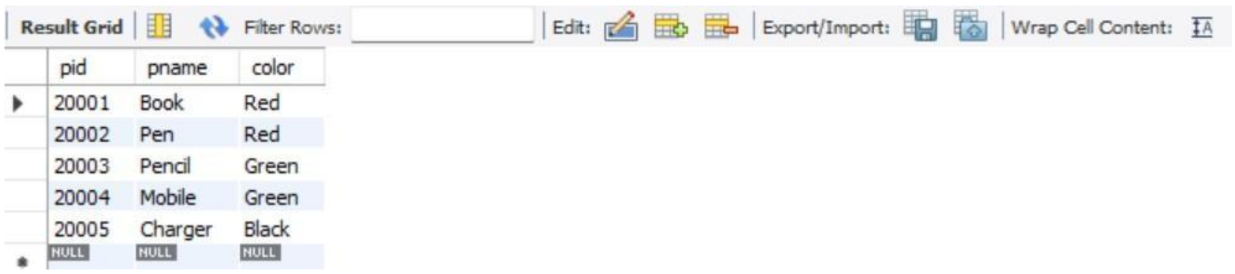
insert into parts value(20002,"Pen","Red");

insert into parts value(20003,"Pencil","Green");

insert into parts value(20004,"Mobile","Green");

insert into parts value(20005,"Charger","Black");

select * from parts;


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

pid	pname	color
20001	Book	Red
20002	Pen	Red
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black
NULL	NULL	NULL

insert into catalog value(10001,20001,10);

insert into catalog value(10001,20001,10);

insert into catalog value(10001,20001,30);

insert into catalog value(10001,20001,10);

insert into catalog value(10001,20001,10);

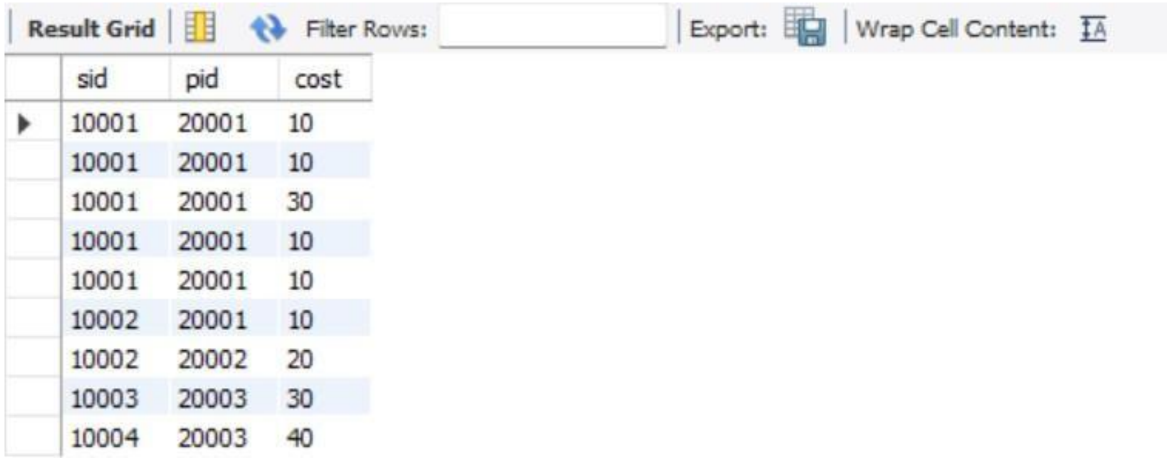
insert into catalog value(10002,20001,10);

insert into catalog value(10002,20002,20);

insert into catalog value(10003,20003,30);

insert into catalog value(10004,20003,40);

select * from catalog;

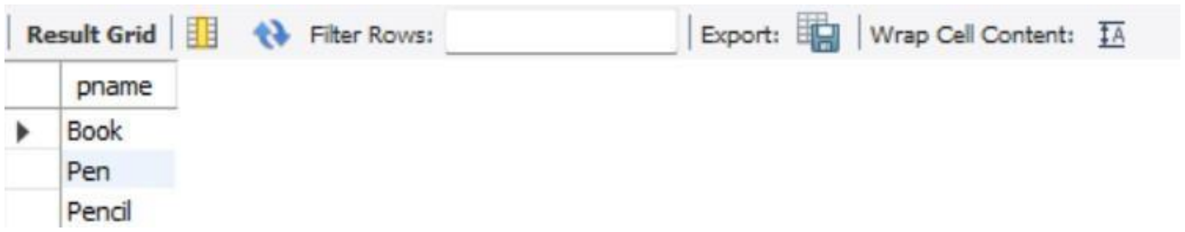

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

sid	pid	cost
10001	20001	10
10001	20001	10
10001	20001	30
10001	20001	10
10001	20001	10
10002	20001	10
10002	20002	20
10003	20003	30
10004	20003	40

Queries:

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

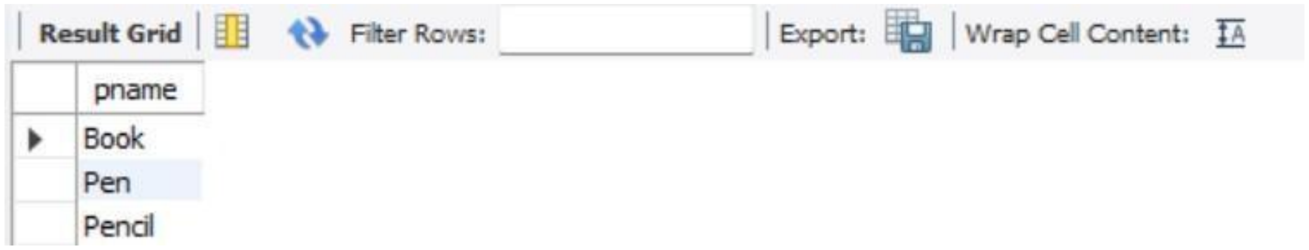
select p.pname from parts p join catalog c on c.pid=p.pid join suppliers s on s.sid=c.sid group by p.pname;



```
pname
Book
Pen
Pencil
```

-Find the snames of suppliers who supply every part.

