VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

PRAGATHI.M (1BM24CS208)

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-2024 to Jan-2025

B.M.S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Database Management Systems (23CS3PCDBM)" carried out by **Pragathi.M (1BM24CS208),** 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.

Surabhi S Assistant Professor Department of CSE, BMSCE	Dr. Joythi S Nayak Professor & HOD Department of CSE, BMSCE

Index

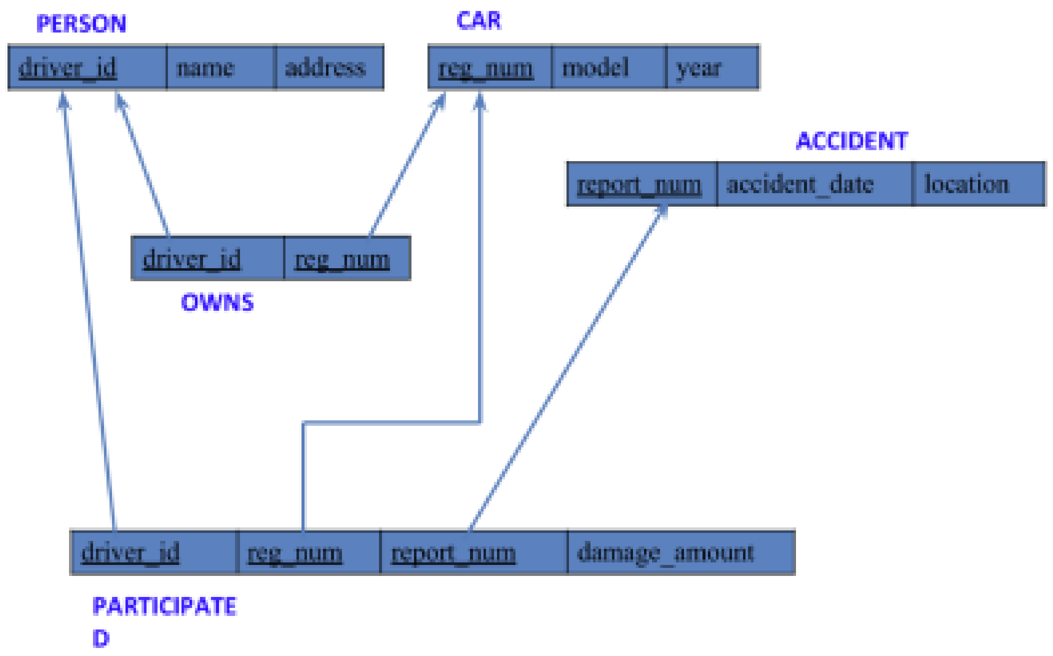
Insurance Database

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),

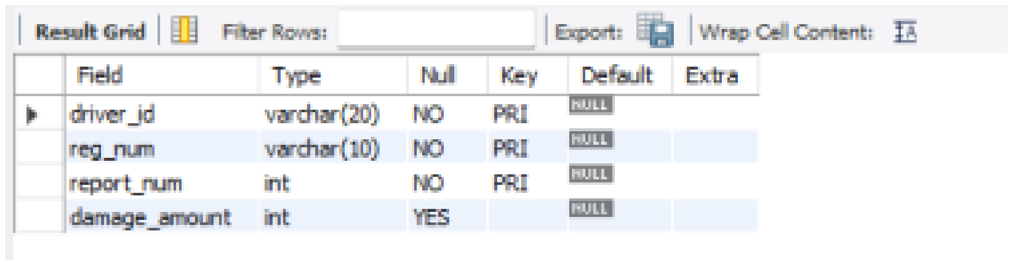
model varchar(10),
year int,

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

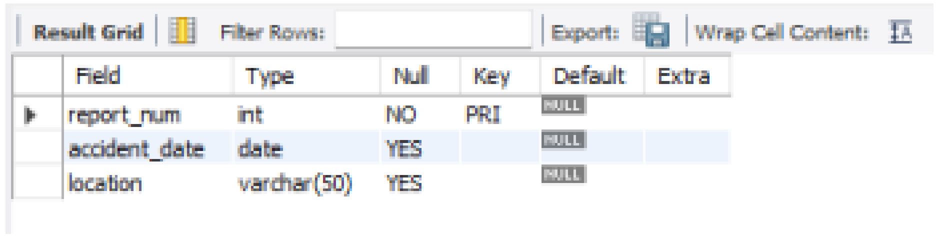
desc person;



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	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

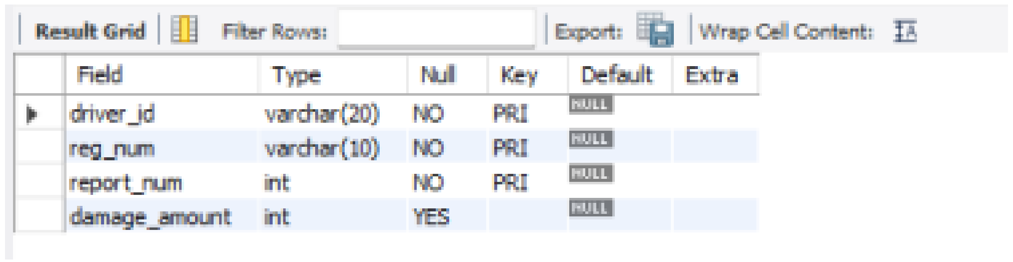
desc accident;



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(50)	YES		NULL	

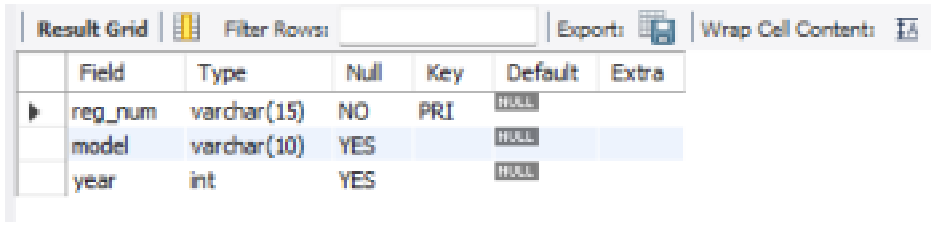
desc participated;



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	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

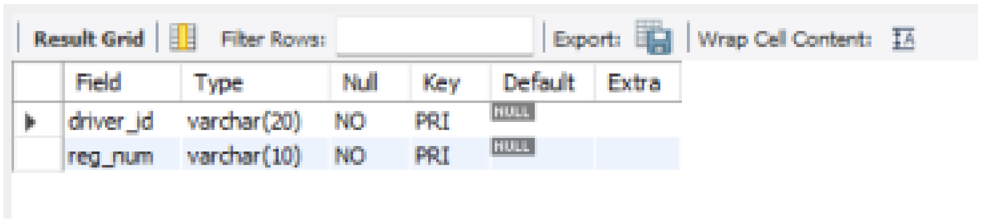
desc car;



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

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

desc owns;



	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");

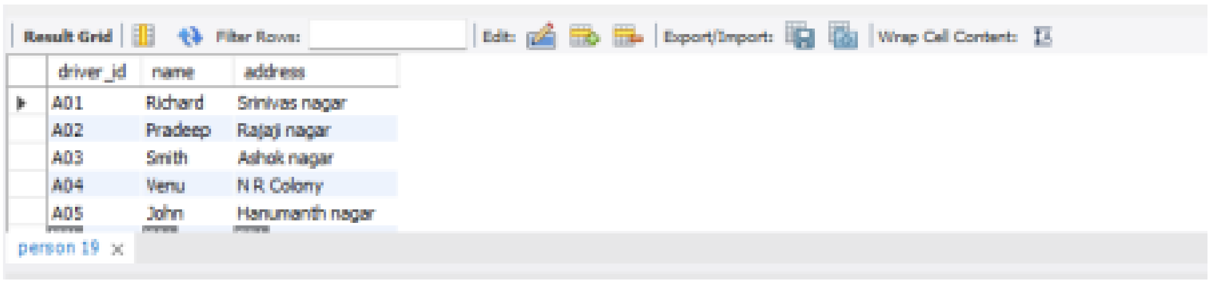
insert into person values("A02","Pradeep", "Rajaji nagar");

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;



	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

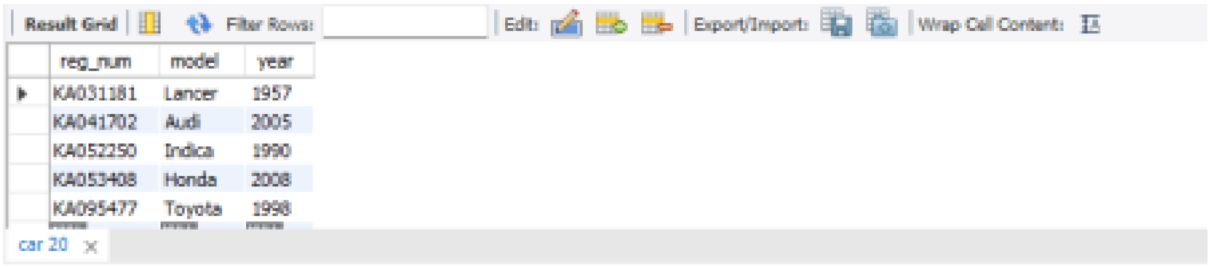
insert into car values("KA052250","Indica", "1990");
insert into car values("KA031181","Lancer", "1957");

insert into car values("KA095477","Toyota", "1998");

insert into car values("KA053408","Honda", "2008");

insert into car values("KA041702","Audi", "2005");

select * from car;



	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");

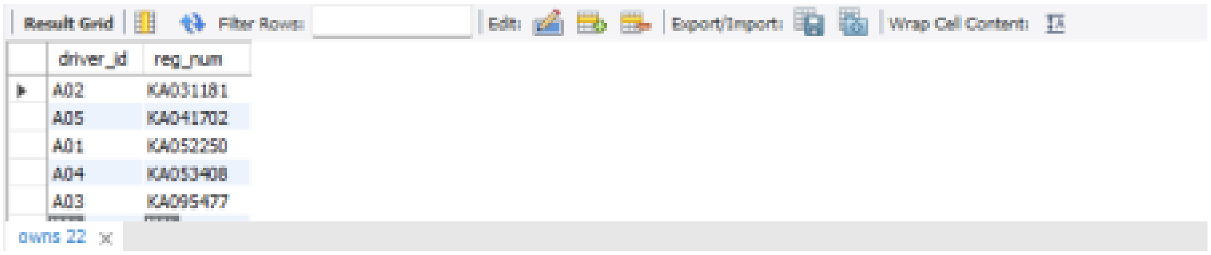
insert into owns values("A02","KA031181");

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

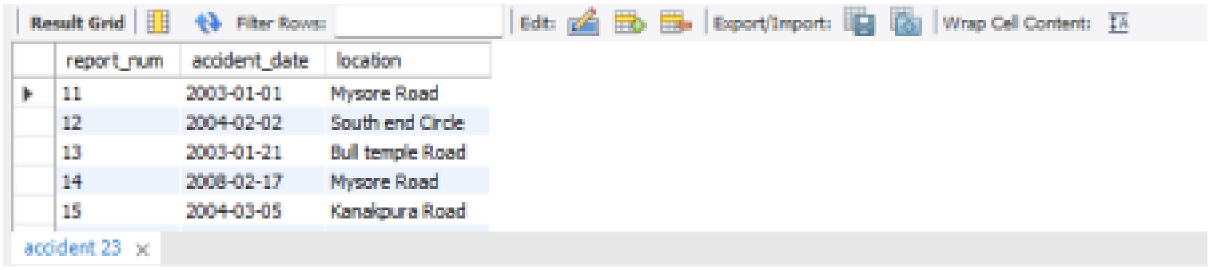
insert into accident values(11,'2003-01-01',"Mysore Road");
insert into accident values(12,'2004-02-02',"South end Circle");

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

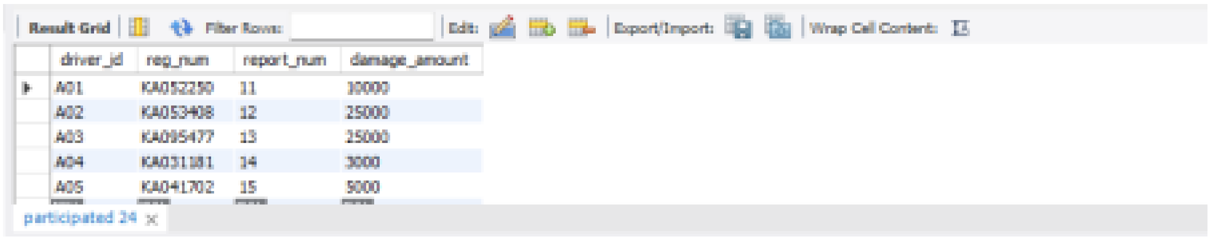
insert into participated values("A01","KA052250",11,10000);
insert into participated values("A02","KA053408",12,50000);

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;



