VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

Gayathri S (24BECS417)

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 **Gayathri S (24BECS417),** 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 during the year 2022. 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.

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

Index

Insurance Database

Question

(Week 1)

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

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

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

- OWNS (driver_id: String, reg_num: String)

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

- Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation

- Display Accident date and location

- Update the damage amount to 25000 for the car with a specific reg_num (example 'K A031181') for which the accident report number was 12.

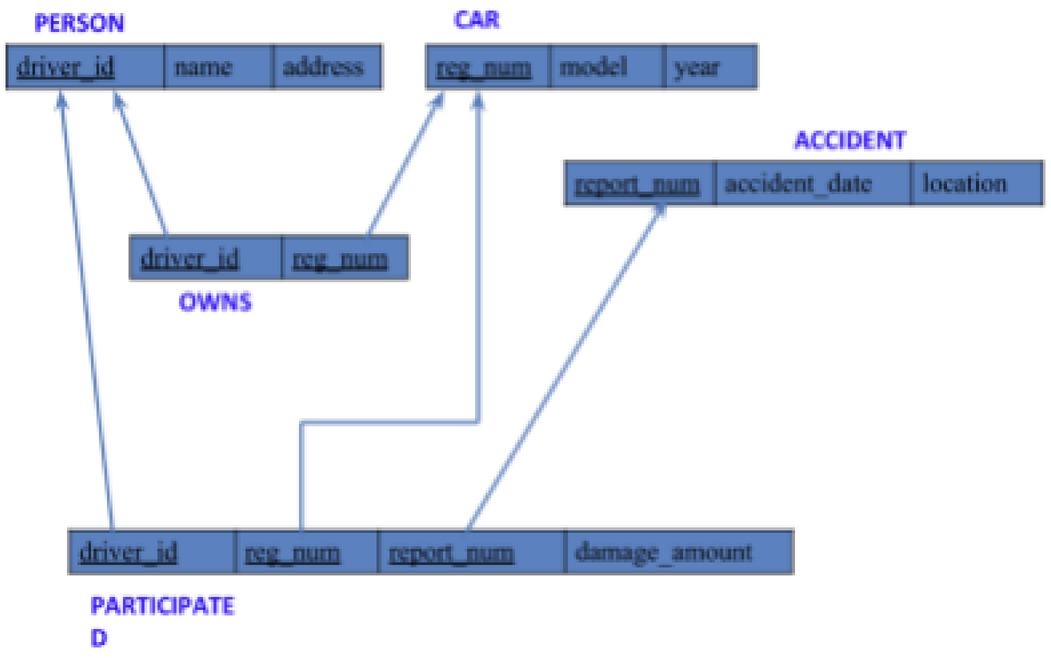
- Add a new accident to the database.

- 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 insurances_421;
use insurances_421;

Create table

create table person(
driver_id varchar(3) **primary key,**
name varchar(20) **not null**,
address varchar(100)
);

create table car(
reg_no char(8) **primary key,**
model varchar(20),
year int(4) **not null**

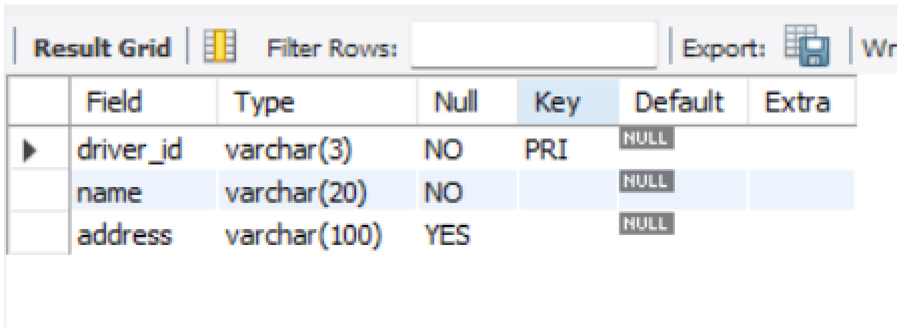
);

create table accident(
report_no int(4) **primary key,**
accident_date date,
location varchar(100)
);

create table owns(
driver_id varchar(3),
reg_no char(8),
foreign key(driver_id) **references** person(driver_id),
foreign key(reg_no) **references** car(reg_no)
);

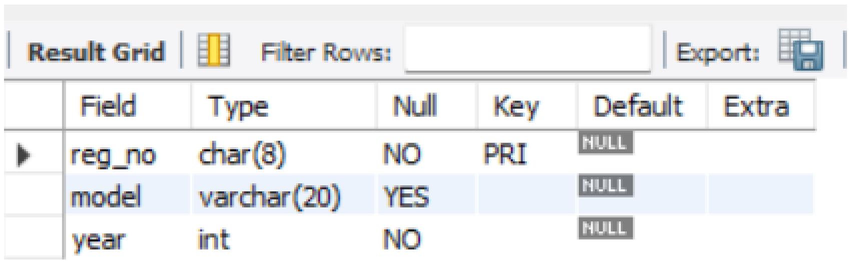
create table participated(
driver_id varchar(3),
reg_no char(8),
report_no int(4),
damage_amt int,
foreign **key**(driver_id) **references** person(driver_id),
foreign **key**(reg_no) **references** car(reg_no),
foreign **key** (report_no) **references** accident(report_no)
);

Person table :



Field	Type	Null	Key	Default	Extra
driver_id	varchar(3)	NO	PRI	NULL	
name	varchar(20)	NO		NULL	
address	varchar(100)	YES		NULL	

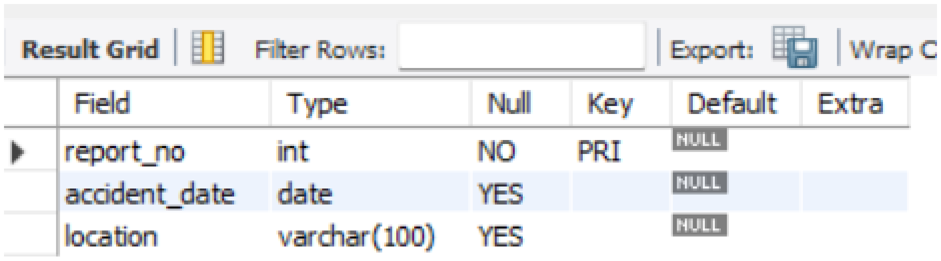
Car table :



Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
reg_no	char(8)	NO	PRI	NULL	
model	varchar(20)	YES		NULL	
year	int	NO		NULL	

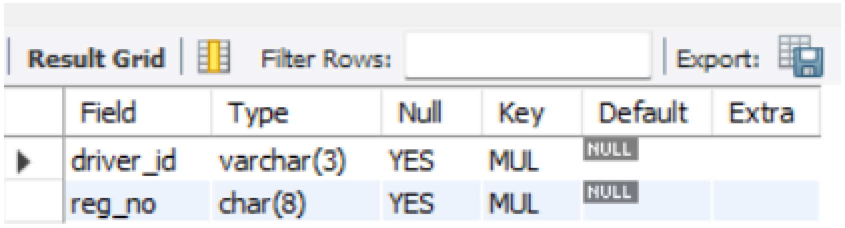
Accident table :