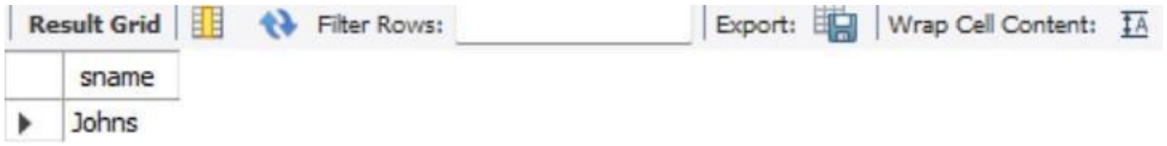
select distinct p.pname from parts p join catalog c on c.pid=p.pid;



```
pname
Book
Pen
Pencil
```

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

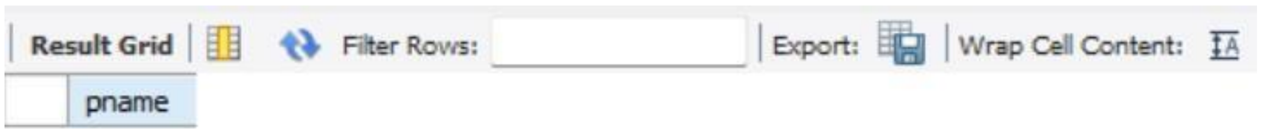
select s.sname from suppliers s join catalog c on s.sid=c.sid join parts p on p.pid=c.pid where color='Red'group by s.sname having count(distinct p.pid) = (select count(*) from parts where color='Red');



```
sname
Johns
```

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

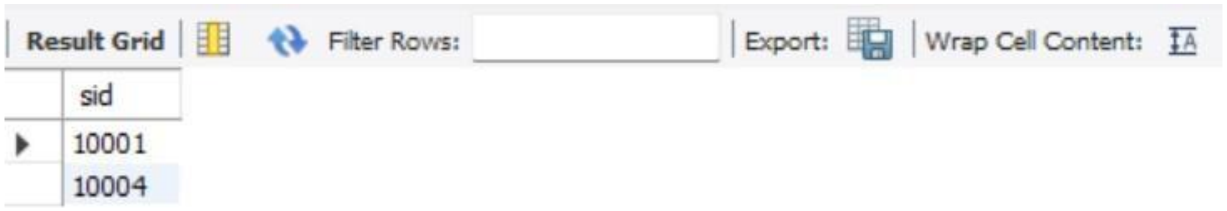
select p.pname from parts p join catalog c on p.pid=c.pid join suppliers s on s.sid=c.sid where s.sname='Acme Widget' and p.pid not in(select c2.pid from catalog c2 join suppliers s2 on s2.sid=c2.sid where s2.sname!='Acme Widget');



pname

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

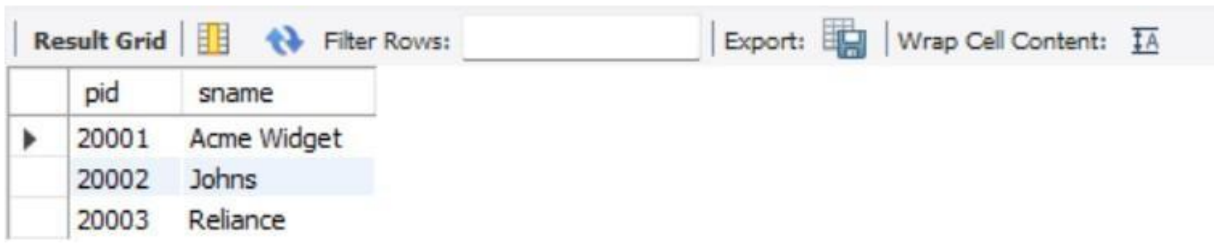
select distinct c.sid from catalog c where c.cost>(select avg(c2.cost) from catalog c2 where c2.pid=c.pid);



sid
10001
10004

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

select p.pid,s.sname from catalog c join suppliers s on s.sid=c.sid join parts p on p.pid=c.pid where c.cost=(select max(c2.cost) from catalog c2 where c2.pid=c.pid);

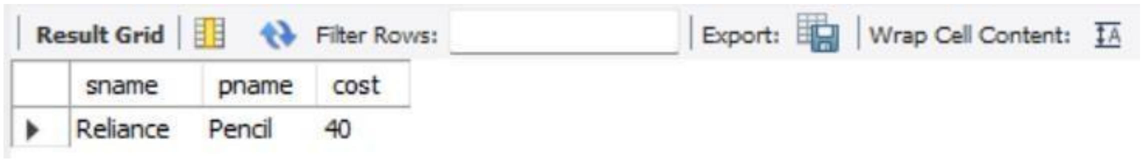


pid	sname
20001	Acme Widget
20002	Johns
20003	Reliance

Experiment 8: More Queries on Supplier Database

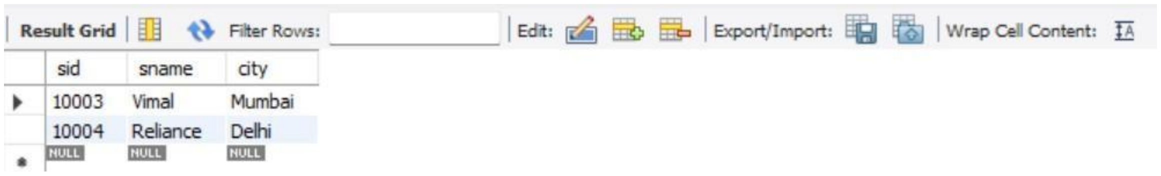
Queries:

-Find the most expensive part overall and the supplier who supplies it.
select s.sname, p.pname, c.cost from catalog c join suppliers s on c.sid = s.sid join parts p on c.pid = p.pid where c.cost = (select max(cost) from catalog);



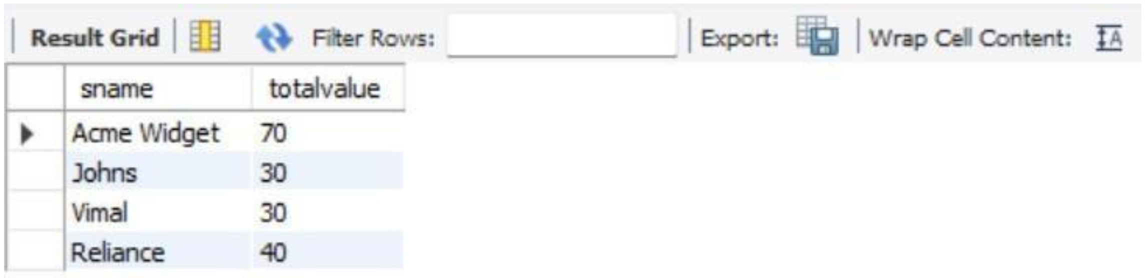
sname	pname	cost
Reliance	Pencil	40

-Find suppliers who do NOT supply any red parts.
select s.* from suppliers s where s.sid not in (select c.sid from catalog c join parts p on c.pid = p.pid where p.color = 'Red');