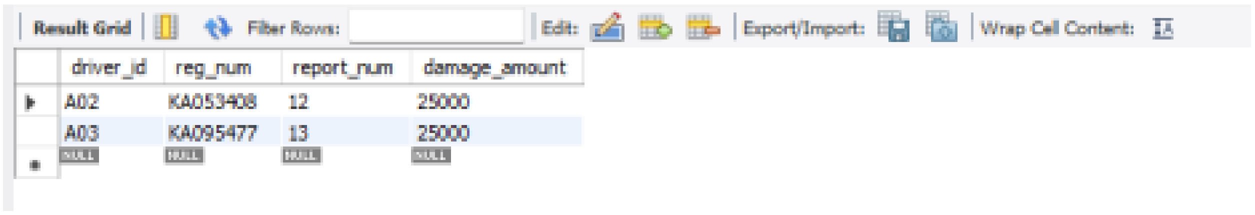
driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A02	KA053408	12	25000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000

participated 24

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.**

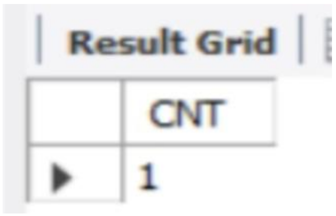
```
update participated

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.**

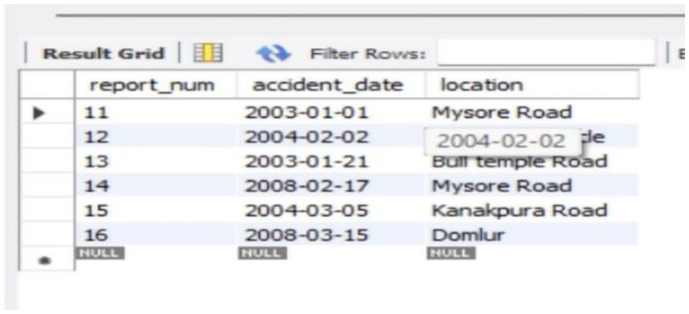
```
select count(distinct driver_id) CNT
from participated a, accident b
where a.report_num=b.report_num and b.accident_date like '2008%';
```



CNT
1

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

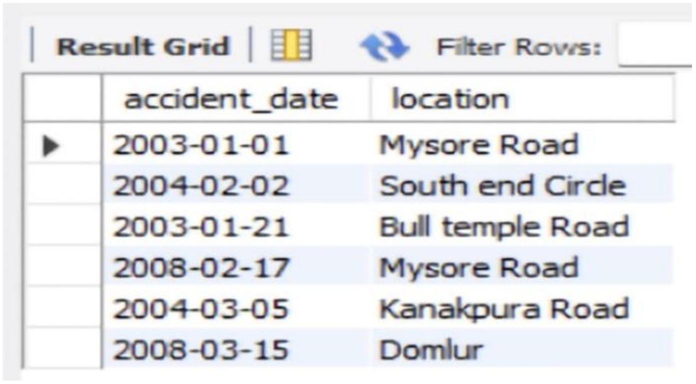
```
insert into accident values(16,'2008-03-08',"Domlur");
select * from accident;
```



	report_num	accident_date	location
▶	11	2003-01-01	Mysore Road
	12	2004-02-02	2004-02-02 cle
	13	2003-01-21	Bull temple Road
	14	2008-02-17	Mysore Road
	15	2004-03-05	Kanakpura Road
	16	2008-03-15	Domlur
●	NULL	NULL	NULL

- **Display Accident date and location**

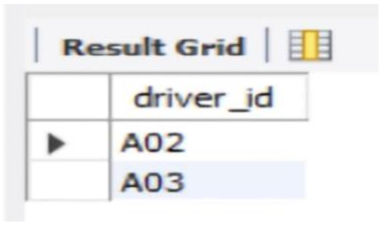
```
select accident_date, location from accident;
```



	accident_date	location
▶	2003-01-01	Mysore Road
	2004-02-02	South end Circle
	2003-01-21	Bull temple Road
	2008-02-17	Mysore Road
	2004-03-05	Kanakpura Road
	2008-03-15	Domlur

- **Display driver id who did accident with damage amount greater than or equal to Rs.25000**

```
select driver_id from PARTICIPATED where damage_amount >=25000;
```



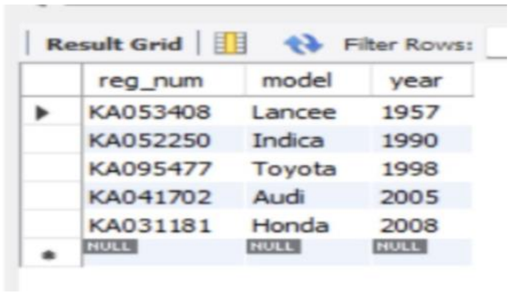
	driver_id
▶	A02
	A03

More Queries on Insurance Database

(Week 2)

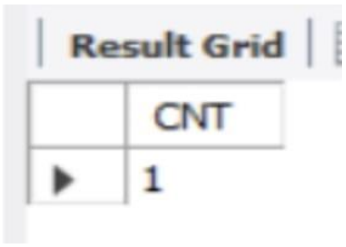
- Display the entire CAR relation in the ascending order of manufacturing year.
- Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
- Find the total number of people who owned cars that involved in accidents in 2008.
- List the entire participated relation in the descending order of damage amount.
- Find the average damage amount.
- Delete the tuple whose damage amount is below the average damage amount.
- List the name of drivers whose damage is greater than the average damage amount.
- Find maximum damage amount.

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



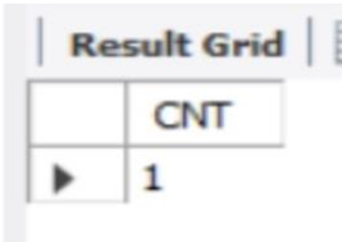
reg_num	model	year
KA053408	Lancee	1957
KA052250	Indica	1990
KA095477	Toyota	1998
KA041702	Audi	2005
KA031181	Honda	2008
NULL	NULL	NULL

- **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';



CNT
1

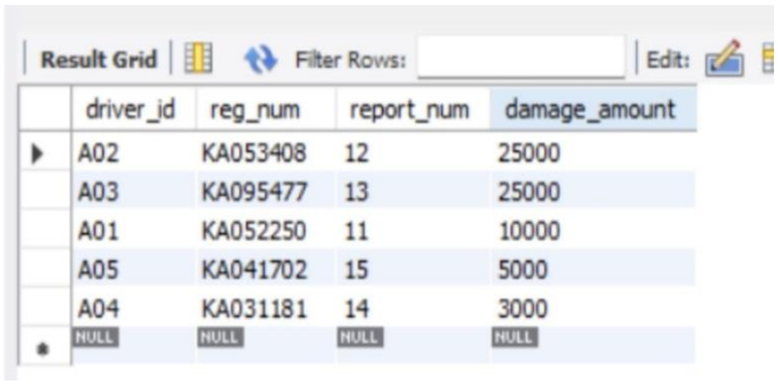
- **Find the total number of people who owned cars that were involved in accidents in 2008.**
  select count(distinct driver_id) CNT
  from participated a, accident b
  where a.report_num=b.report_num and b.accident_date like '_ _08%';



CNT
1

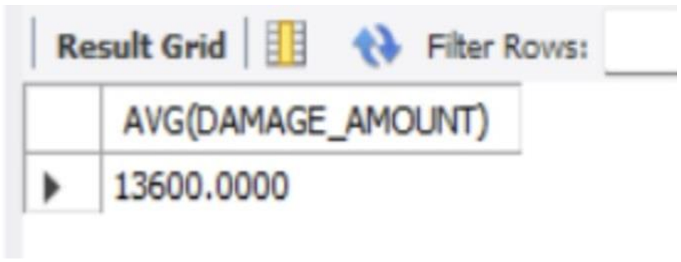
- **List the entire participated relation in the descending order of damage amount.**

  select * from participated order by damage_amount desc;



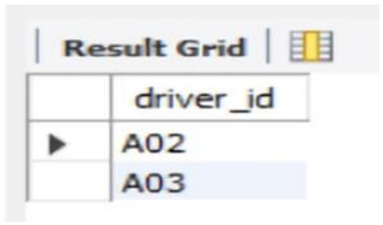
	driver_id	reg_num	report_num	damage_amount
▶	A02	KA053408	12	25000
	A03	KA095477	13	25000
	A01	KA052250	11	10000
	A05	KA041702	15	5000
	A04	KA031181	14	3000
*	NULL	NULL	NULL	NULL

- **Find the average damage amount.**
  select avg(damage_amount) from participated;



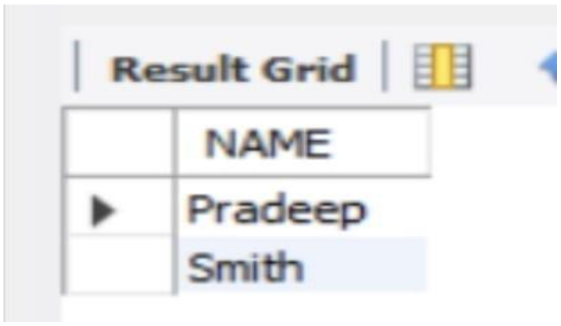
	AVG(DAMAGE_AMOUNT)
▶	13600.0000

- **Delete the tuple whose damage amount is below the average damage amount.**
  delete from participated where damage_amount<(select avg (damage_amount) from participated);
  select * from participated;



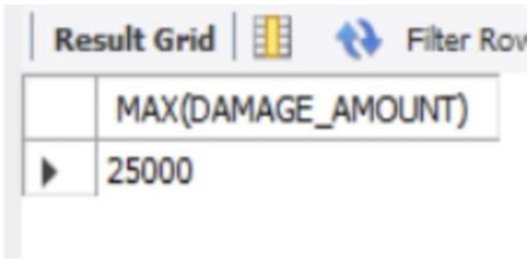
	driver_id
▶	A02
	A03

- **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
Pradeep
Smith
```

- **Find maximum damage amount.**
  select max(damage_amount) from participated;



```
MAX(DAMAGE_AMOUNT)
25000
```

Bank Database

(Week 3)

Branch (branch-name: String, branch-city: String, assets: real)

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

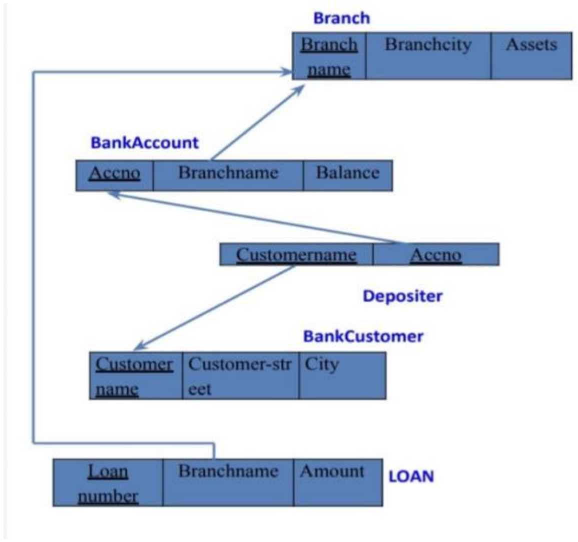
BankCustomer (customer-name: String, customer-street: String,customer-city: String)

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

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

1. Create the above tables by properly specifying the primary keys and the foreign keys.
2. Enter at least five tuples for each relation.
3. Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
4. Find all the customers who have at least two accounts at the same branch (ex.SBI_ResidencyRoad).
5. 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 dhiksha_bank;
use dhiksha_bank;
```

Create Table

```
create table dhiksha_bank.branch(
Branch_name varchar(30),
Branch_city varchar(25),
assets int,
PRIMARY KEY (Branch_name)
);
create table dhiksha_bank.BankAccount(
Accno int,
Branch_name varchar(30),
Balance int,
PRIMARY KEY(Accno),
foreign key (Branch_name) references branch(Branch_name)
);
create table dhiksha_bank.BankCustomer(
Customername varchar(20),
Customer_street varchar(30),
CustomerCity varchar (35),
PRIMARY KEY(Customername));
```

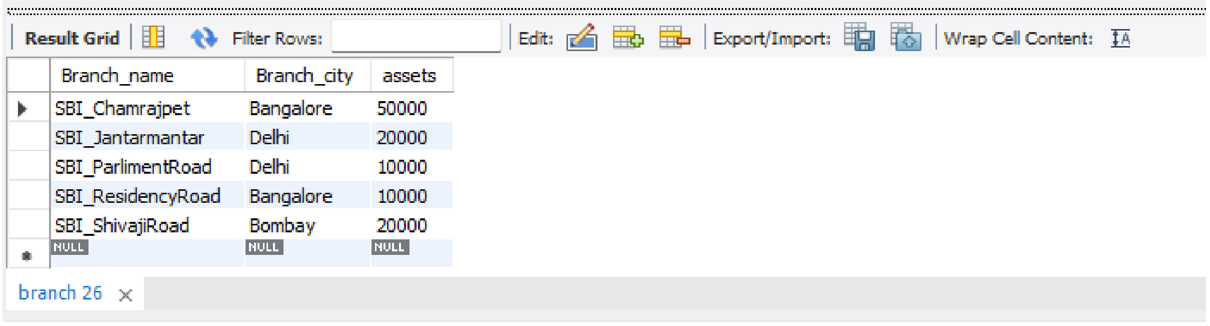
```
create table dhiksha_bank.Depositer(
Customername varchar(20),
Accno int,
PRIMARY KEY(Customername,Accno),
foreign key (Accno) references BankAccount(Accno),
foreign key (Customername) references BankCustomer(Customername)
);
create table dhiksha_bank.Loan(
Loan_number int,
Branch_name varchar(30),
Amount int,
PRIMARY KEY(Loan_number),
foreign key (Branch_name) references branch(Branch_name)
);
```

Enter at least five tuples for each relation.