Result Grid | Filter Rows: | Export: | Wrap C

Field	Type	Null	Key	Default	Extra
report_no	int	NO	PRI	NULL	
accident_date	date	YES		NULL	
location	varchar(100)	YES		NULL	

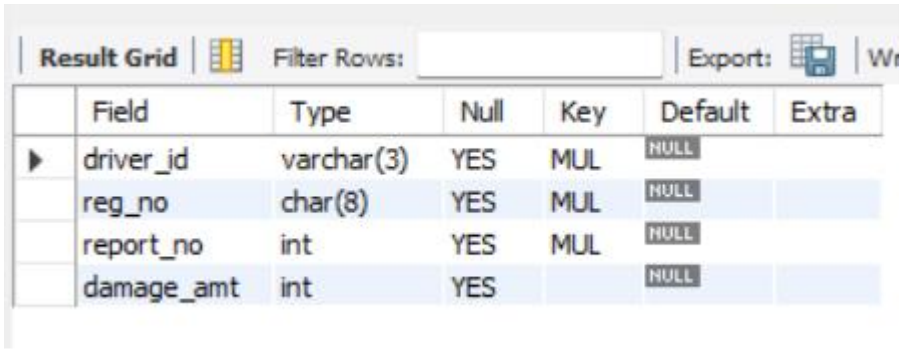
Owns table :



Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
driver_id	varchar(3)	YES	MUL	NULL	
reg_no	char(8)	YES	MUL	NULL	

Participated table :

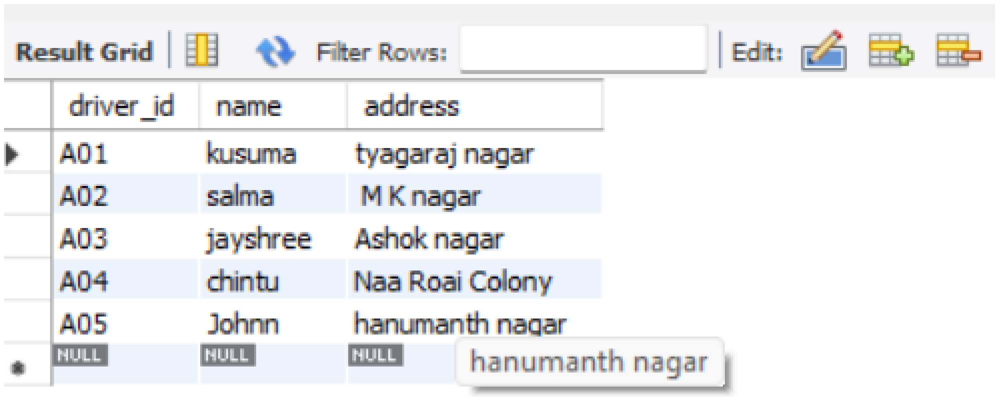


Result Grid | Filter Rows: | Export: | Wı

Field	Type	Null	Key	Default	Extra
driver_id	varchar(3)	YES	MUL	NULL	
reg_no	char(8)	YES	MUL	NULL	
report_no	int	YES	MUL	NULL	
damage_amt	int	YES		NULL	

Inserting Values into the table

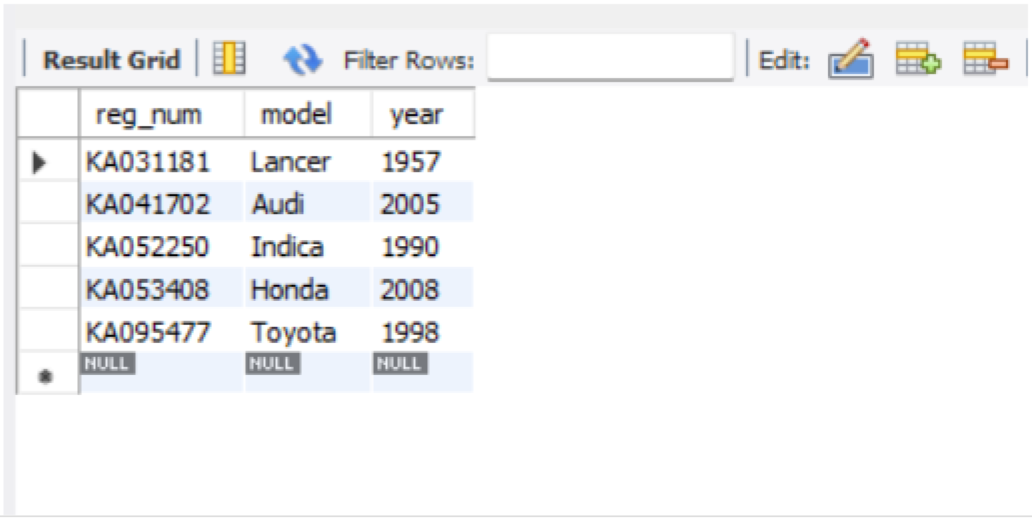
insert into person values("A01","kusuma", "tyagaraj nagar");
insert into person values("A02","salma", " M K nagar");
insert into person values("A03","jayshree", "Ashok nagar");
insert into person values("A04","chintu", "Naa Roai Colony");
insert into person values("A05","Johnn", "hanumanth nagar");
select * from person;



driver_id	name	address
A01	kusuma	tyagaraj nagar
A02	salma	M K nagar
A03	jayshree	Ashok nagar
A04	chintu	Naa Roai Colony
A05	Johnn	hanumanth nagar
NULL	NULL	NULL

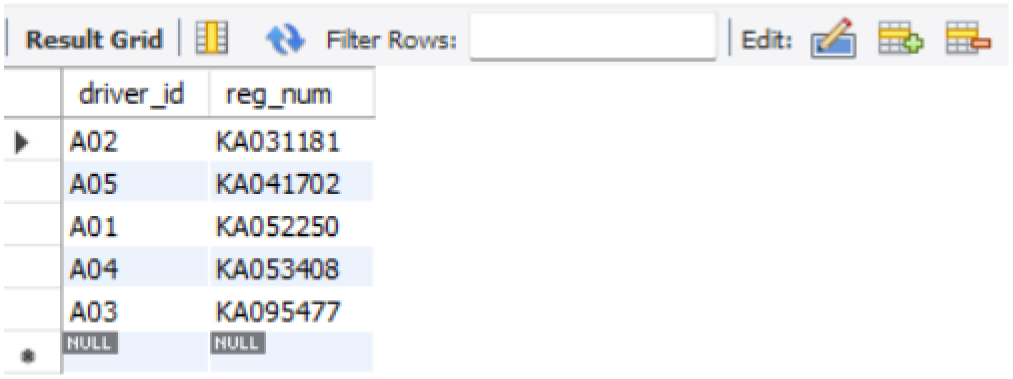
hanumanth nagar

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;