sid	sname	city
10003	Vimal	Mumbai
10004	Reliance	Delhi
NULL	NULL	NULL

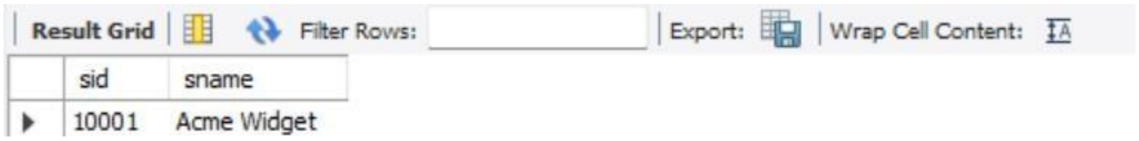
-Show each supplier and total value of all parts they supply.
select s.sname, sum(c.cost) as totalvalue from suppliers s join catalog c on s.sid = c.sid group by s.sid;



sname	totalvalue
Acme Widget	70
Johns	30
Vimal	30
Reliance	40

-Find suppliers who supply at least 2 parts cheaper than ₹20.

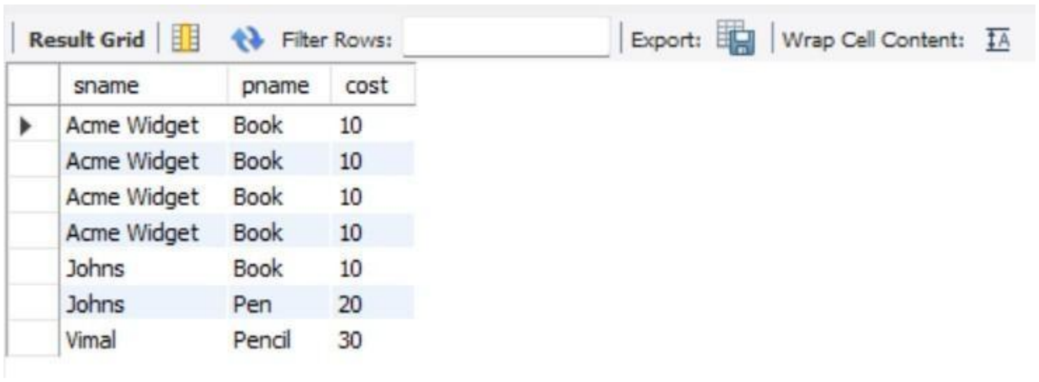
select s.sid, s.sname from suppliers s join catalog c on s.sid = c.sid where c.cost < 20 group by s.sid having count(c.pid) >= 2;



sid	sname
10001	Acme Widget

-List suppliers who offer the cheapest cost for each part

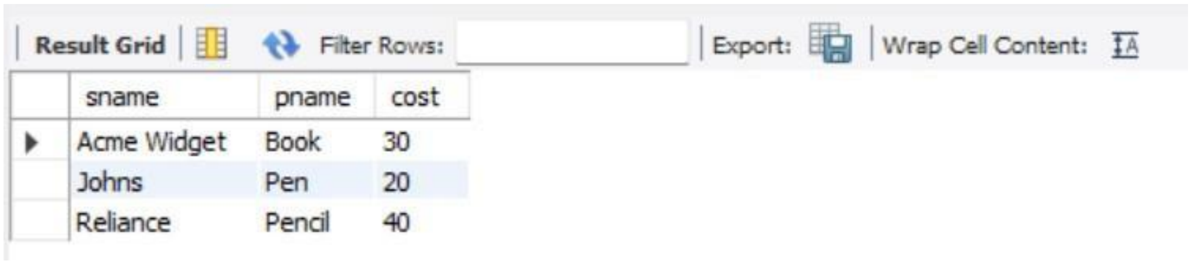
select s.sname, p.pname, c.cost from catalog c join suppliers s on c.sid = s.sid join parts p on c.pid = p.pid where c.cost = (select min(c2.cost) from catalog c2 where c2.pid = c.pid);



sname	pname	cost
Acme Widget	Book	10
Acme Widget	Book	10
Acme Widget	Book	10
Acme Widget	Book	10
Johns	Book	10
Johns	Pen	20
Vimal	Pencil	30

-Create a view of the most expensive supplier for each part.

create view most_expensive_supplier as select s.sname, p.pname, c.cost from catalog c join suppliers s on c.sid = s.sid join parts p on c.pid = p.pid where c.cost = (select max(c2.cost) from catalog c2 where c2.pid = c.pid); select * from most_expensive_supplier;



sname	pname	cost
Acme Widget	Book	30
Johns	Pen	20
Reliance	Pencil	40

-Create a Trigger to prevent inserting a Catalog cost below 1.

DELIMITER //

create trigger prevent_low_cost before insert on catalog for each row begin if new.cost < 1 then signal sqlstate '45000' set message_text ='Cost must be at least 1'; end if; end;

//DELIMITER ;

Experiment 9 : NOSQL- Student Database

Create database “Student” with the following attributes Rollno, Age, ContactNo, Email-Id.

db.createCollection("Student");

Creating table and inserting values

db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});

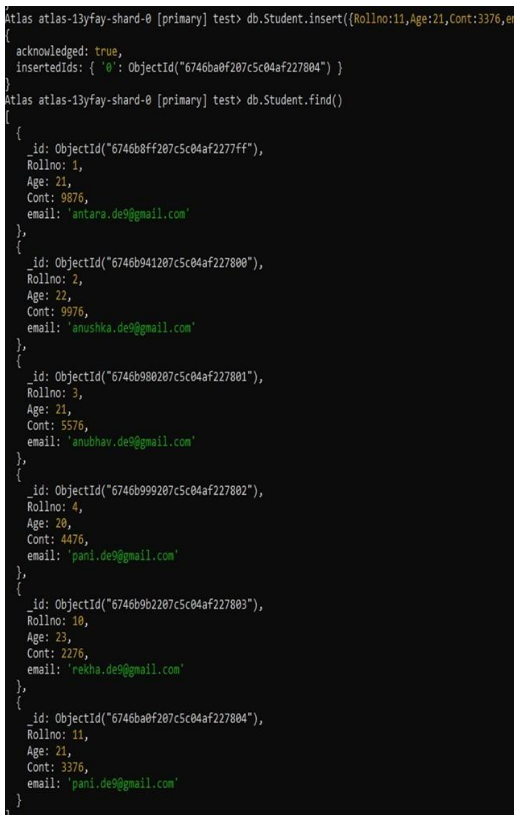
db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com
"});