```
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);

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);

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");
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);

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);
```

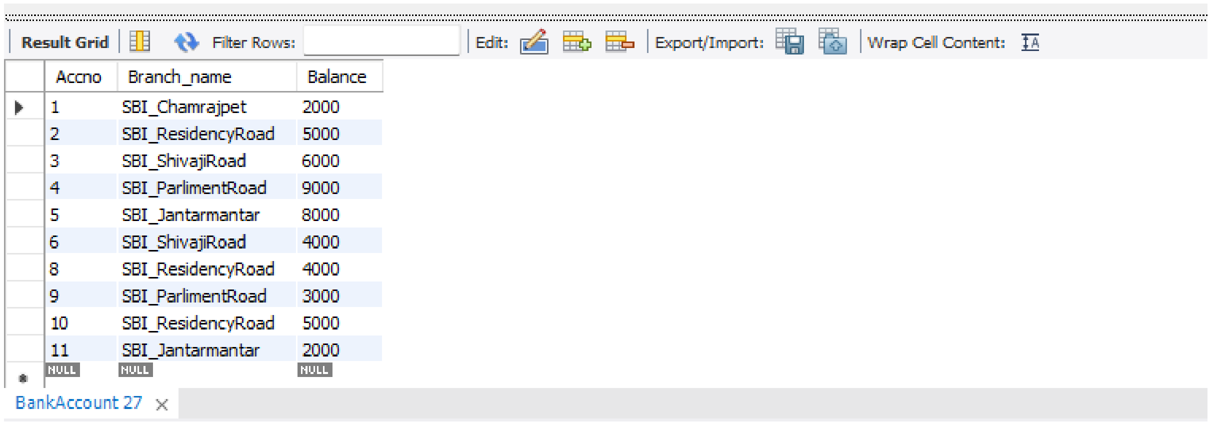
Select from table (SELECTION)

```
select * from branch;
```



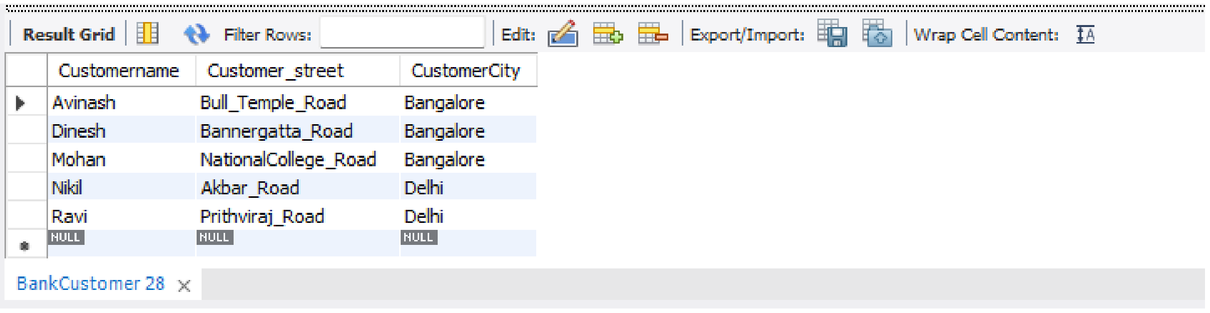
Branch_name	Branch_city	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

select * from BankAccount;



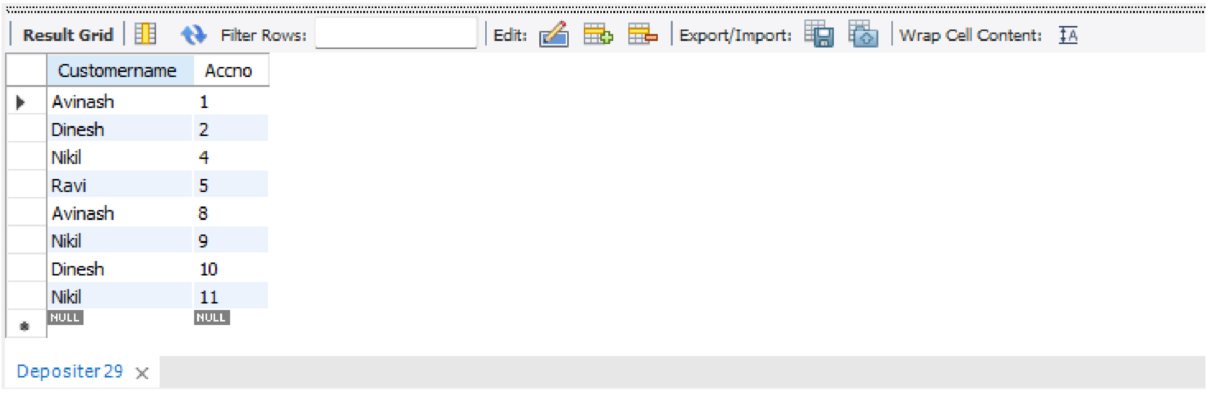
Accno	Branch_name	Balance
1	SBI_Chamrajpet	2000
2	SBI_ResidencyRoad	5000
3	SBI_ShivajiRoad	6000
4	SBI_ParlimentRoad	9000
5	SBI_Jantarmantar	8000
6	SBI_ShivajiRoad	4000
8	SBI_ResidencyRoad	4000
9	SBI_ParlimentRoad	3000
10	SBI_ResidencyRoad	5000
11	SBI_Jantarmantar	2000
NULL	NULL	NULL

select * from BankCustomer;



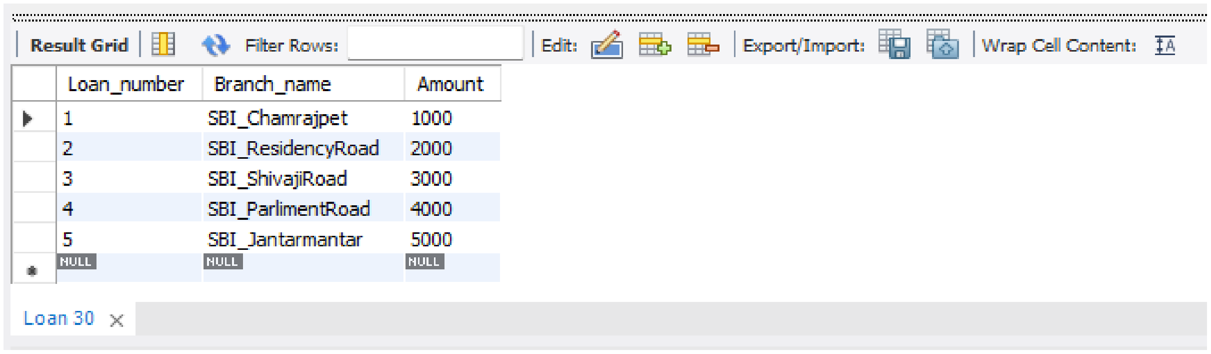
Customername	Customer_street	CustomerCity
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

select * from Depositer;



Customername	Accno
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11
NULL	NULL

select * from Loan;


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

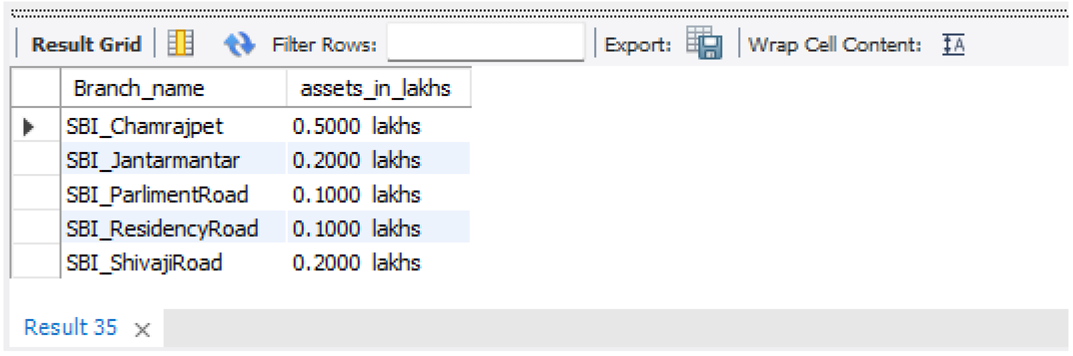
Loan_number	Branch_name	Amount
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_ParlimentRoad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

Loan 30

Queries

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

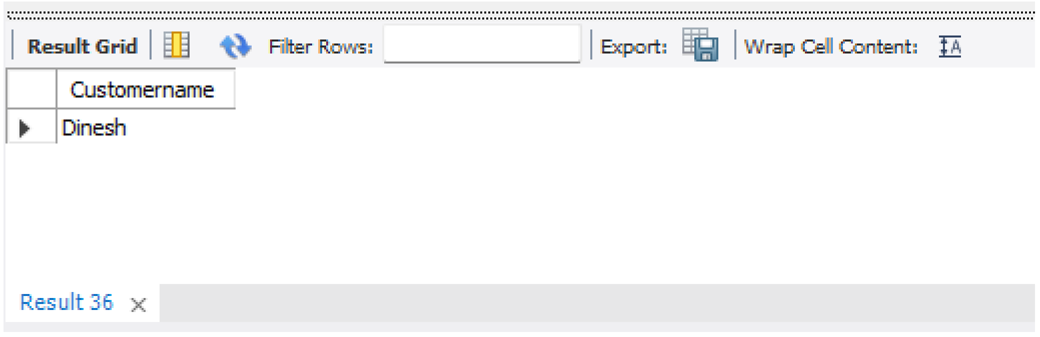
select Branch_name, CONCAT(assets/100000,' lakhs')assets_in_lakhs from branch;



Branch_name	assets_in_lakhs
SBI_Chamrajpet	0.5000 lakhs
SBI_Jantarmantar	0.2000 lakhs
SBI_ParlimentRoad	0.1000 lakhs
SBI_ResidencyRoad	0.1000 lakhs
SBI_ShivajiRoad	0.2000 lakhs

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

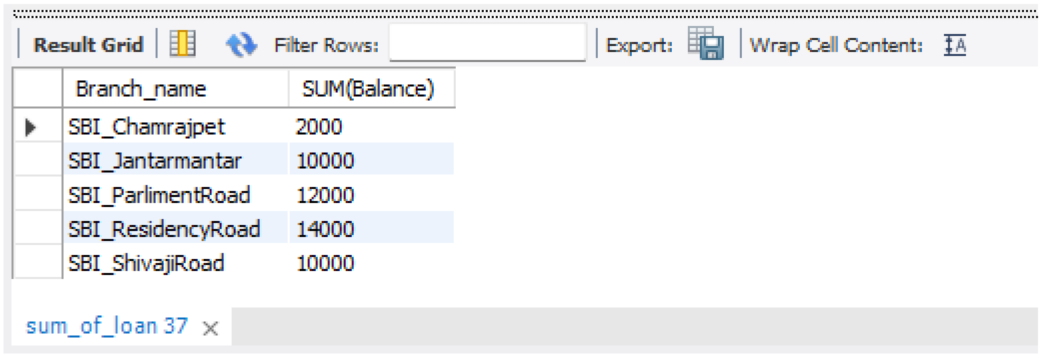
select d.Customername from Depositer d, BankAccount b
where b.Branch_name='SBI_ResidencyRoad' and d.Accno=b.Accno group by d.Customername
having count(d.Accno)>=2;



Customername
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;


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

sum_of_loan 37

More Queries on Bank Database

(Week 4)

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

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

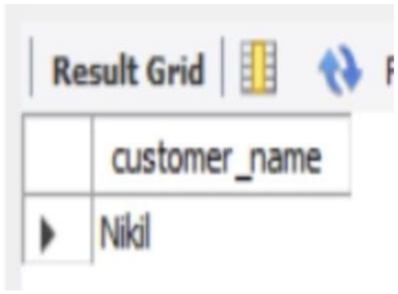
3. Find all customers who have both an account and a loan at the Bangalore branch

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

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

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

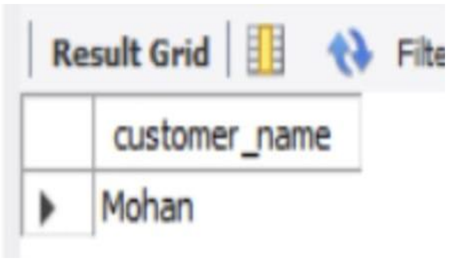
select d.customer_name from bankaccount a,branch b, depositer d where b.branch_name=a.branch_name and a.accno=d.accno and b.branch_city='delhi' group by d.customer_name having count(distinct b.branch_name)=(select count(branch_name)from branch where branch_city='delhi');



customer_name
Nikil

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

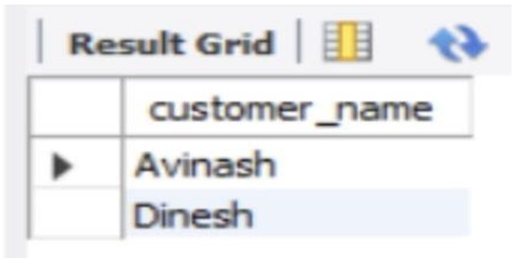
select distinct customer_name from borrower where customer_name not in
(select customer_name from depositor);



customer_name
Mohan

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

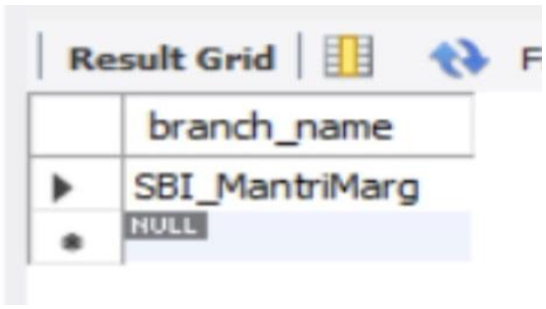
Select customer_name From Borrower ,loan Where borrower.loan_number=loan.loan_number and loan.branch_name in (select branch_name from depositer, bankaccount where depositer.accno = bankaccount.accno And bankaccount.branch_name in(Select branch_name from branch WHERE branch.branch_city='Bangalore'));



customer_name
Avinash
Dinesh

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

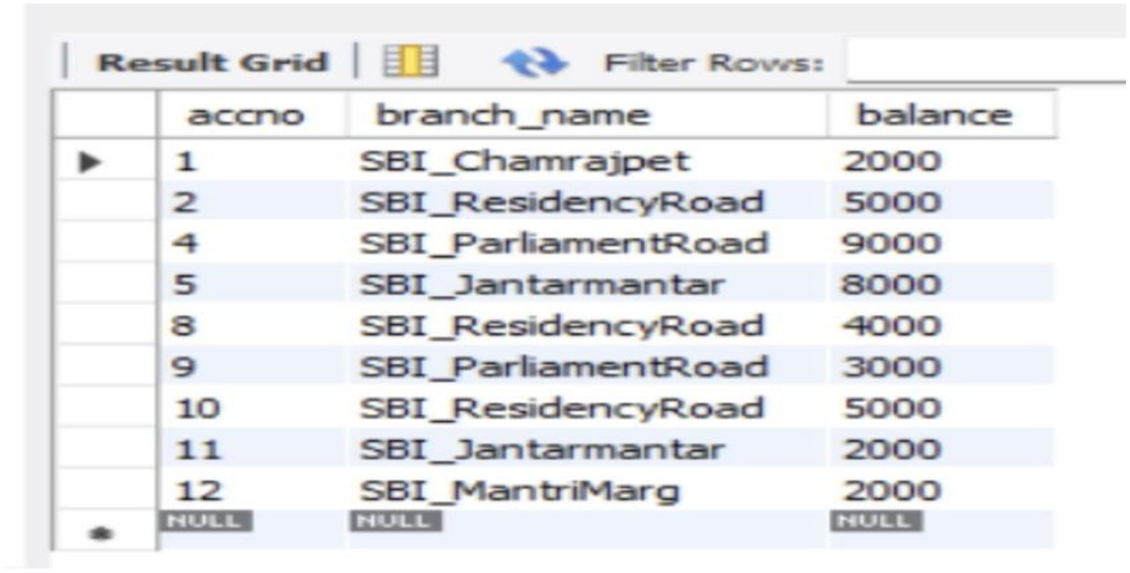
Select branch_name From Branch Where assets>(Select Sum(assets) from branch Where branch_city='Bangalore');



	branch_name
▶	SBI_MantriMarg
*	NULL

- **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);
select * from bankaccount;



	accno	branch_name	balance
▶	1	SBI_Chamrajpet	2000
	2	SBI_ResidencyRoad	5000
	4	SBI_ParliamentRoad	9000
	5	SBI_Jantarmantar	8000
	8	SBI_ResidencyRoad	4000
	9	SBI_ParliamentRoad	3000
	10	SBI_ResidencyRoad	5000
	11	SBI_Jantarmantar	2000
	12	SBI_MantriMarg	2000
*	NULL	NULL	NULL

Employee Database

(Week 5)

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

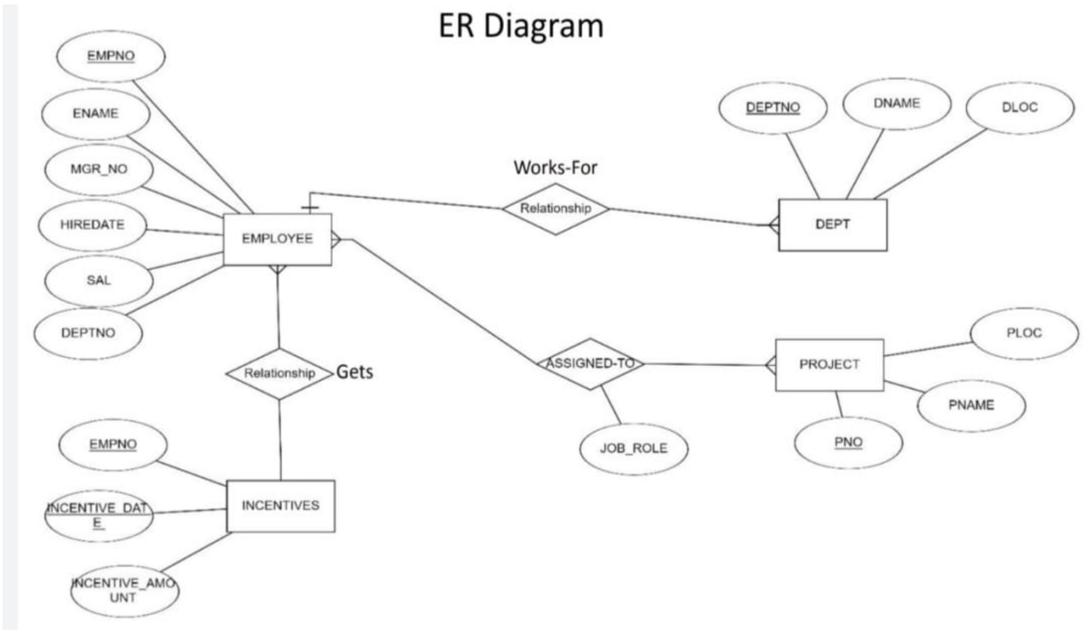
2. Enter greater than five tuples for each table.

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

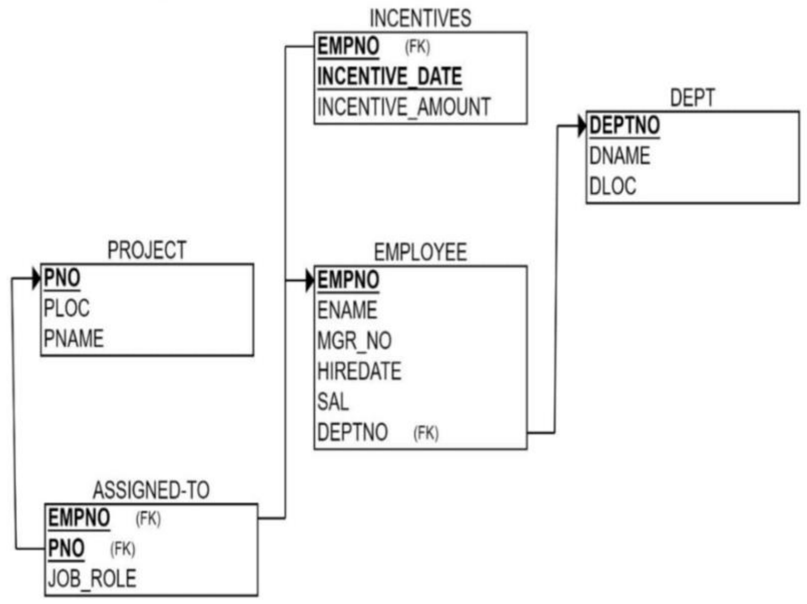
4. Get Employee ID's of those employees who didn't receive incentives

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.

ER Diagram



Schema Diagram



Create Database

create database EmployeeDB;
use EmployeeDB;

Create Table

create table dept (deptno int primary key,dname varchar(30),dloc varchar(30));

create table employee (empno int primary key,ename varchar(30),mgr_no int,hiredate date,sal decimal(10,2),deptno int,foreign key (deptno) references dept(deptno));

create table project (pno int primary key,ploc varchar(30),pname varchar(40));

create table assigned_to (empno int,pno int,job_role varchar(30),primary key (empno, pno),foreign key (empno) references employee(empno),foreign key (pno)references project(pno));

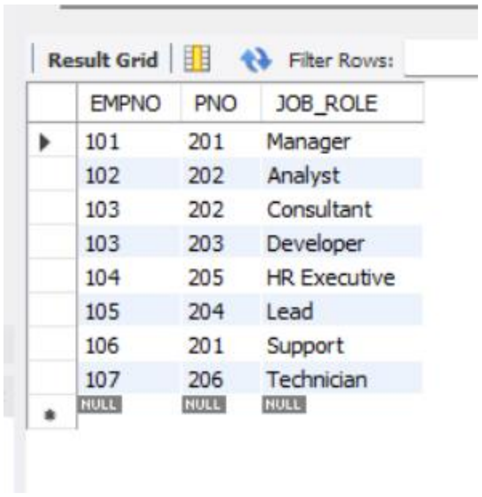
create table incentives (empno int,incentive_date date,incentive_amount decimal(10,2),foreign key (empno) references employee(empno));

- **Enter greater than five tuples for each table.**