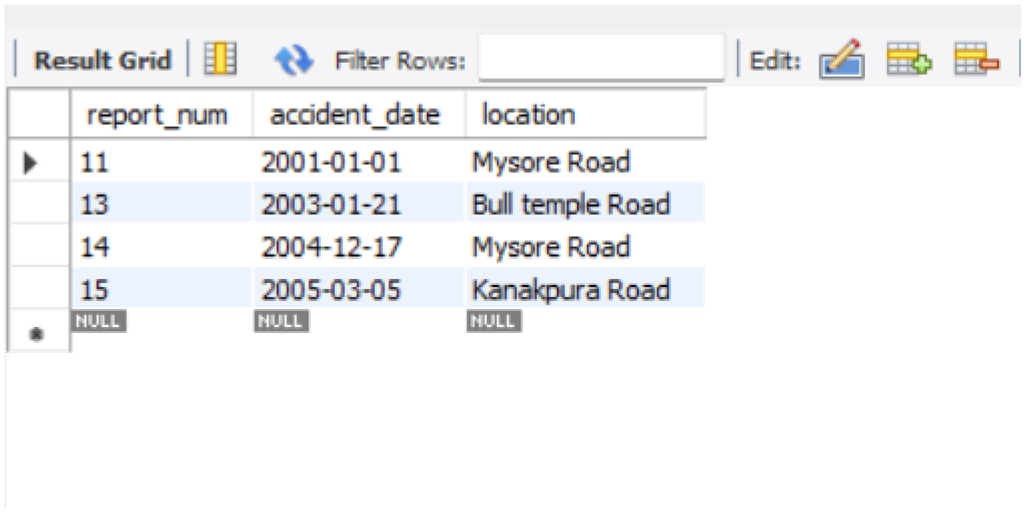
reg_num	model	year
KA031181	Lancer	1957
KA041702	Audi	2005
KA052250	Indica	1990
KA053408	Honda	2008
KA095477	Toyota	1998
NULL	NULL	NULL

insert into owns values("A01","KA052250");
insert into owns values("A02","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
NULL	NULL

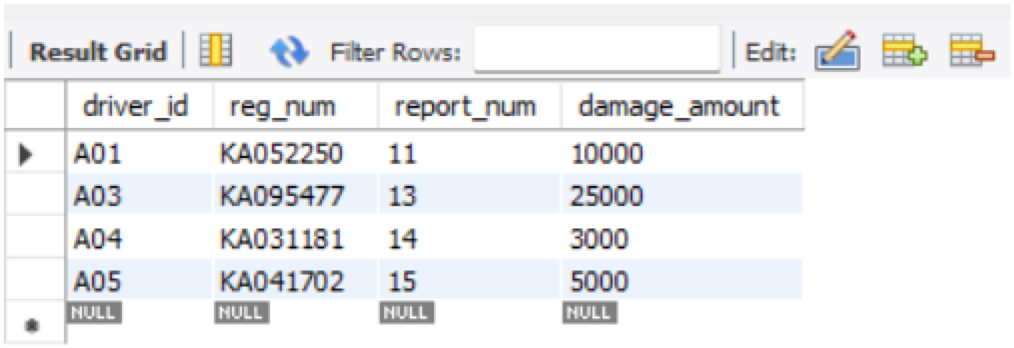
insert into accident values(11,'2001-01-01',"Mysore Road");
insert into accident values(12,'2002-02-02',"South end Circle");
insert into accident values(13,'2003-01-21',"Bull temple Road");
insert into accident values(14,'2004-12-17',"Mysore Road");
insert into accident values(15,'2005-03-05',"Kanakpura Road");
select * from accident;



report_num	accident_date	location
11	2001-01-01	Mysore Road
13	2003-01-21	Bull temple Road
14	2004-12-17	Mysore Road
15	2005-03-05	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;



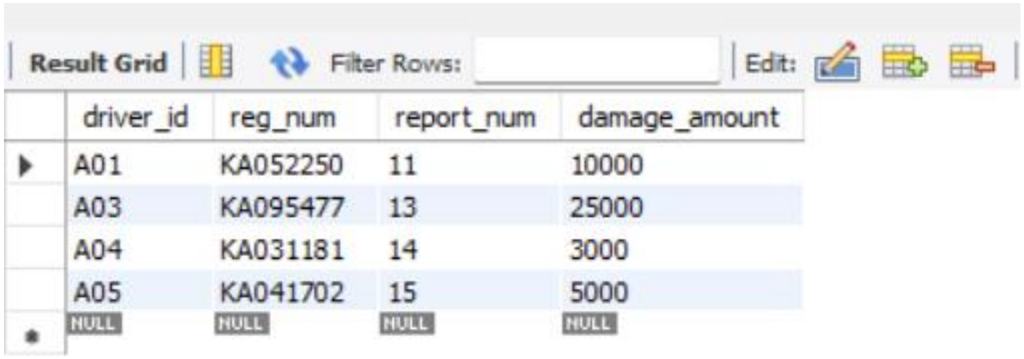
driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000
NULL	NULL	NULL	NULL

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;

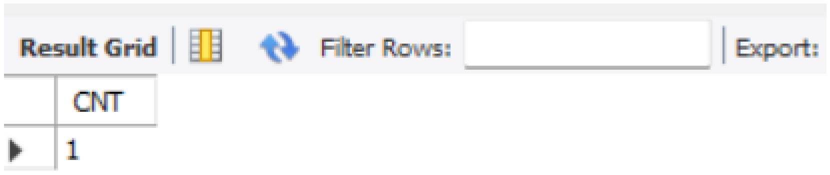
select * from participated;



driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000
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 '2001%';



Result Grid | Filter Rows: | Export:

```
CNT
1
```

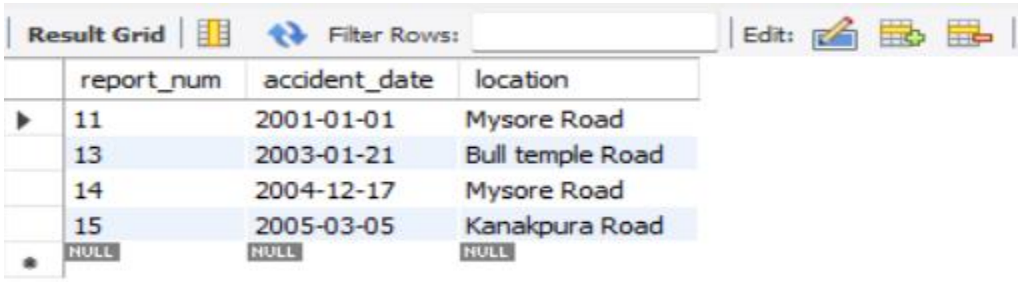
- Add a new accident to the database.
insert into accident values(16,'2003-03-08',"Domlur");

select * from accident;