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"});

Displaying tables

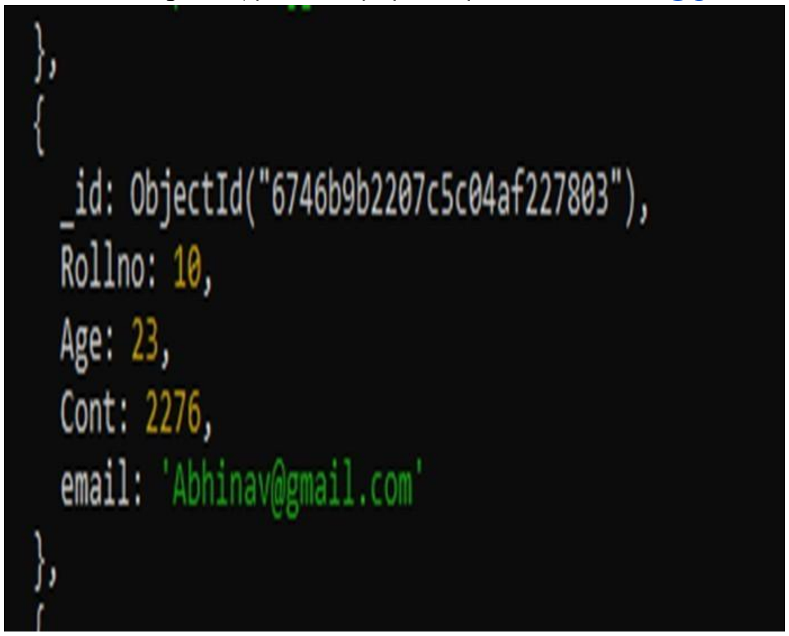


```
Atlas atlas-13yfay-shard-0 [primary] test> db.Student.insert({Rollno:11,Age:21,Cont:3376,e
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746ba0f207c5c04af227804") }
}
Atlas atlas-13yfay-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("6746b8ff207c5c04af2277ff"),
    Rollno: 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 5.

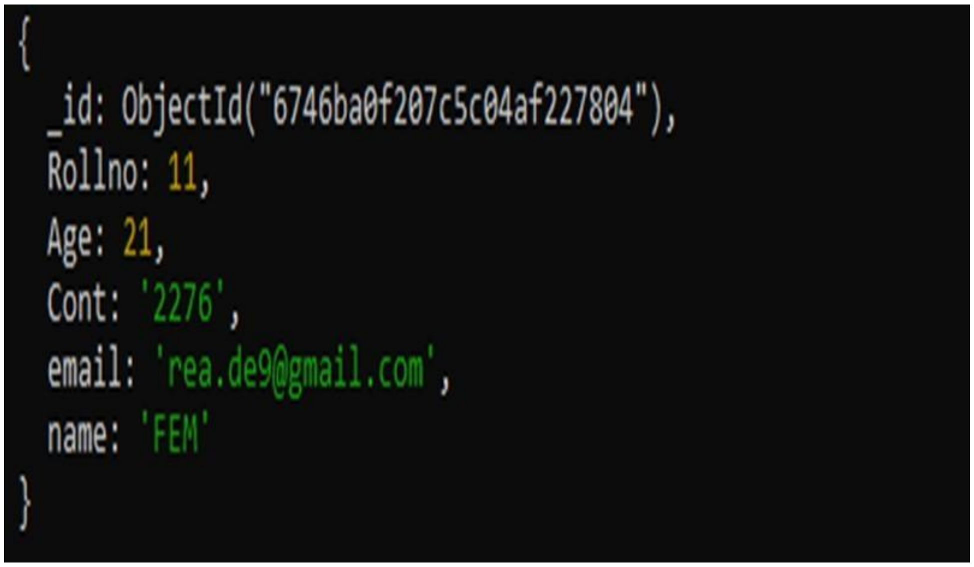
db.Student.update({rollno:5},{$set:{email:"abhinav@gmail.com"}});



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

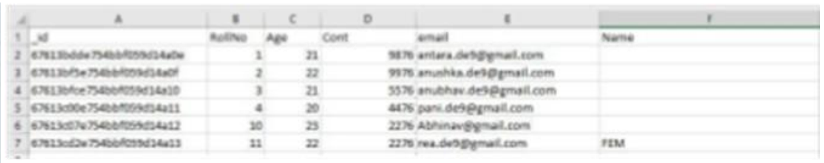
Replace the student name from "ABC" to "FEM" of rollno 11.

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'
}
```

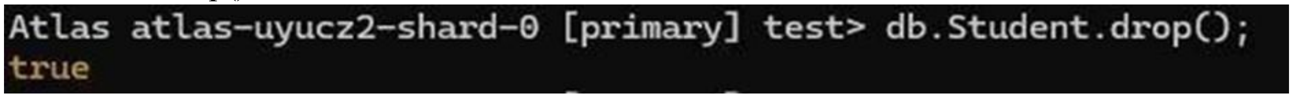
Export the created table into local files



	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	25	2276	Abhinav@gmail.com	
7	67613cd2e754bbf059d14a13	11	22	2276	rea.de9@gmail.com	FEM

Drop table

db.Student.drop()