```
insert into dept values
(10, 'HR', 'Bengaluru'),
(20, 'Finance', 'Hyderabad'),
(30, 'IT', 'Mysuru'),
(40, 'Admin', 'Chennai'),
(50, 'Marketing', 'Delhi'),
(60, 'Support', 'Pune');


insert into employee values
(101, 'Rita', NULL, '2022-06-10', 45000, 10),
(102, 'Kiran', 101, '2021-07-12', 50000, 20),
(103, 'Ravi', 101, '2023-01-20', 48000, 30),
(104, 'Sneha', 102, '2022-03-15', 47000, 40),
(105, 'Anjali', 102, '2021-12-10', 52000, 50),
(106, 'Rohit', 103, '2019-01-22', 46000, 10),
(107, 'Tejas', 104, '2023-04-18', 44000, 60);

insert into project values
(201, 'Bengaluru', 'Payroll System'),
(202, 'Hyderabad', 'ERP Upgrade'),
(203, 'Mysuru', 'AI Tool'),
(204, 'Delhi', 'Marketing Automation'),
(205, 'Chennai', 'Database Migration'),
(206, 'Pune', 'Support Portal');

insert into assigned_to values
(101, 201, 'Manager'),
(102, 202, 'Analyst'),
(103, 203, 'Developer'),
(104, 205, 'HR Executive'),
(105, 204, 'Lead'),
(106, 201, 'Support'),
(107, 206, 'Technician'),
(103, 202, 'Consultant');
```

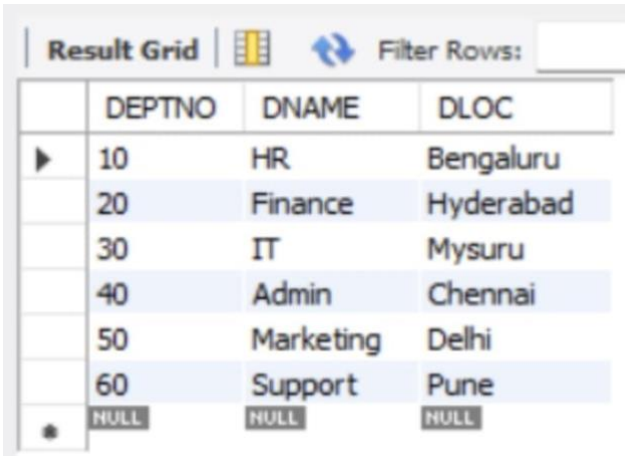
```
insert into incentives values
(101, '2024-12-01', 5000),
(102, '2024-11-15', 3000),
(105, '2024-12-20', 2500),
(107, '2024-09-12', 4000),
(101, '2025-01-10', 2000),
(102, '2025-02-14', 2500);
```

```
select * from employeedb.assigned_to;
```



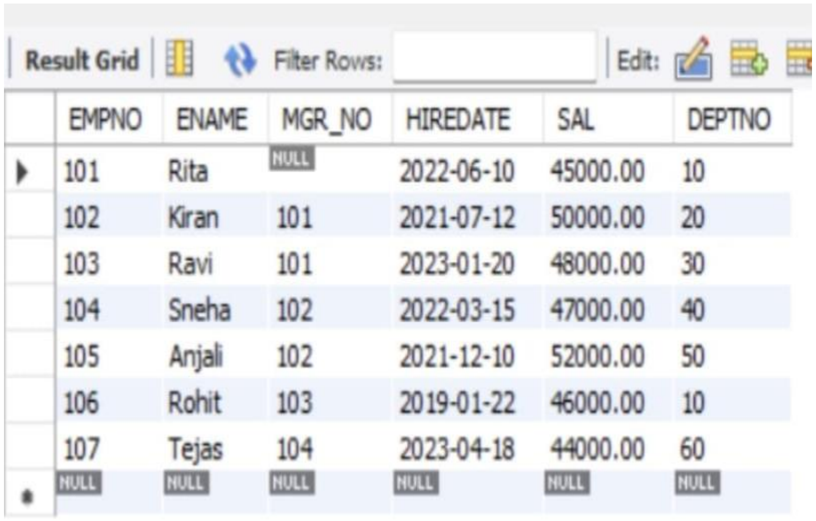
EMPNO	PNO	JOB_ROLE
101	201	Manager
102	202	Analyst
103	202	Consultant
103	203	Developer
104	205	HR Executive
105	204	Lead
106	201	Support
107	206	Technician
NULL	NULL	NULL