<u>TO DO:</u>

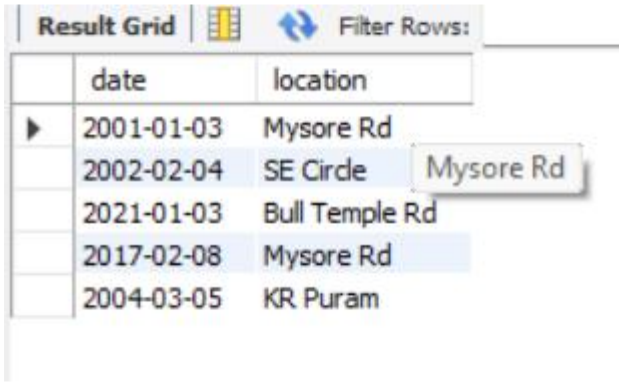
● DISPLAY ACCIDENT DATE AND LOCATION

select accident_date **as** date, location **from** accident;



Result Grid | Filter Rows: | Edit:

```
report_num  accident_date  location
11          2001-01-01     Mysore Road
13          2003-01-21     Bull temple Road
14          2004-12-17     Mysore Road
15          2005-03-05     Kanakpura Road
NULL        NULL           NULL
```

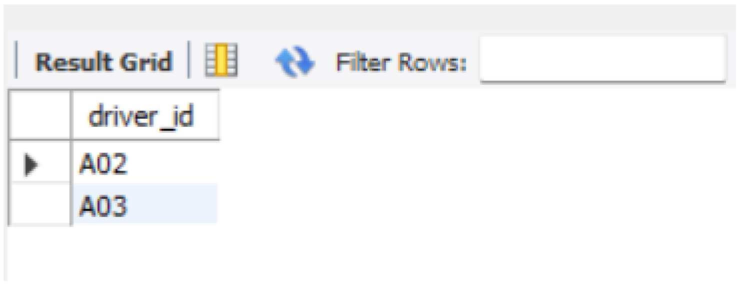

Result Grid | Filter Rows:

date	location
2001-01-03	Mysore Rd
2002-02-04	SE Circle
2021-01-03	Bull Temple Rd
2017-02-08	Mysore Rd
2004-03-05	KR Puram

Mysore Rd

● DISPLAY DRIVER ID WHO DID ACCIDENT WITH DAMAGE AMOUNT GREATER THAN OR EQUAL TO RS.25000

Select participated.driver_id as driver_id from accident,participated **where** accident.report_no = participated.report_no and participated.damage_amt >= 25000;


Result Grid | Filter Rows:

driver_id
A02
A03

More Queries on Insurance Database

Question

(Week 2)

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

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

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

- OWNS (driver_id: String, reg_num: String)

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

- Create the above tables by properly specifying the primary keys and the foreign keys. - 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 A031181') for which the accident report number was 12.

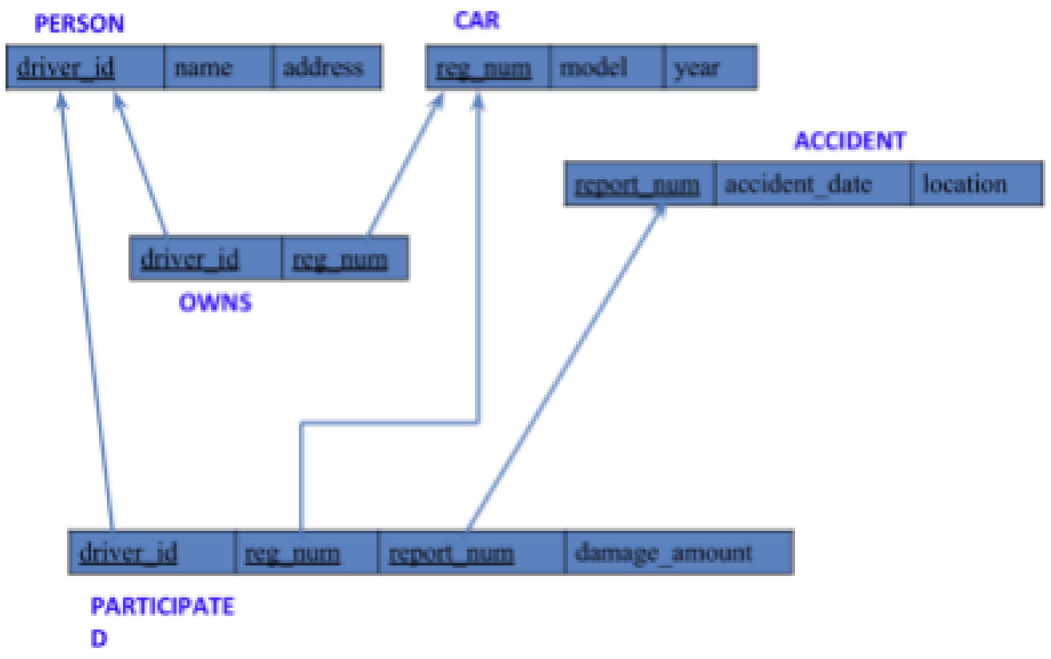
- Add a new accident to the database.

- To Do

- Display Accident date and location

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

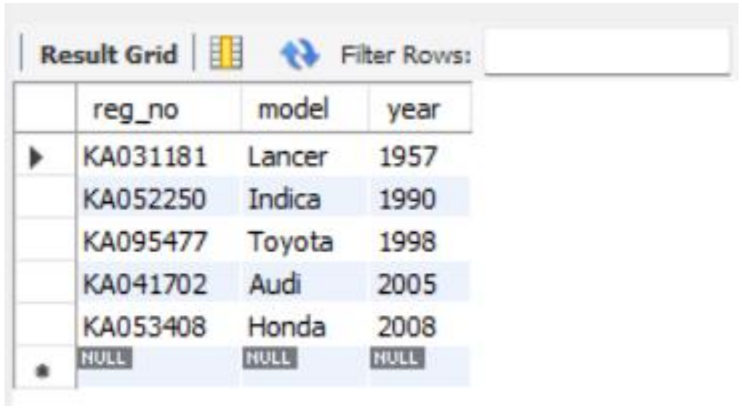
Schema Diagram



Queries

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

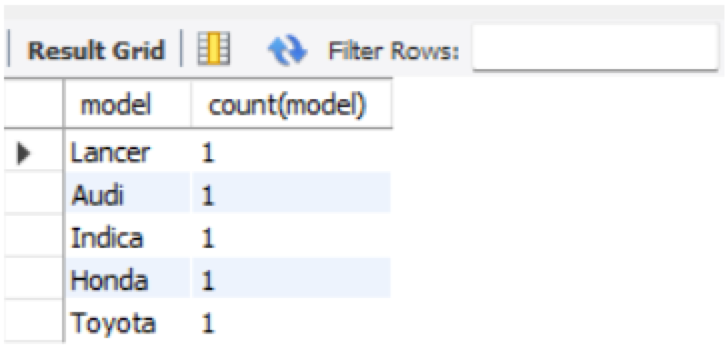
select * from car **order by** year asc;



reg_no	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 model, **count**(**model**) **from** participated, car **where** participated.reg_no = car.reg_no **group by** model;