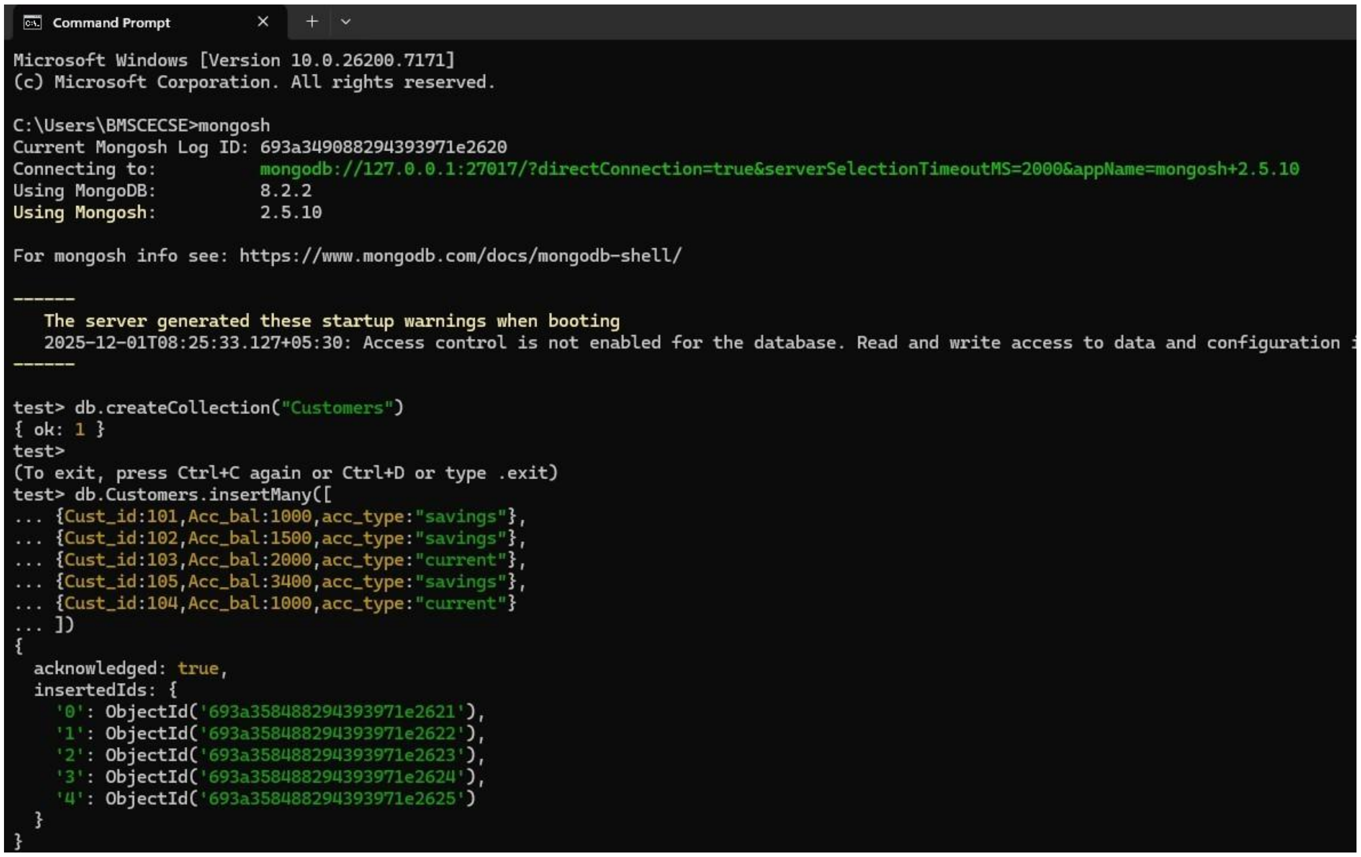


```
Atlas atlas-uyucz2-shard-0 [primary] test> db.Student.drop();
true
```

Experiment 10: NOSQL- Customer Database

Create database

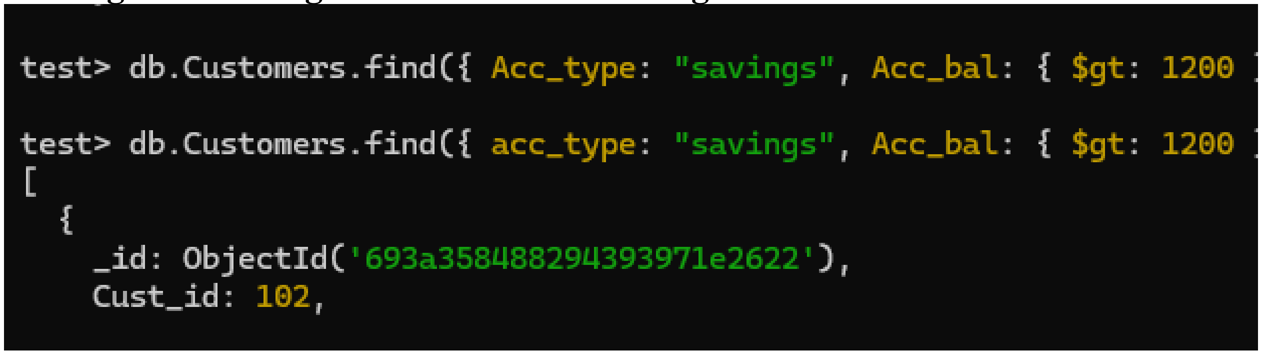
Inserting Values:



```
Microsoft Windows [Version 10.0.26200.7171]
(c) Microsoft Corporation. All rights reserved.

C:\Users\BMSCECSE>mongosh
Current Mongosh Log ID: 693a349088294393971e2620
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.10
Using MongoDB:          8.2.2
Using Mongosh:          2.5.10

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-12-01T08:25:33.127+05:30: Access control is not enabled for the database. Read and write access to data and configuration i
------

test> db.createCollection("Customers")
{ ok: 1 }
test>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
test> db.Customers.insertMany([
... {Cust_id:101,Acc_bal:1000,acc_type:"savings"},
... {Cust_id:102,Acc_bal:1500,acc_type:"savings"},
... {Cust_id:103,Acc_bal:2000,acc_type:"current"},
... {Cust_id:105,Acc_bal:3400,acc_type:"savings"},
... {Cust_id:104,Acc_bal:1000,acc_type:"current"}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693a358488294393971e2621'),
    '1': ObjectId('693a358488294393971e2622'),
    '2': ObjectId('693a358488294393971e2623'),
    '3': ObjectId('693a358488294393971e2624'),
    '4': ObjectId('693a358488294393971e2625')
  }
}
```

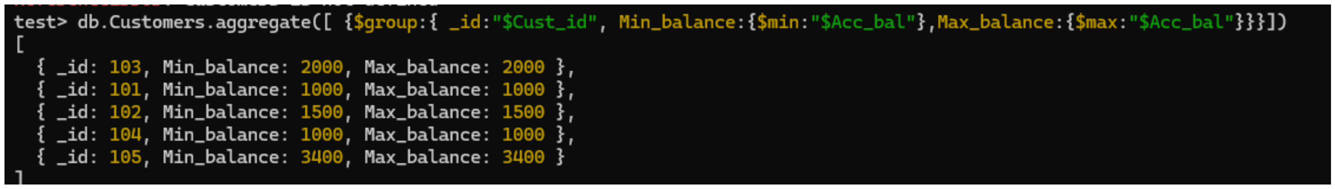
QUERIES:

Finding all checking accounts with balance greater than 12000



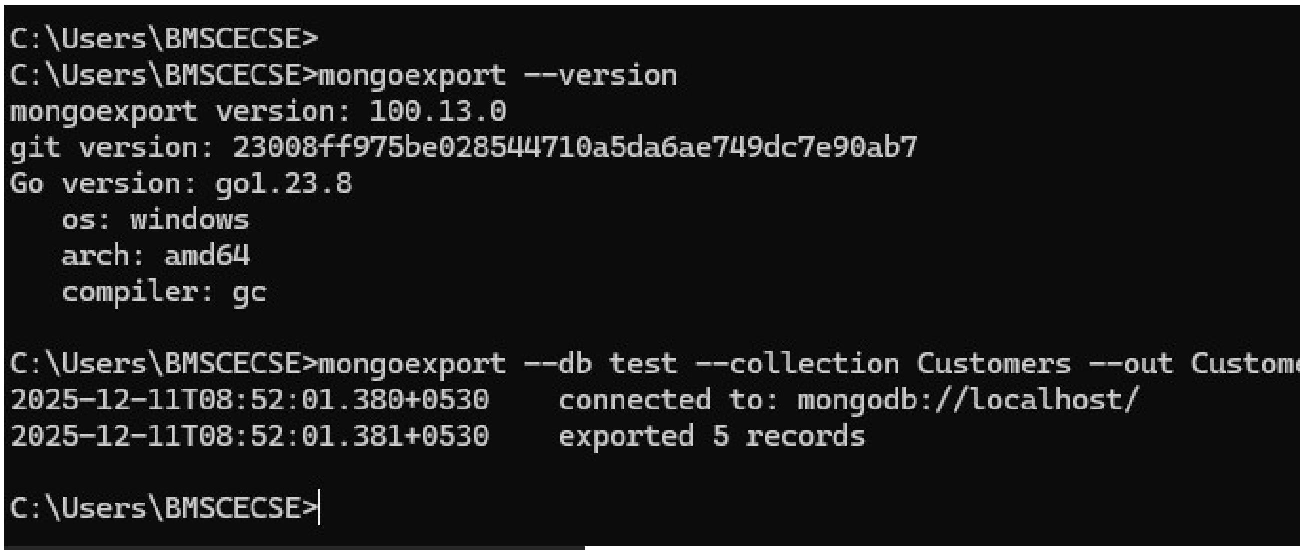
```
test> db.Customers.find({ Acc_type: "savings", Acc_bal: { $gt: 1200

test> db.Customers.find({ acc_type: "savings", Acc_bal: { $gt: 1200
[
  {
    _id: ObjectId('693a358488294393971e2622'),
    Cust_id: 102,
```

Finding the maximum and minimum balance of each customer

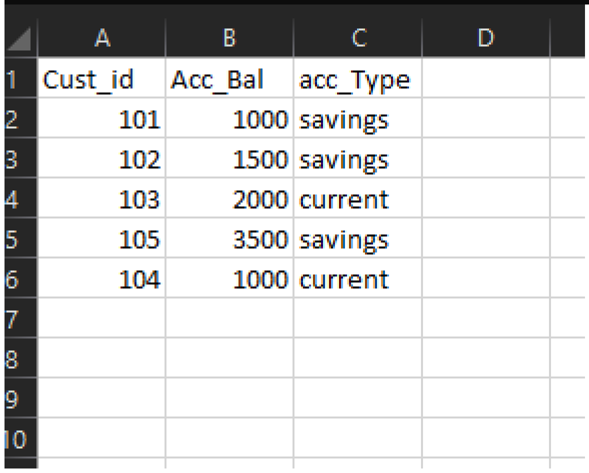


```
test> db.Customers.aggregate([ {$group:{ _id:"$Cust_id", Min_balance:{$min:"$Acc_bal"},Max_balance:{$max:"$Acc_bal"}}}])
[
  { _id: 103, Min_balance: 2000, Max_balance: 2000 },
  { _id: 101, Min_balance: 1000, Max_balance: 1000 },
  { _id: 102, Min_balance: 1500, Max_balance: 1500 },
  { _id: 104, Min_balance: 1000, Max_balance: 1000 },
  { _id: 105, Min_balance: 3400, Max_balance: 3400 }
]
```

Exporting the collection to a json file

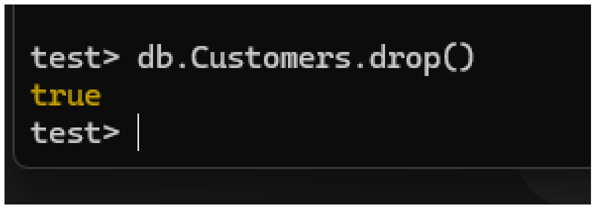


```
C:\Users\BMSCECSE>
C:\Users\BMSCECSE>mongoexport --version
mongoexport version: 100.13.0
git version: 23008ff975be028544710a5da6ae749dc7e90ab7
Go version: go1.23.8
   os: windows
   arch: amd64
   compiler: gc

C:\Users\BMSCECSE>mongoexport --db test --collection Customers --out Custome
2025-12-11T08:52:01.380+0530    connected to: mongodb://localhost/
2025-12-11T08:52:01.381+0530    exported 5 records

C:\Users\BMSCECSE>
```



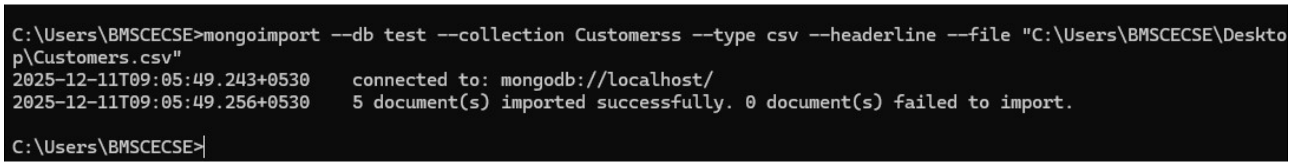
	A	B	C	D
1	Cust_id	Acc_Bal	acc_Type	
2	101	1000	savings	
3	102	1500	savings	
4	103	2000	current	
5	105	3500	savings	
6	104	1000	current	
7				
8				
9				
10				

Dropping collection "Customer"



```
test> db.Customers.drop()
true
test>
```

Exporting from a json file to the collection



```
C:\Users\BMSCECSE>mongoimport --db test --collection Customerss --type csv --headerline --file "C:\Users\BMSCECSE\Deskto
p\Customers.csv"
2025-12-11T09:05:49.243+0530    connected to: mongodb://localhost/
2025-12-11T09:05:49.256+0530    5 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\BMSCECSE>
```