```
select * from employeedb.dept;
```



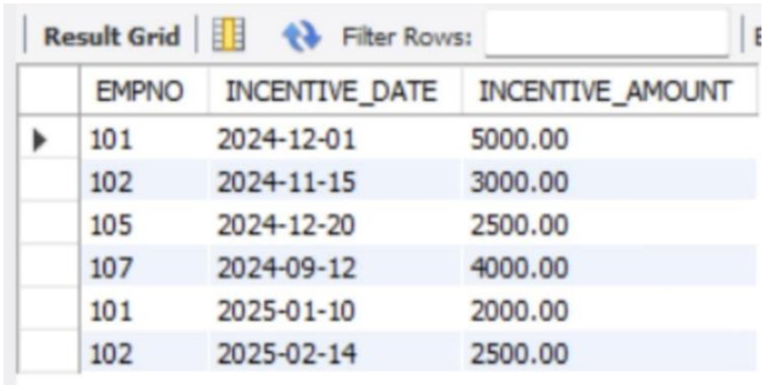
DEPTNO	DNAME	DLOC
10	HR	Bengaluru
20	Finance	Hyderabad
30	IT	Mysuru
40	Admin	Chennai
50	Marketing	Delhi
60	Support	Pune
NULL	NULL	NULL

select * from employeedb.employee;



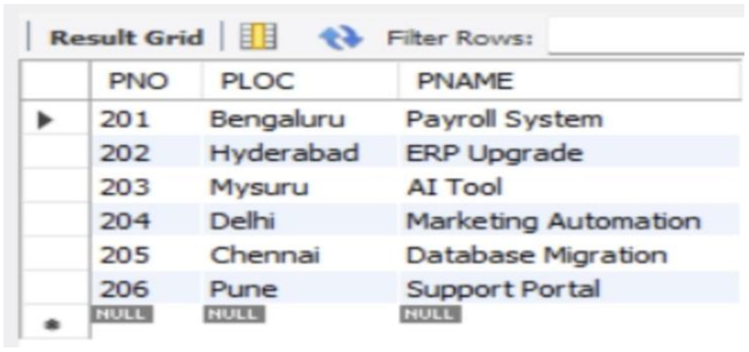
EMPNO	ENAME	MGR_NO	HIREDATE	SAL	DEPTNO
101	Rita	NULL	2022-06-10	45000.00	10
102	Kiran	101	2021-07-12	50000.00	20
103	Ravi	101	2023-01-20	48000.00	30
104	Sneha	102	2022-03-15	47000.00	40
105	Anjali	102	2021-12-10	52000.00	50
106	Rohit	103	2019-01-22	46000.00	10
107	Tejas	104	2023-04-18	44000.00	60
NULL	NULL	NULL	NULL	NULL	NULL

select * from employeedb.incentives;



EMPNO	INCENTIVE_DATE	INCENTIVE_AMOUNT
101	2024-12-01	5000.00
102	2024-11-15	3000.00
105	2024-12-20	2500.00
107	2024-09-12	4000.00
101	2025-01-10	2000.00
102	2025-02-14	2500.00

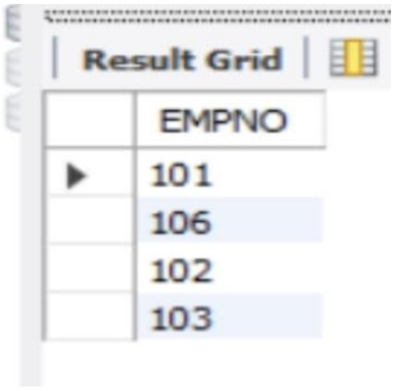
select * from employeedb.project;



PNO	PLOC	PNAME
201	Bengaluru	Payroll System
202	Hyderabad	ERP Upgrade
203	Mysuru	AI Tool
204	Delhi	Marketing Automation
205	Chennai	Database Migration
206	Pune	Support Portal
NULL	NULL	NULL

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

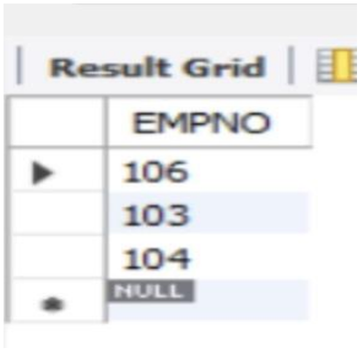
```
select distinct a.empno
from assigned_to a
join project p on a.pno = p.pno
where p.ploc in ('Bengaluru', 'Hyderabad', 'Mysuru');
```



```
Result Grid
EMPNO
101
106
102
103
```

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

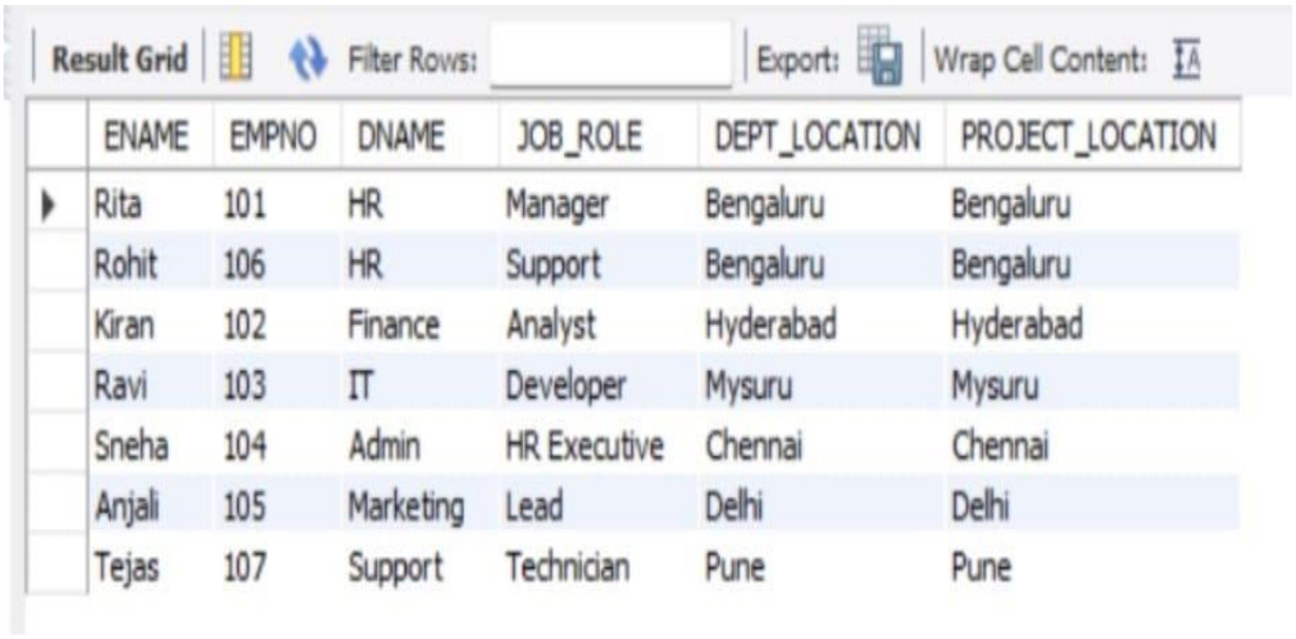
```
select empno
from employee
where empno not in (select empno from incentives);
```



```
Result Grid
EMPNO
106
103
104
NULL
```

- **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, a.job_role, d.dloc as dept_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;
```



ENAME	EMPNO	DNAME	JOB_ROLE	DEPT_LOCATION	PROJECT_LOCATION
Rita	101	HR	Manager	Bengaluru	Bengaluru
Rohit	106	HR	Support	Bengaluru	Bengaluru
Kiran	102	Finance	Analyst	Hyderabad	Hyderabad
Ravi	103	IT	Developer	Mysuru	Mysuru
Sneha	104	Admin	HR Executive	Chennai	Chennai
Anjali	105	Marketing	Lead	Delhi	Delhi
Tejas	107	Support	Technician	Pune	Pune

More Queries on Employee Database

(Week 6)

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

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

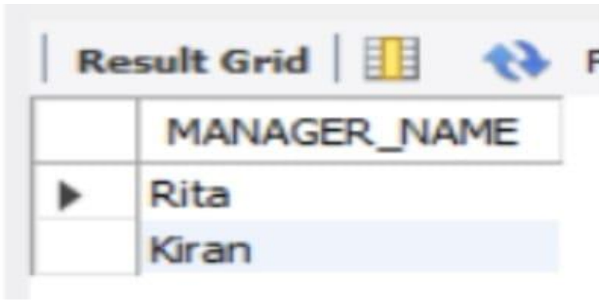
3. Find the name of the second top level managers of each department.

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

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

- **List the name of the managers with the maximum employees.**

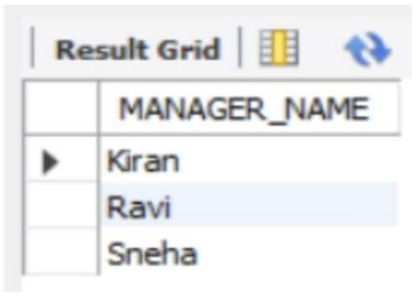
select e.ename as manager_name from employee e where e.empno in (select mgr_no from employee where mgr_no is not null group by mgr_no
 having count(*) >= all (select count(*) from employee where mgr_no is not null group by mgr_no));


Result Grid

```
MANAGER_NAME
Rita
Kiran
```

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

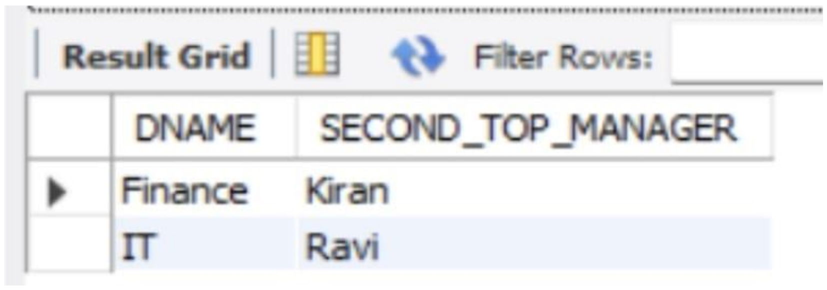
```
select m.ename as manager_name
from employee m
where m.empno in (
   select e.mgr_no
   from employee e
   where e.mgr_no is not null
)
and m.sal > (
   select avg(e2.sal)
   from employee e2
   where e2.mgr_no = m.empno
);
```


Result Grid

```
MANAGER_NAME
Kiran
Ravi
Sneha
```

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

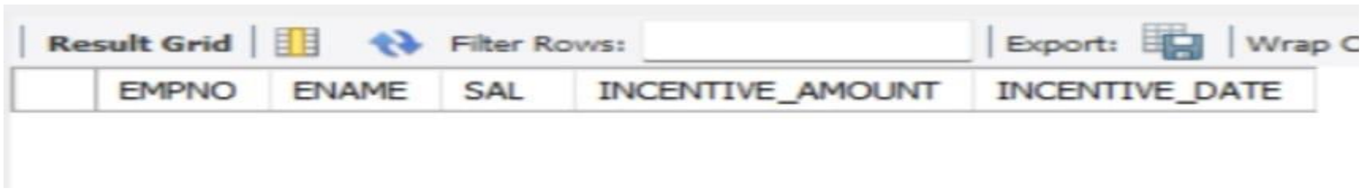
```
select d.dname, e.ename as second_top_manager
from employee e
join dept d on e.deptno = d.deptno
where e.mgr_no is not null
and e.mgr_no in (select empno from employee where mgr_no is null);
```



DNAME	SECOND_TOP_MANAGER
Finance	Kiran
IT	Ravi

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

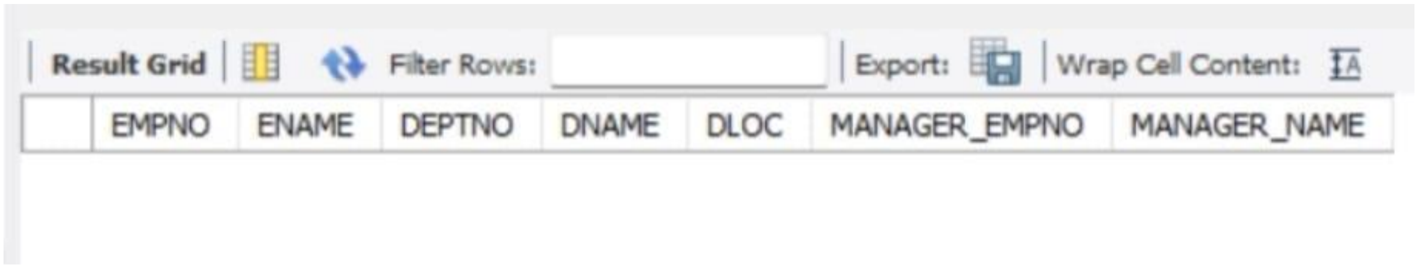
```
select e.empno, e.ename, e.sal, i.incentive_amount, i.incentive_date
from employee e
join incentives i on e.empno = i.empno
where i.incentive_date between '2019-01-01' and '2019-01-31'
  and i.incentive_amount = (
    select max(incentive_amount)
    from incentives
    where incentive_date between '2019-01-01' and '2019-01-31'
      and incentive_amount < (
        select max(incentive_amount)
        from incentives
        where incentive_date between '2019-01-01' and '2019-01-31'
      )
  );
```



EMPNO	ENAME	SAL	INCENTIVE_AMOUNT	INCENTIVE_DATE

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

```
 select e.empno, e.ename, e.deptno, d.dname, d.dloc, m.empno as manager_empno, m.ename as
manager_name
from employee e
join employee m on e.mgr_no = m.empno
join dept d on e.deptno = d.deptno
where e.deptno = m.deptno;
```

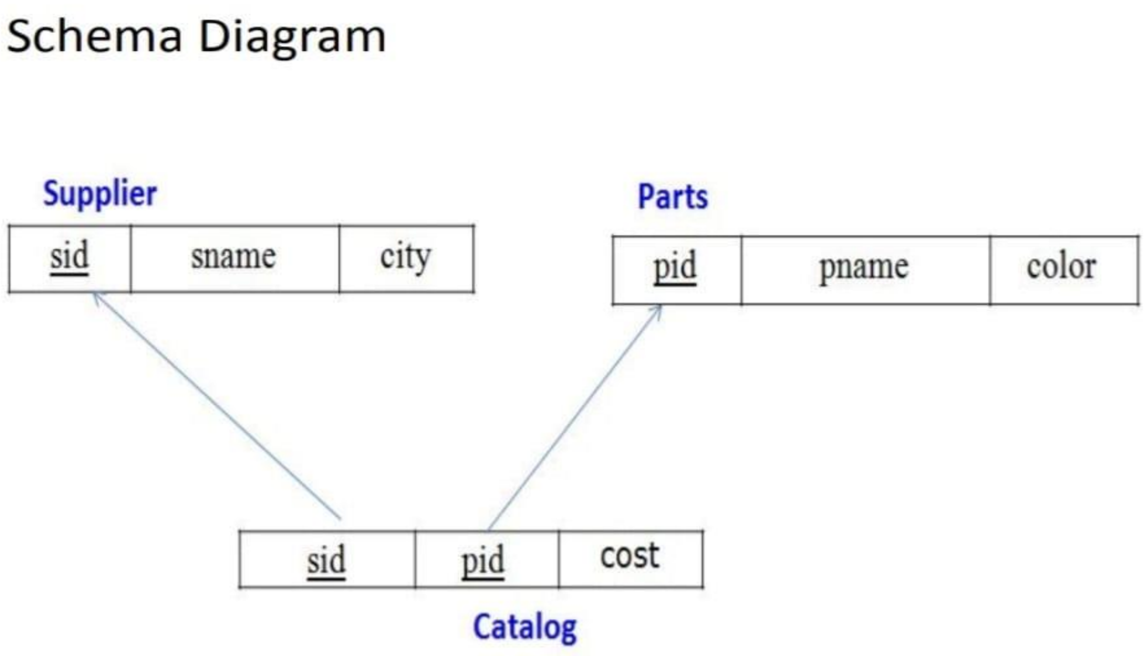


	EMPNO	ENAME	DEPTNO	DNAME	DLOC	MANAGER_EMPNO	MANAGER_NAME

Supplier Database

(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.