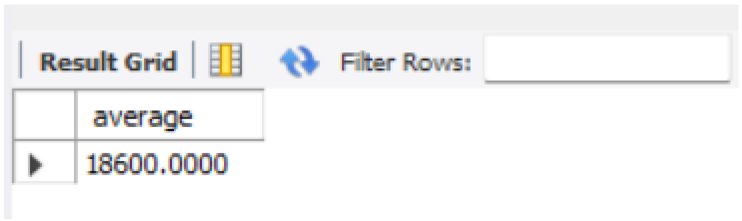


model	count(model)
Lancer	1
Audi	1
Indica	1
Honda	1
Toyota	1

<u>TO DO:</u>

● FIND THE AVERAGE DAMAGE AMOUNT

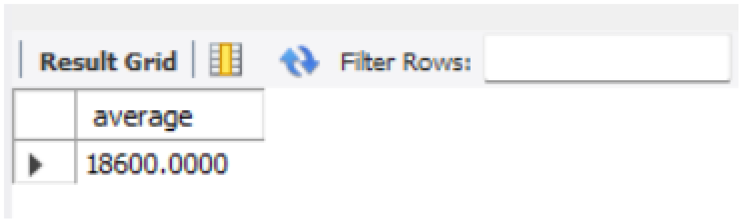
select avg(damage_amout) **as** average **from** participated;



average
18600.0000

● DELETE THE TUPLE WHOSE DAMAGE AMOUNT IS BELOW THE AVERAGE DAMAGE AMOUNT

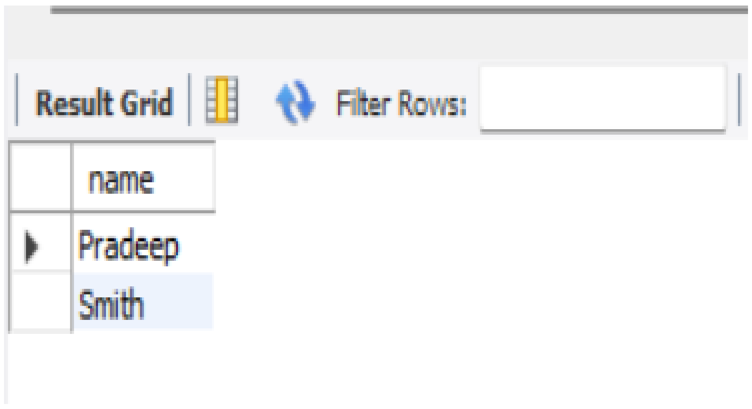
delete from participated **where** damage_amt < (select * from (select **avg**(damage_amount) **from** participated) **as** average);



average
18600.0000

● LIST THE NAME OF DRIVERS WHOSE DAMAGE IS GREATER THAN THE AVERAGE DAMAGE AMOUNT.

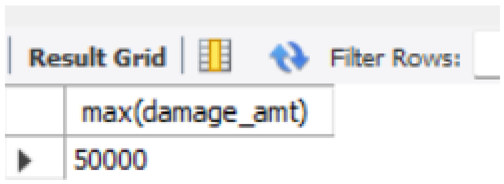
select name from person, participated **where** person.driver_id = participated.driver_id and participated.damage_amount > (select **avg**(damage_amount) from participated);



```
name
Pradeep
Smith
```

● FIND MAXIMUM DAMAGE AMOUNT.

select max(damage_amount) **from** participated;



```
max(damage_amt)
50000
```

Bank Database

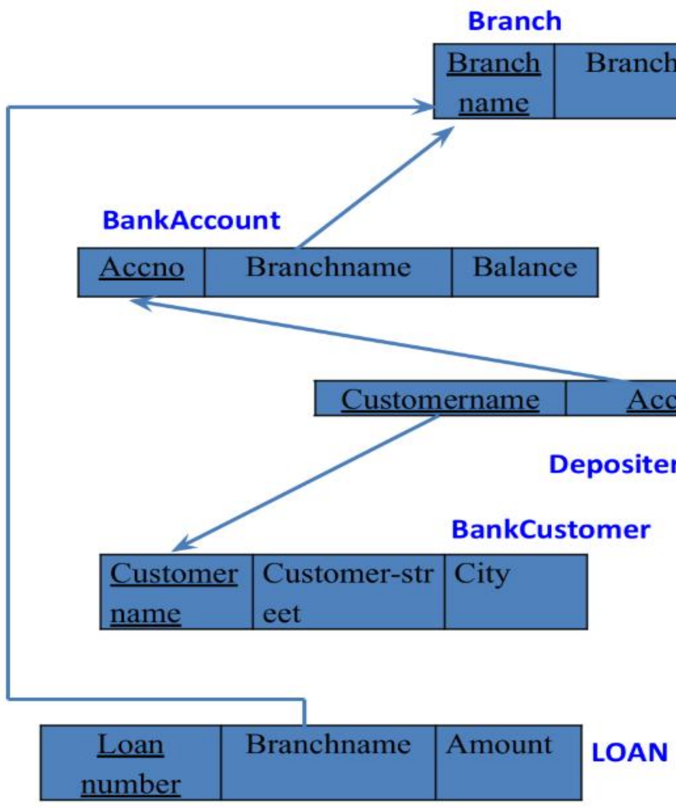
Question

(Week 3)

- Branch (branch-name: String, branch-city: String, assets: real)
- BankAccount(accno: int, branch-name: String, balance: real)
- BankCustomer (customer-name: String, customer-street: String, customer-city: String) - Depositer(customer-name: String, accno: int)
- LOAN (loan-number: int, branch-name: String, amount: real)

- Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation.
- Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
- Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).
- Create a view which gives each branch the sum of the amount of all the loans at the branch.