Experiment 11: NOSQL- Restaurant Database

Creating database

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

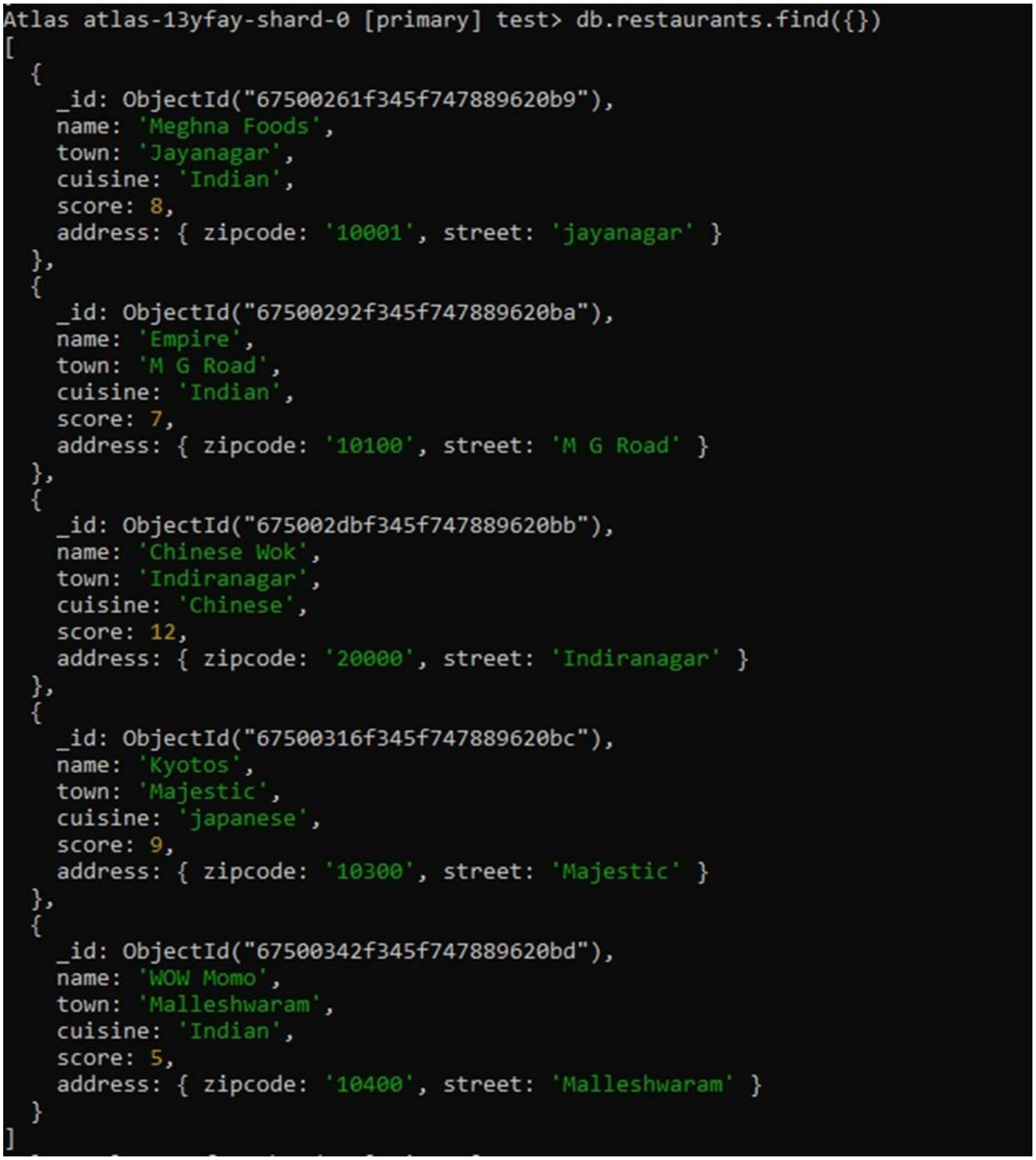
Inserting Values:

```
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" }} ])
```

QUERIES

Write a Mo.ngoDB query to display all the documents in the collection restaurants.

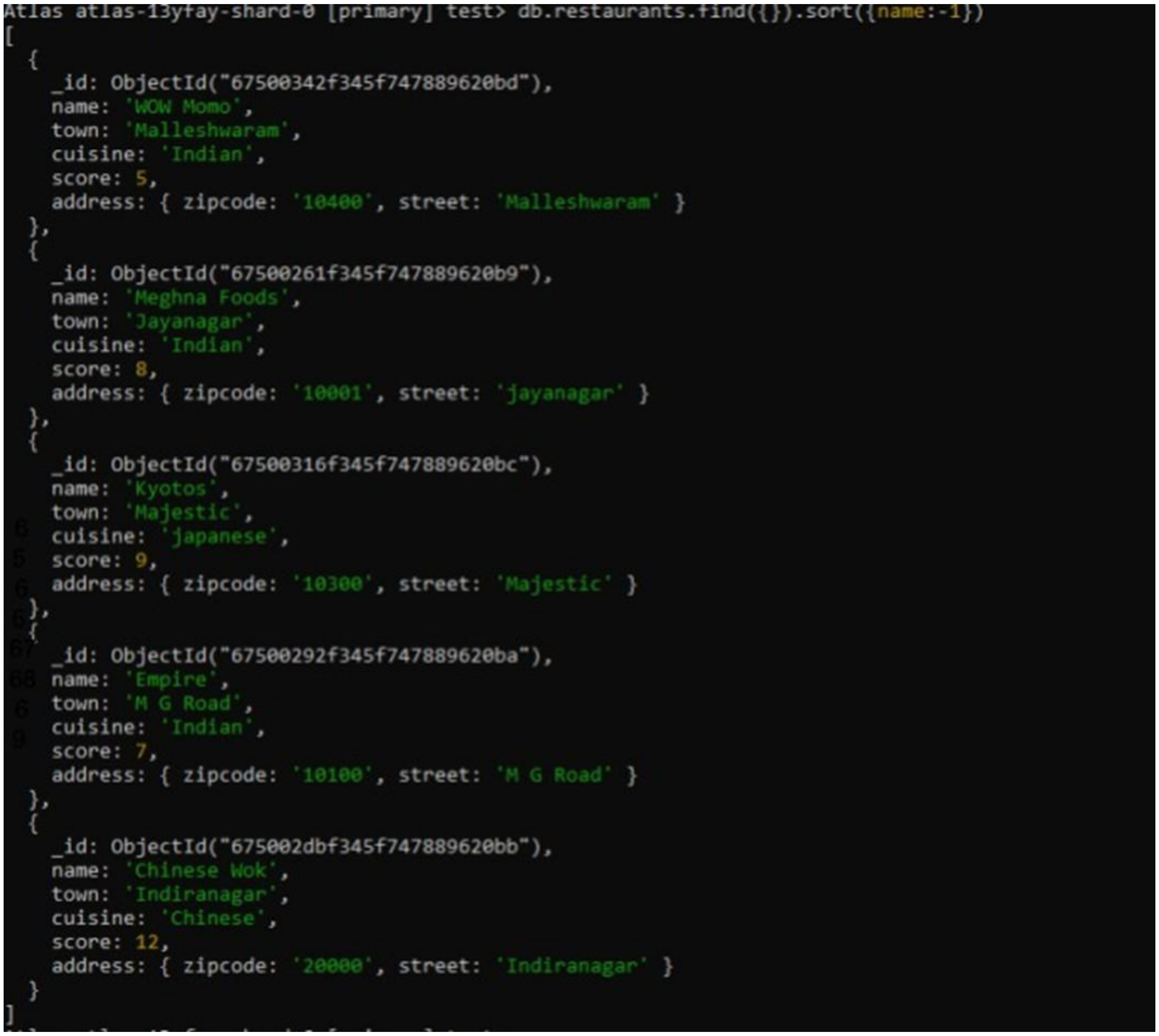
```
db.Restraunt.find()
```



```
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' }
  }
]
```