Create Database

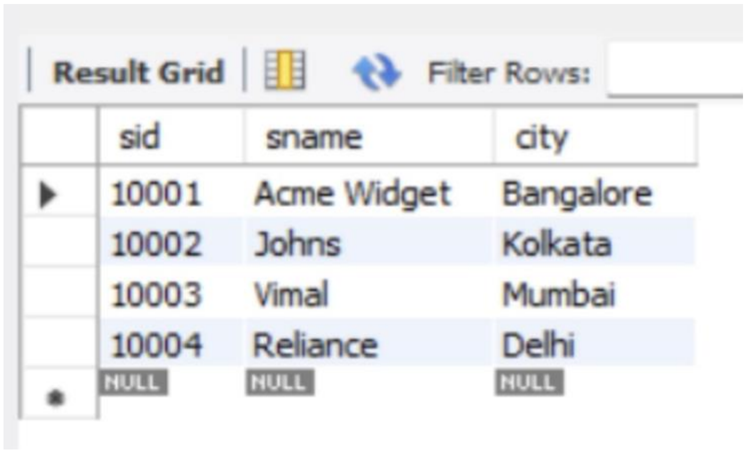
```
create database supplierdb;
use supplierdb;
```

Create Table

```
CREATE TABLE Supplier
(
  sid INT PRIMARY KEY,
  sname VARCHAR(30),
  city VARCHAR(30)
);

CREATE TABLE Parts
(
  pid INT PRIMARY KEY,
  pname VARCHAR(30),
  color VARCHAR(20)
);

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)
);
```

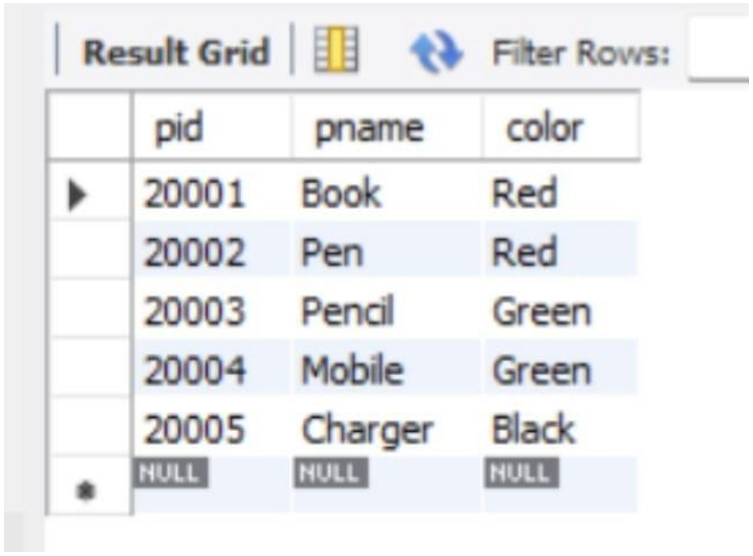
- **Insert appropriate records in each table.**

```
INSERT INTO Supplier VALUES
(10001, 'Acme Widget', 'Bangalore'),
(10002, 'Johns', 'Kolkata'),
(10003, 'Vimal', 'Mumbai'),
(10004, 'Reliance', 'Delhi');

INSERT INTO Parts VALUES
(20001, 'Book', 'Red'),
(20002, 'Pen', 'Red'),
(20003, 'Pencil', 'Green'),
(20004, 'Mobile', 'Green'),
(20005, 'Charger', 'Black');

INSERT INTO Catalog 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 supplierdb.catalog;
```



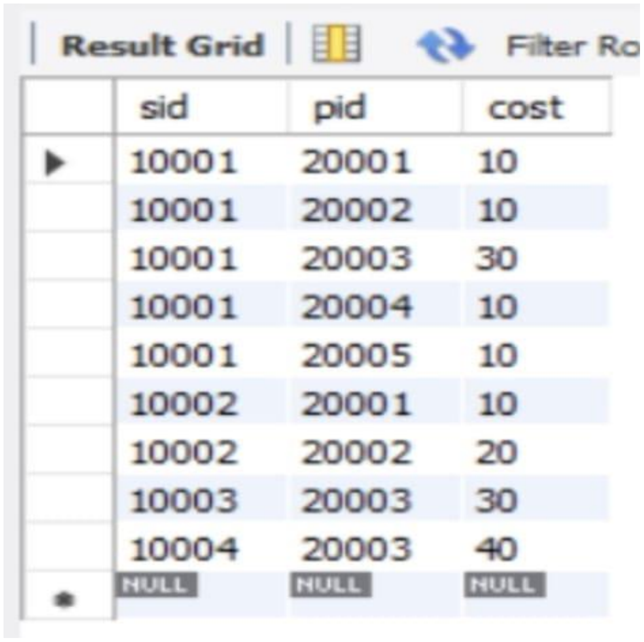
sid	sname	city
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi
NULL	NULL	NULL

select * from supplierdb.parts;


Result Grid | Filter Rows:

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

select * from supplierdb.supplier;

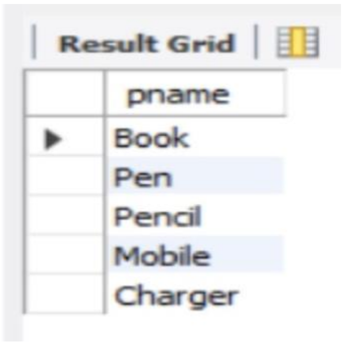

Result Grid | Filter Ro

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

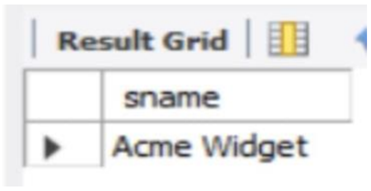


Result Grid

```
pname
Book
Pen
Pencil
Mobile
Charger
```

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

  select s.sname from supplier s where not exists (select p.pid from parts p where not exists (select * from catalog c where c.sid = s.sid and c.pid = p.pid));

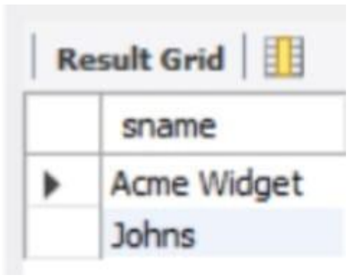


Result Grid

```
sname
Acme Widget
```

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

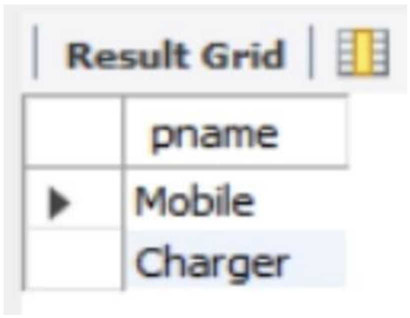
select s.sname from supplier s where not exists (select p.pid from parts p where p.color = 'red' and not exists (select * from catalog c where c.sid = s.sid and c.pid = p.pid));



Result Grid

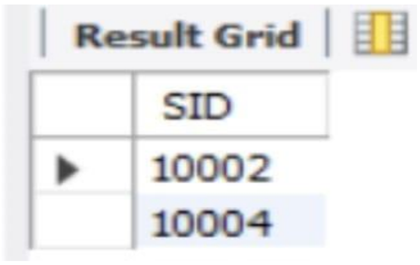
```
sname
Acme Widget
Johns
```

- **Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.**
  select p.pname from parts p where p.pid in (select c.pid from catalog c join supplier s on s.sid = c.sid where s.sname = 'acme widget')
  and p.pid not in
  (
    select c.pid
    from catalog c
    join supplier s on s.sid = c.sid
    where s.sname <> 'acme widget'
  );



	pname
▶	Mobile
	Charger

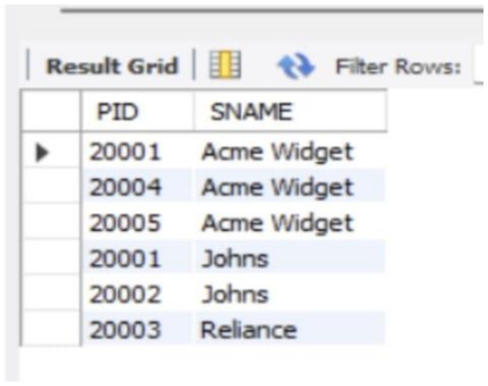
- **Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).**
  select distinct c.sid from catalog c where c.cost>(select avg(c2.cost)from catalog c2 where c2.pid=c.pid);



	SID
▶	10002
	10004

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

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



PID	SNAME
20001	Acme Widget
20004	Acme Widget
20005	Acme Widget
20001	Johns
20002	Johns
20003	Reliance

More Queries on
Supplier Database

(Week 8)

1. Find the most expensive part overall and the supplier who supplies it.

2. Find suppliers who do NOT supply any red parts.

3. Show each supplier and total value of all parts they supply.

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

5. List suppliers who offer the cheapest cost for each part.

6. Create a view showing suppliers and the total number of parts they supply.

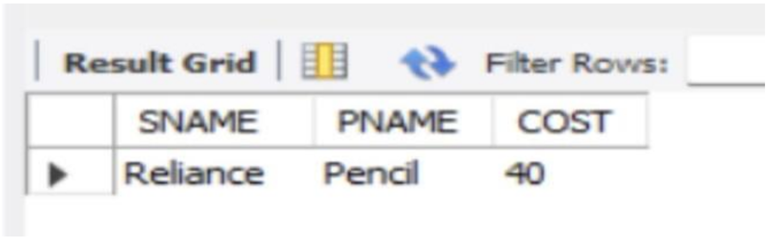
7. Create a view of the most expensive supplier for each part.

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

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

- **Find the most expensive part overall and the supplier who supplies it.**

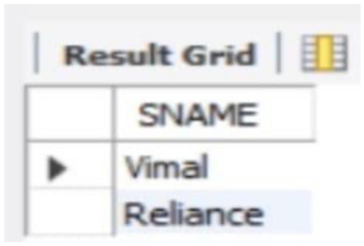
select s.sname, p.pname, c.cost from catalog c join supplier s on c.sid = s.sid join parts p on c.pid = p.pid where c.cost = (select max(cost) from catalog);



	SNAME	PNAME	COST
▶	Reliance	Pencil	40

- **Find suppliers who do NOT supply any red parts.**

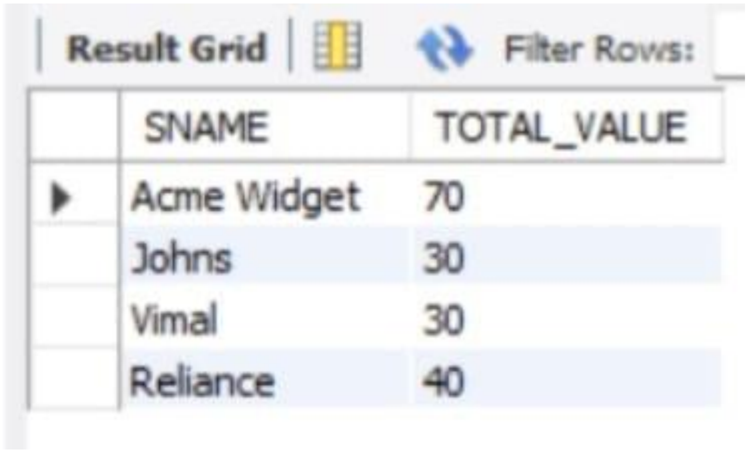
select s.sname from supplier s where s.sid not in (select c.sid from catalog c join parts p on c.pid = p.pid where p.color = 'red');



	SNAME
▶	Vimal
	Reliance

- **Show each supplier and total value of all parts they supply.**

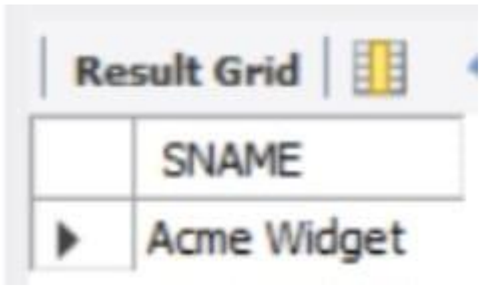
select s.sname, sum(c.cost) as total_value from supplier s left join catalog c on s.sid = c.sid group by s.sname;