Schema Diagram



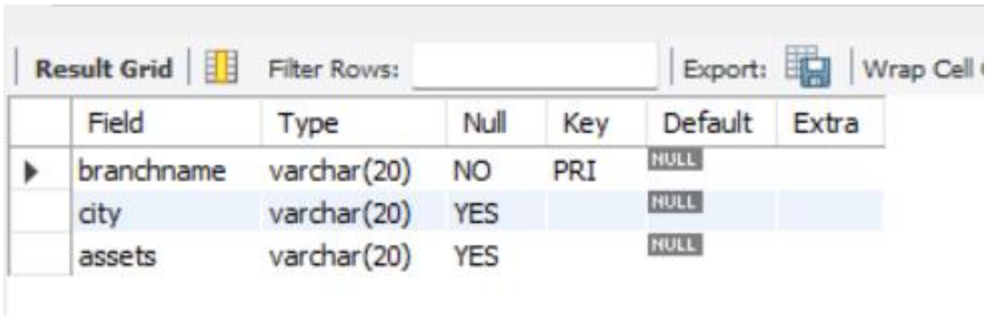
Create database

create database bank_421;
use bank_421;

Create table

create table Branch(
branchname varchar(20),
city varchar(20),
assets varchar(20),
primary key(branchname));

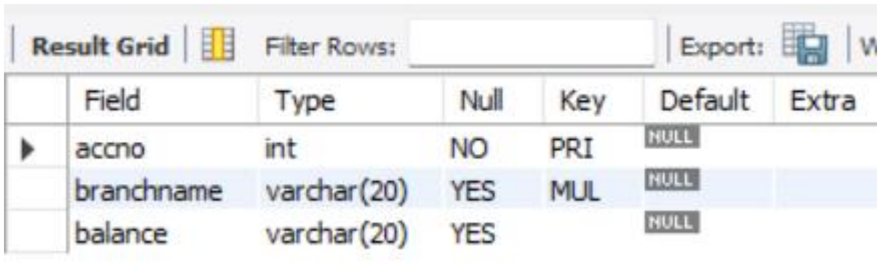
desc Branch;



Field	Type	Null	Key	Default	Extra
branchname	varchar(20)	NO	PRI	NULL	
city	varchar(20)	YES		NULL	
assets	varchar(20)	YES		NULL	

create table Bankaccount(
accno int,
branchname varchar(20),
balance varchar(20),
primary key (accno),
foreign key(branchname) **references** Branch(branchname));

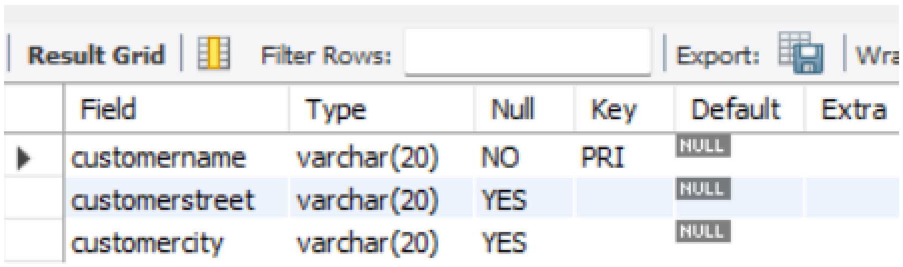
desc Bankaccount;



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

create table bankcustomer(
customername varchar(20),
customerstreet varchar(20),
customercity varchar(20),
primary key(customername));

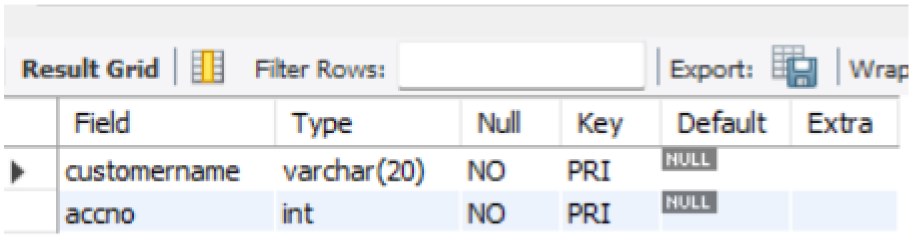
desc bankcustomer;



Field	Type	Null	Key	Default	Extra
customername	varchar(20)	NO	PRI	NULL	
customerstreet	varchar(20)	YES		NULL	
customercity	varchar(20)	YES		NULL	

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

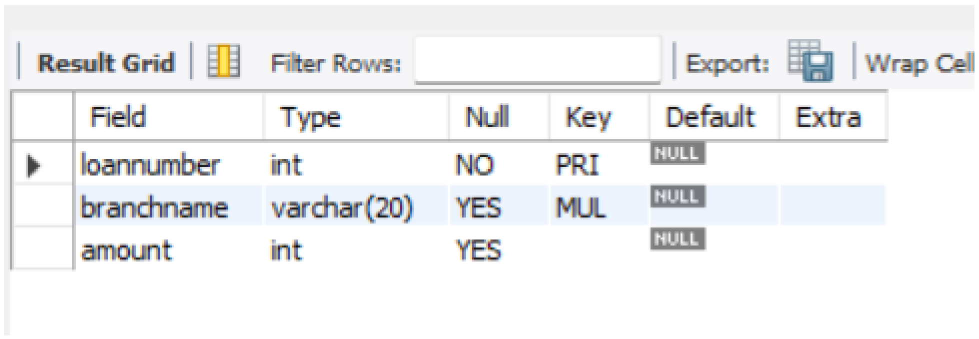
desc depositer;



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

create table loan(
loannumber int,
branchname varchar(20),
amount int,
primary key(loannumber),
foreign key(branchname)**references** Branch(branchname));

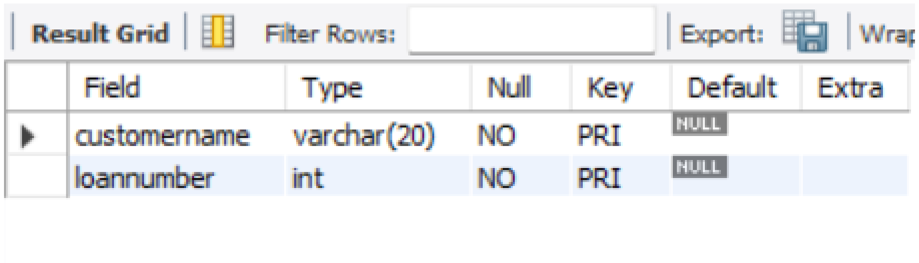
desc loan;



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

create table Borrower(
customername varchar(20),
loannumber int,
Primary key(customername,loannumber),
foreign key(loannumber)**references** loan_402(loannumber),
foreign key(customername) **references** bankcustomer_402(customername));

Desc Borrower;

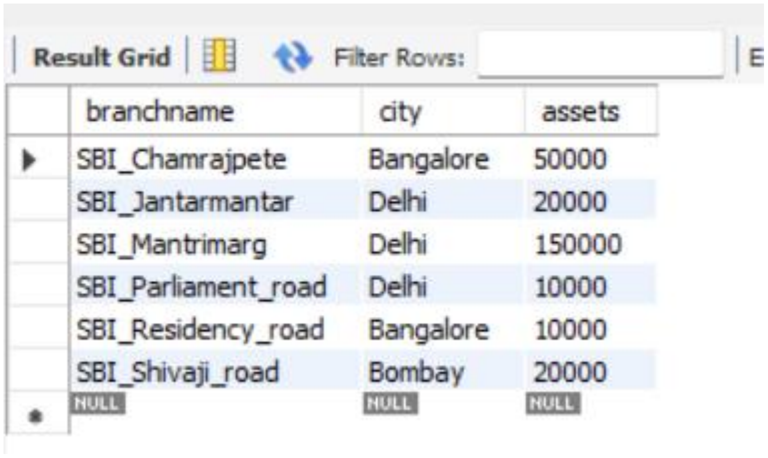


Result Grid | Filter Rows: | Export: | Wrap

Field	Type	Null	Key	Default	Extra
customername	varchar(20)	NO	PRI	NULL	
loannumber	int	NO	PRI	NULL	