Query to arrange the name of the restaurants in descending along with all the columns.

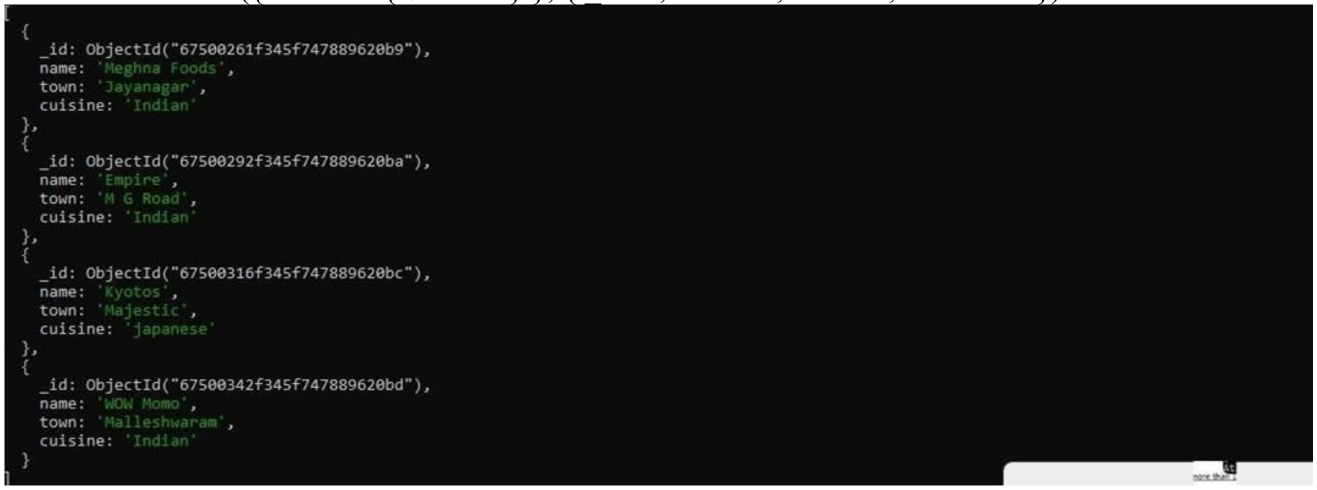
db.restaurants.find({}).sort({ name: -1 })



```
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' }
  }
]
```

Query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10

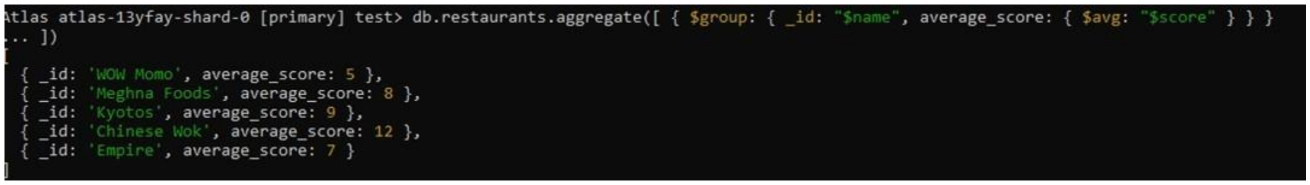
db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })



```
[
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese'
  },
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
]
```

Query to find the average score for each restaurant

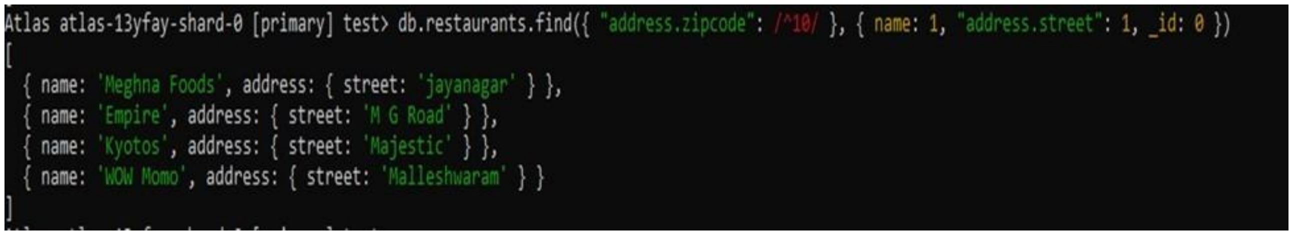
db.restaurants.aggregate([{ $group: { _id: "$name", average_score: { $avg: "$score" } } }])



```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }
... ])
[
  { _id: 'WOW Momo', average_score: 5 },
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Chinese Wok', average_score: 12 },
  { _id: 'Empire', average_score: 7 }
]
```

Query to find the 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 })



```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
[
  { name: 'Meghna Foods', address: { street: 'jayanagar' } },
  { name: 'Empire', address: { street: 'M G Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momo', address: { street: 'Malleshwaram' } }
]
```