	SNAME	TOTAL_VALUE
▶	Acme Widget	70
	Johns	30
	Vimal	30
	Reliance	40

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

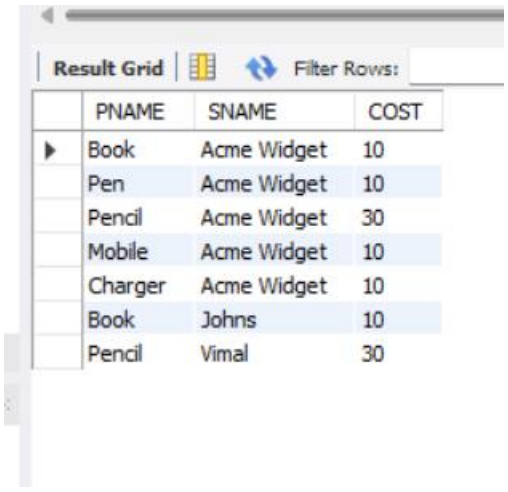
select s.sname from supplier s join catalog c on s.sid = c.sid where c.cost < 20 group by s.sname having count(c.pid) >= 2;



SNAME
Acme Widget

- **List suppliers who offer the cheapest cost for each part.**

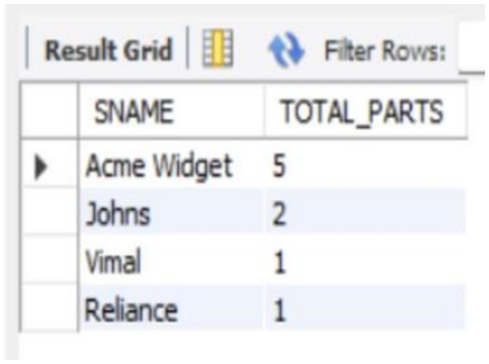
select p.pname, s.sname, c.cost from catalog c join parts p on c.pid = p.pid join supplier s on c.sid = s.sid where c.cost = (select min(c2.cost) from catalog c2 where c2.pid = c.pid);



PNAME	SNAME	COST
Book	Acme Widget	10
Pen	Acme Widget	10
Pencil	Acme Widget	30
Mobile	Acme Widget	10
Charger	Acme Widget	10
Book	Johns	10
Pencil	Vimal	30

- **Create a view showing suppliers and the total number of parts they supply.**

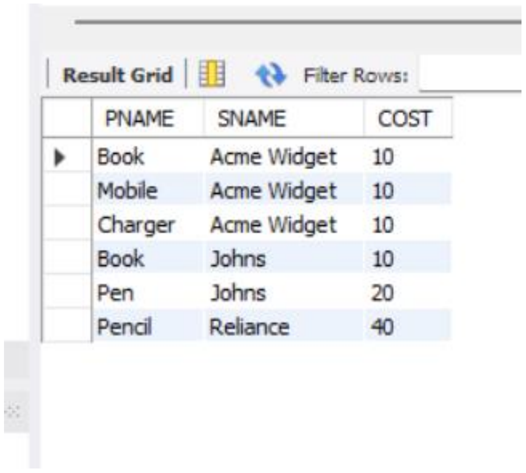
create or replace view supplierpartcount as select s.sname, count(c.pid) as total_parts from supplier s left join catalog c on s.sid = c.sid group by s.sname;
select *from supplierpartcount;



SNAME	TOTAL_PARTS
Acme Widget	5
Johns	2
Vimal	1
Reliance	1

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

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



PNAME	SNAME	COST
Book	Acme Widget	10
Mobile	Acme Widget	10
Charger	Acme Widget	10
Book	Johns	10
Pen	Johns	20
Pencil	Reliance	40

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

```
drop trigger if exists preventlowcost;
drop trigger if exists defaultcost;
 delimiter $$
create trigger preventlowcost
before insert on catalog
for each row
begin
   if new.cost < 1 then
      signal sqlstate '45000'
      set message_text = 'cost cannot be less than 1';
   end if;
end$$
delimiter ;
insert into catalog values (1,1,0);
```

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

```
 delimiter $$
create trigger defaultcost
before insert on catalog
for each row
begin
   if new.cost is null then
      set new.cost = 100;
   end if;
end$$
delimiter ;
insert into catalog (sid, pid) values (1,4);
select * from catalog where sid=1 and pid=4;
```

NoSQL Restaurant Database

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

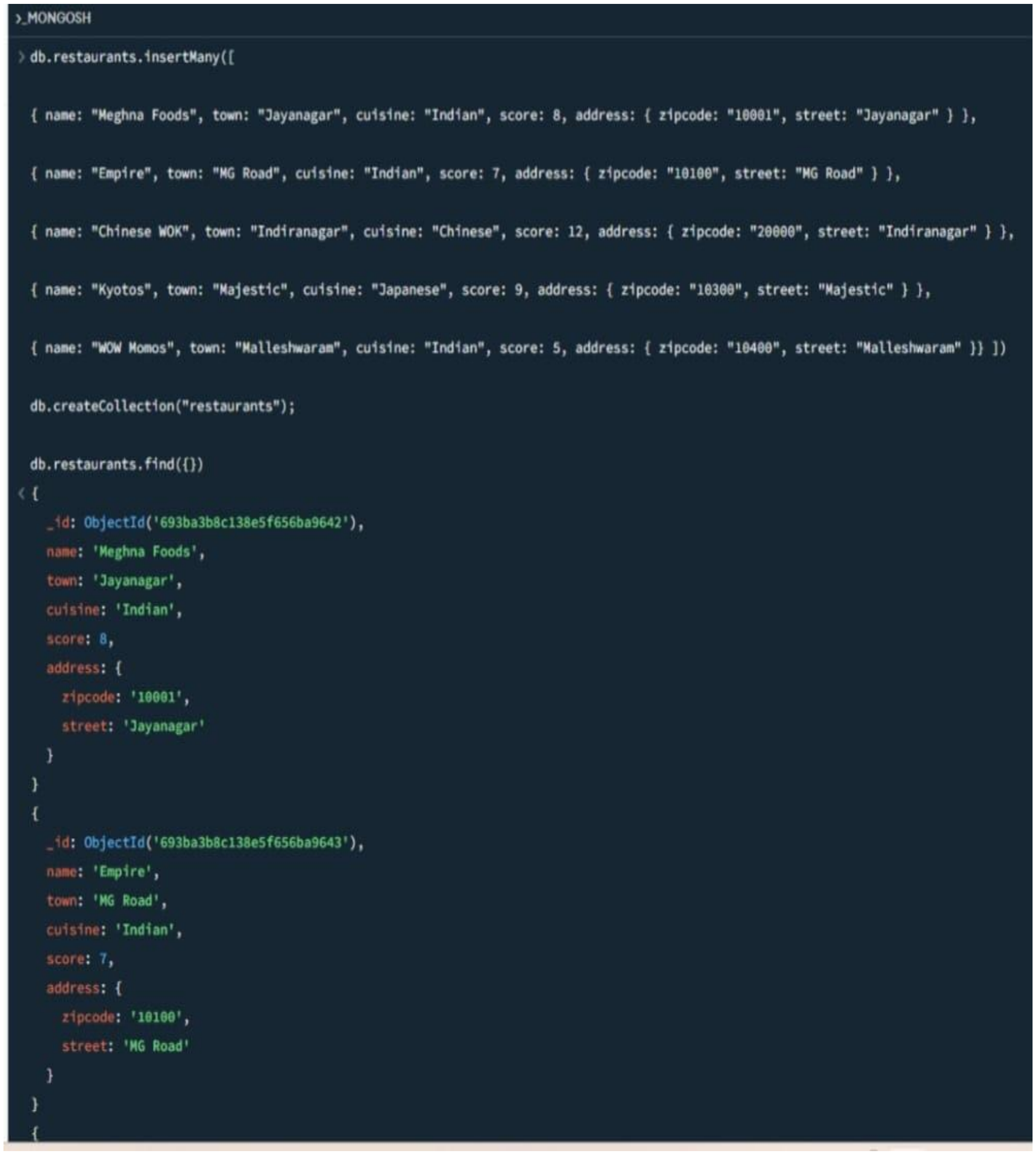
2.Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

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.

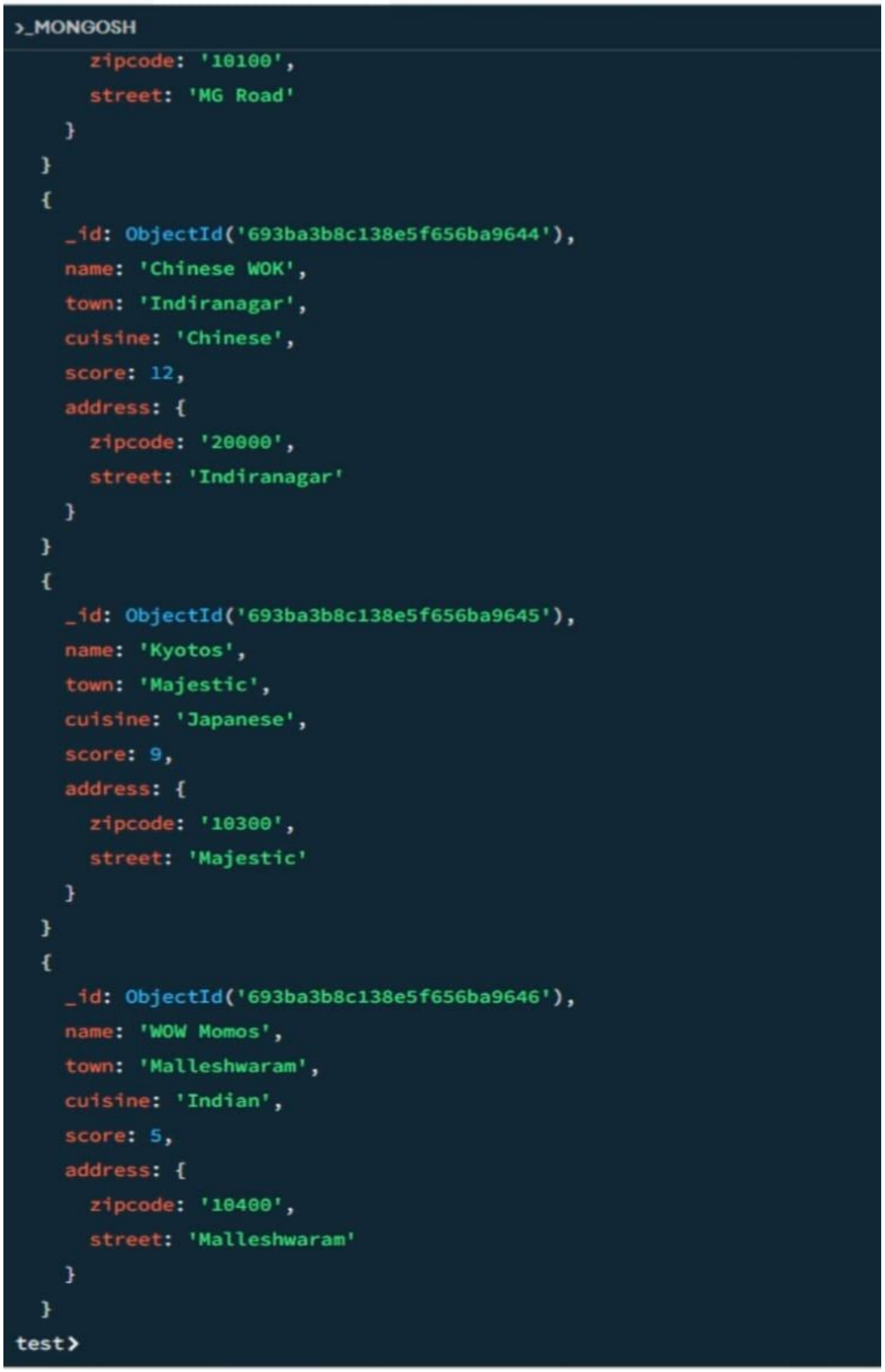
4.Write a MongoDB query to find the average score for each restaurant.