Inserting the values

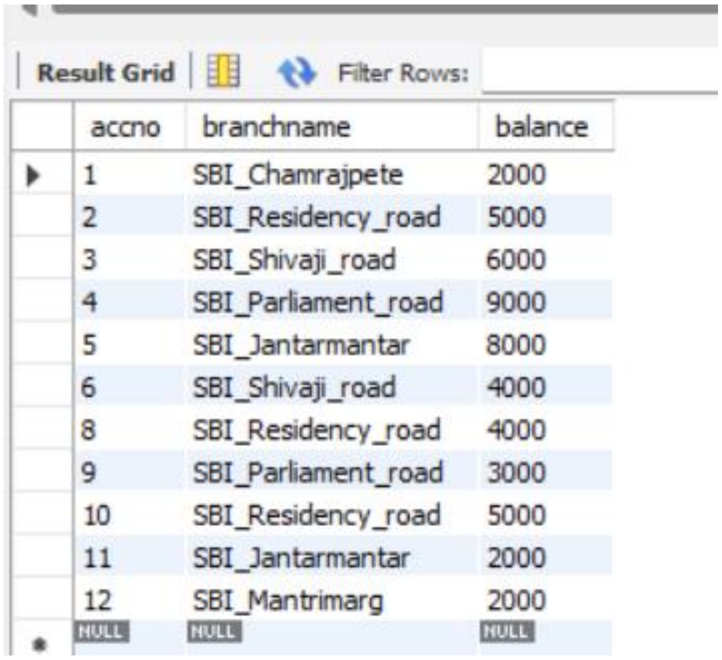
insert into Branch values('SBI_Chamrajpete', 'Bangalore', 50000);
insert into Branch values('SBI_Residency_road', 'Bangalore',10000);
insert into Branch values('SBI_Shivaji_road', 'Bombay', 20000);
insert into Branch values('SBI_Parliament_road','Delhi', 10000);
insert into Branch values('SBI_Jantarmantar', 'Delhi',20000);
insert into Branch values('SBI_Mantrimarg','Delhi',150000);
select * from Branch;



Result Grid | Filter Rows:

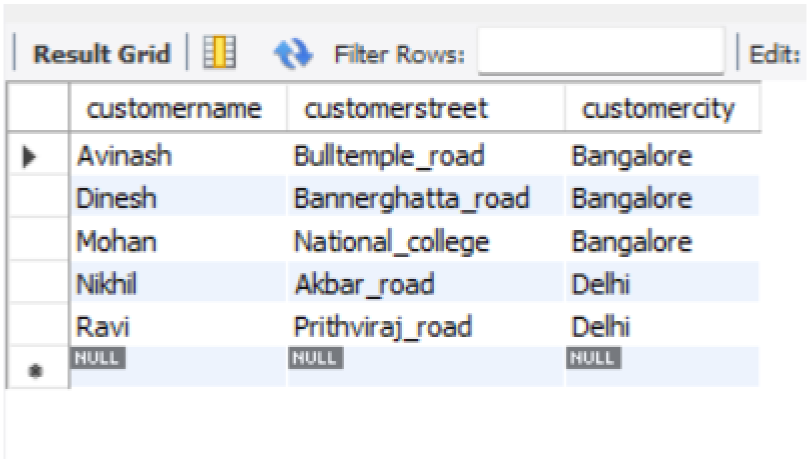
branchname	city	assets
SBI_Chamrajpete	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_Mantrimarg	Delhi	150000
SBI_Parliament_road	Delhi	10000
SBI_Residency_road	Bangalore	10000
SBI_Shivaji_road	Bombay	20000
NULL	NULL	NULL

```
insert into Bankaccount values(1, 'SBI_Chamrajpete',2000);
insert into Bankaccount values(2,'SBI_Residency_road', 5000);
insert into Bankaccount values(3,'SBI_Shivaji_road', 6000);
insert into Bankaccount values(4, 'SBI_Parliament_road', 9000);
insert into Bankaccount values(5, 'SBI_Jantarmantar', 8000);
insert into Bankaccount values(6, 'SBI_Shivaji_road', 4000);
insert into Bankaccount values(8, 'SBI_Residency_road', 4000);
insert into Bankaccount values(9, 'SBI_Parliament_road',  3000);
insert into Bankaccount values(10, 'SBI_Residency_road', 5000);
insert into Bankaccount values(11, 'SBI_Jantarmantar', 2000);
insert into Bankaccount values(12, 'SBI_Mantrimarg',2000);
select * from Bankaccount;
```



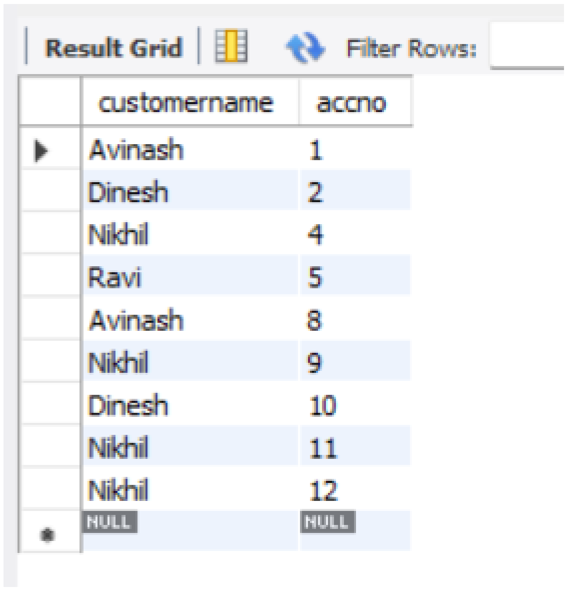
accno	branchname	balance
1	SBI_Chamrajpete	2000
2	SBI_Residency_road	5000
3	SBI_Shivaji_road	6000
4	SBI_Parliament_road	9000
5	SBI_Jantarmantar	8000
6	SBI_Shivaji_road	4000
8	SBI_Residency_road	4000
9	SBI_Parliament_road	3000
10	SBI_Residency_road	5000
11	SBI_Jantarmantar	2000
12	SBI_Mantrimarg	2000
NULL	NULL	NULL

```
insert into bankcustomer values('Avinash','Bulltemple_road','Bangalore');
insert into bankcustomer values('Dinesh', 'Bannerghatta_road','Bangalore');
insert into bankcustomer values('Mohan', 'National_college','Bangalore');
insert into bankcustomer values('Nikhil', 'Akbar_road', 'Delhi');
insert into bankcustomer values('Ravi', 'Prithviraj_road', 'Delhi');
select * from bankcustomer;
```