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

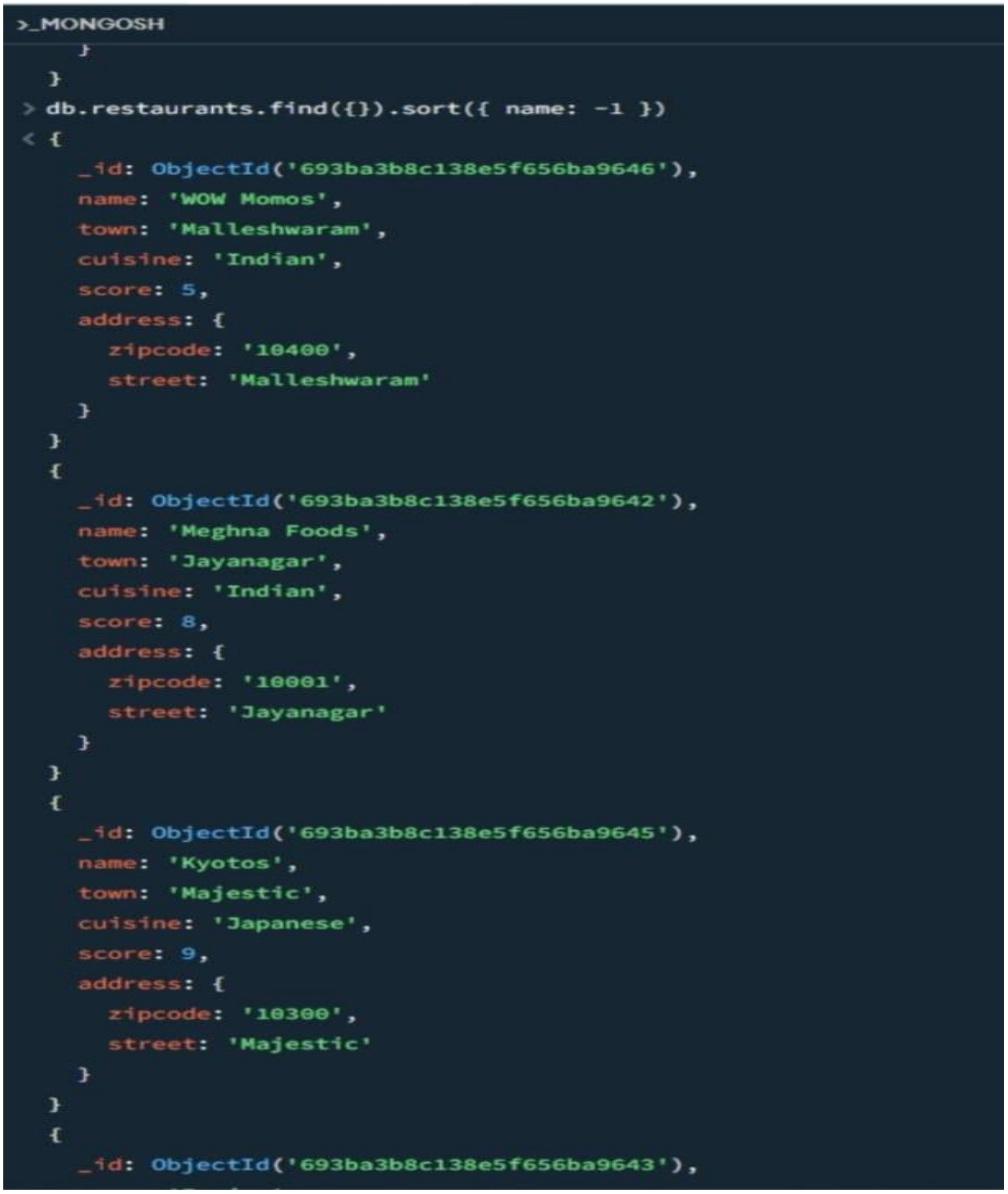
- **Write a MongoDB query to display all the documents in the collection restaurants.**



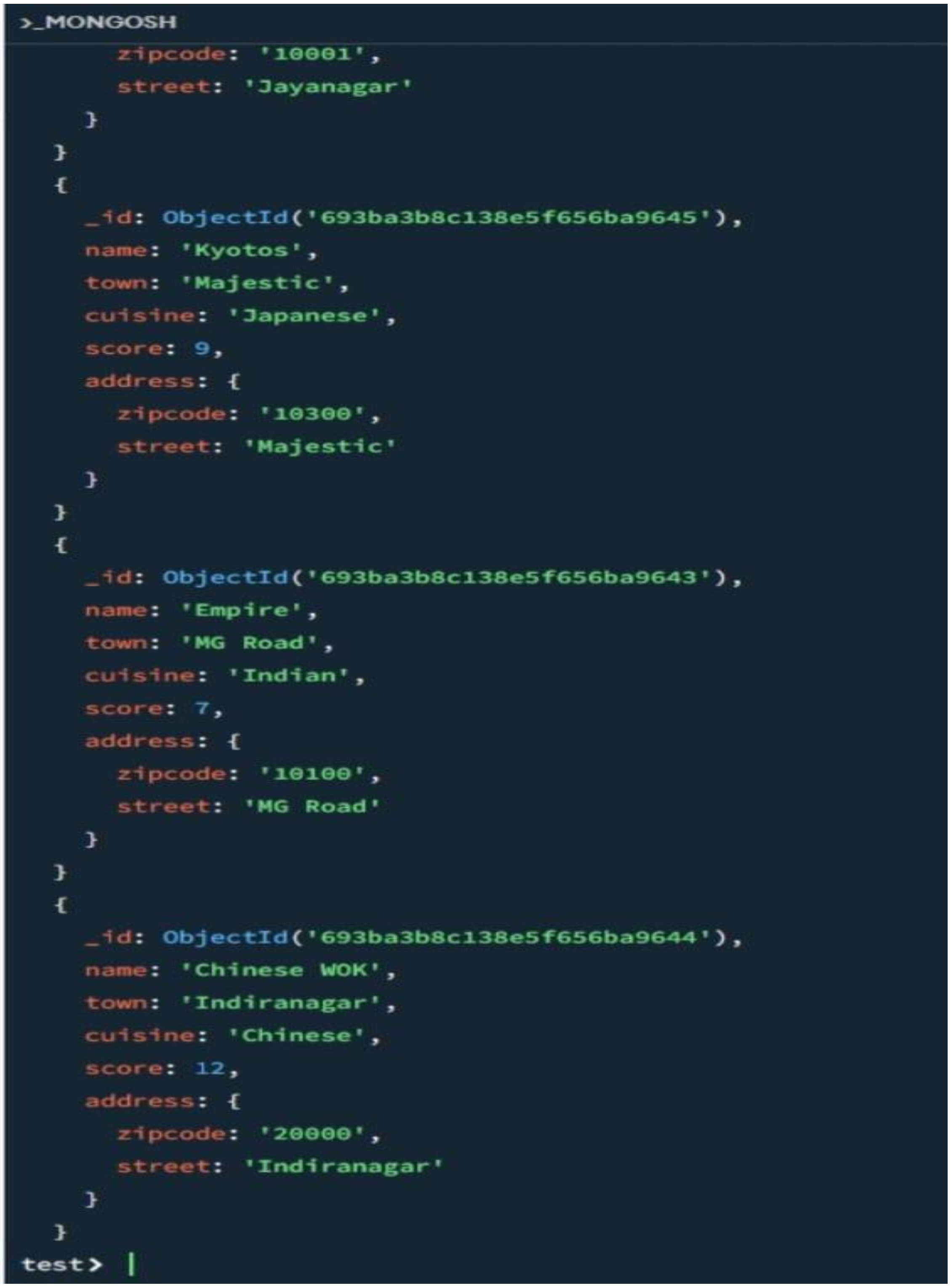
```
>_MONGOSH
> db.restaurants.insertMany([

  { 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" }} ])

  db.createCollection("restaurants");

  db.restaurants.find({})
< {
    _id: ObjectId('693ba3b8c138e5f656ba9642'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: {
      zipcode: '10001',
      street: 'Jayanagar'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9643'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: {
      zipcode: '10100',
      street: 'MG Road'
    }
  }
  {
```


```
>_MONGOSH
      zipcode: '10100',
      street: 'MG Road'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9644'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: {
      zipcode: '20000',
      street: 'Indiranagar'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9645'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: {
      zipcode: '10300',
      street: 'Majestic'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9646'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: {
      zipcode: '10400',
      street: 'Malleshwaram'
    }
  }
test>
```

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

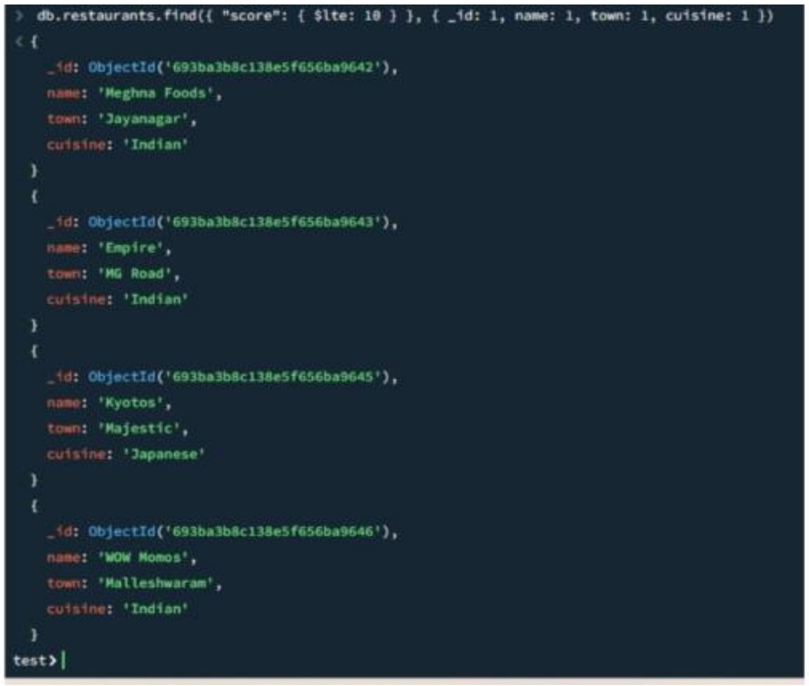


```
>_MONGOSH
    }
  }
> db.restaurants.find({}).sort({ name: -1 })
< {
    _id: ObjectId('693ba3b8c138e5f656ba9646'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: {
      zipcode: '10400',
      street: 'Malleshwaram'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9642'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: {
      zipcode: '10001',
      street: 'Jayanagar'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9645'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: {
      zipcode: '10300',
      street: 'Majestic'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9643'),
```



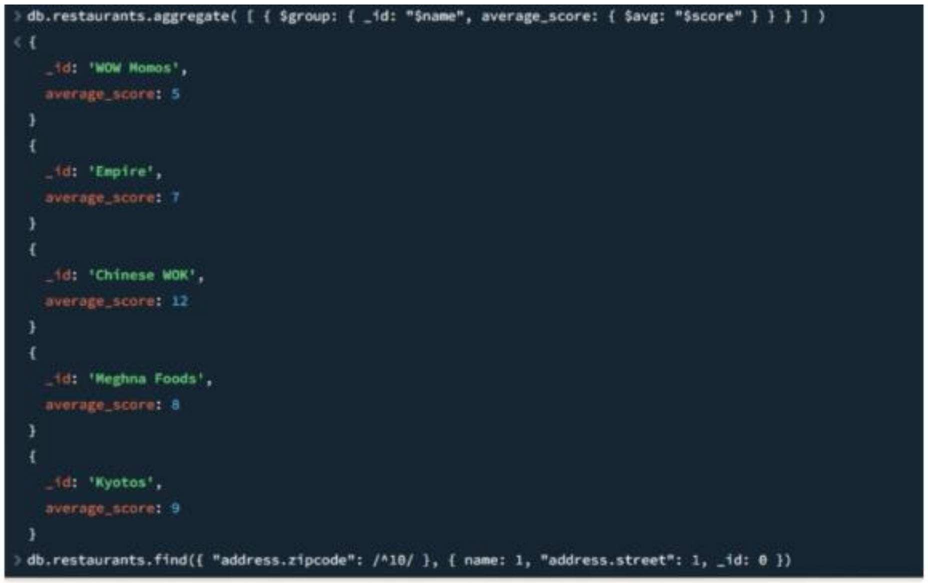
```
>_MONGOSH
      zipcode: '10001',
      street: 'Jayanagar'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9645'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: {
      zipcode: '10300',
      street: 'Majestic'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9643'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: {
      zipcode: '10100',
      street: 'MG Road'
    }
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9644'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: {
      zipcode: '20000',
      street: 'Indiranagar'
    }
  }
test> |
```

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



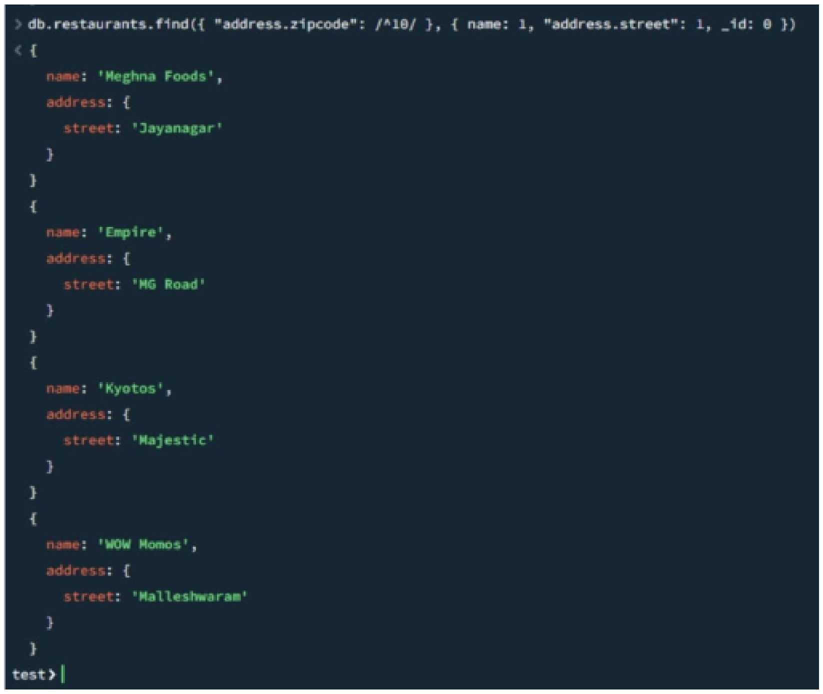
```
> db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })
< {
    _id: ObjectId('693ba3b8c138e5f656ba9642'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9643'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9645'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  }
  {
    _id: ObjectId('693ba3b8c138e5f656ba9646'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
test>
```

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



```
> db.restaurants.aggregate( [ { $group: { _id: "$name", average_score: { $avg: "$score" } } } ] )
< {
    _id: 'WOW Momos',
    average_score: 5
  }
  {
    _id: 'Empire',
    average_score: 7
  }
  {
    _id: 'Chinese WOK',
    average_score: 12
  }
  {
    _id: 'Meghna Foods',
    average_score: 8
  }
  {
    _id: 'Kyotos',
    average_score: 9
  }
> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
```

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



```
> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
< {
    name: 'Meghna Foods',
    address: {
      street: 'Jayanagar'
    }
  }
  {
    name: 'Empire',
    address: {
      street: 'MG Road'
    }
  }
  {
    name: 'Kyotos',
    address: {
      street: 'Majestic'
    }
  }
  {
    name: 'WOW Momos',
    address: {
      street: 'Malleshwaram'
    }
  }
test>
```