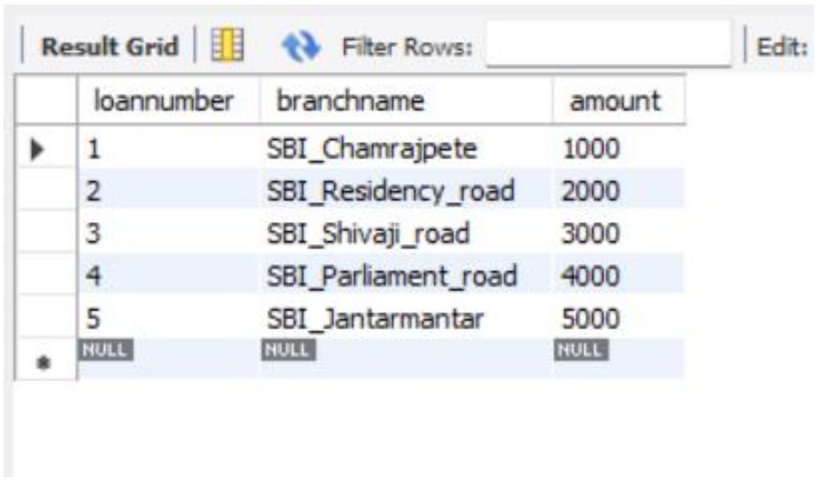
	customername	customerstreet	customercity
▶	Avinash	Bulltemple_road	Bangalore
	Dinesh	Bannerghatta_road	Bangalore
	Mohan	National_college	Bangalore
	Nikhil	Akbar_road	Delhi
	Ravi	Prithviraj_road	Delhi
*	NULL	NULL	NULL

```
insert into depositer values('Avinash' , 1);
insert into depositer values('Dinesh',2);
insert into depositer values('Nikhil',4);
insert into depositer values('Ravi', 5);
insert into depositer values('Avinash',8);
insert into depositer values('Nikhil', 9);
insert into depositer values('Dinesh',10);
insert into depositer values('Nikhil',11);
insert into depositer values('Nikhil',12);
select * from depositer;
```



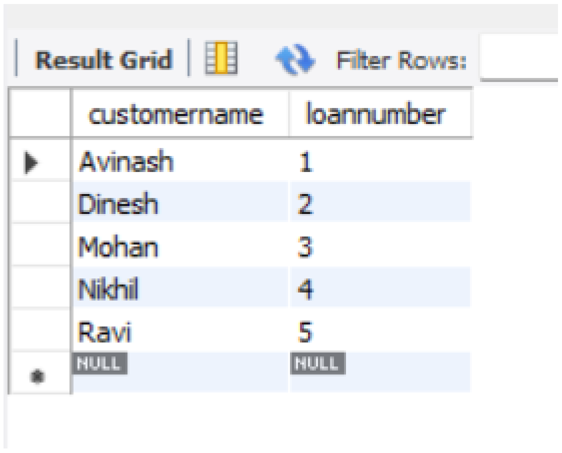
	customername	accno
▶	Avinash	1
	Dinesh	2
	Nikhil	4
	Ravi	5
	Avinash	8
	Nikhil	9
	Dinesh	10
	Nikhil	11
	Nikhil	12
*	NULL	NULL

```
insert into loan values(1, 'SBI_Chamrajpete',1000);
insert into loan values(2, 'SBI_Residency_road', 2000);
insert into loan values(3, 'SBI_Shivaji_road', 3000);
insert into loan values(4, 'SBI_Parliament_road', 4000);
insert into loan values(5, 'SBI_Jantarmantar', 5000);
select * from loan;
```



loannumber	branchname	amount
1	SBI_Chamrajpete	1000
2	SBI_Residency_road	2000
3	SBI_Shivaji_road	3000
4	SBI_Parliament_road	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

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

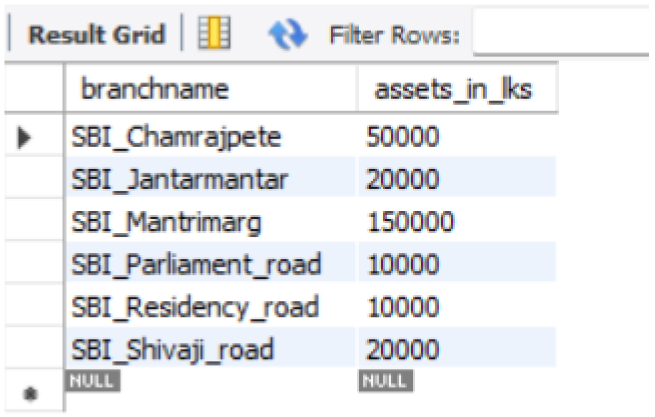


customername	loannumber
Avinash	1
Dinesh	2
Mohan	3
Nikhil	4
Ravi	5
NULL	NULL

Queries

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

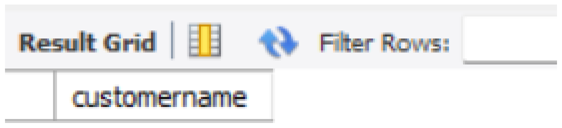
alter table Branch **rename column** assets **to** assets_in_lks;
select branchname, assets_in_lks **from** Branch;



branchname	assets_in_lks
SBI_Chamrajpete	50000
SBI_Jantarmantar	20000
SBI_Mantrimarg	150000
SBI_Parliament_road	10000
SBI_Residency_road	10000
SBI_Shivaji_road	20000
NULL	NULL

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

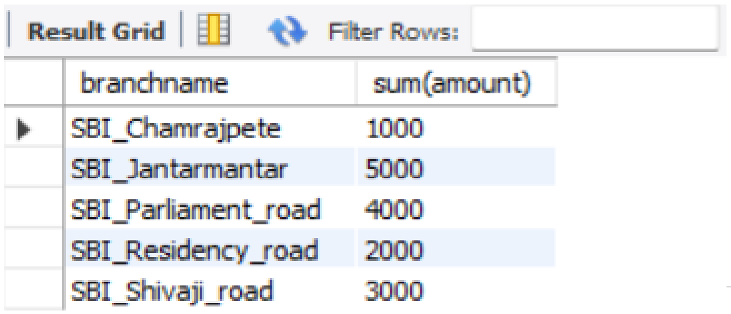
select d.customername **from** depositer d, Bankaccount b **where** b.branchname=**'ResideRoad' and** d.accno=b.accno **group by** d.customername **having count**(d.accno)>=2;



customername

● Create a view which gives each branch the sum of the amount of all the loans at the branch.
create view br as select branchname, **sum**(amount) from loan
group by branchname;

select * from br;



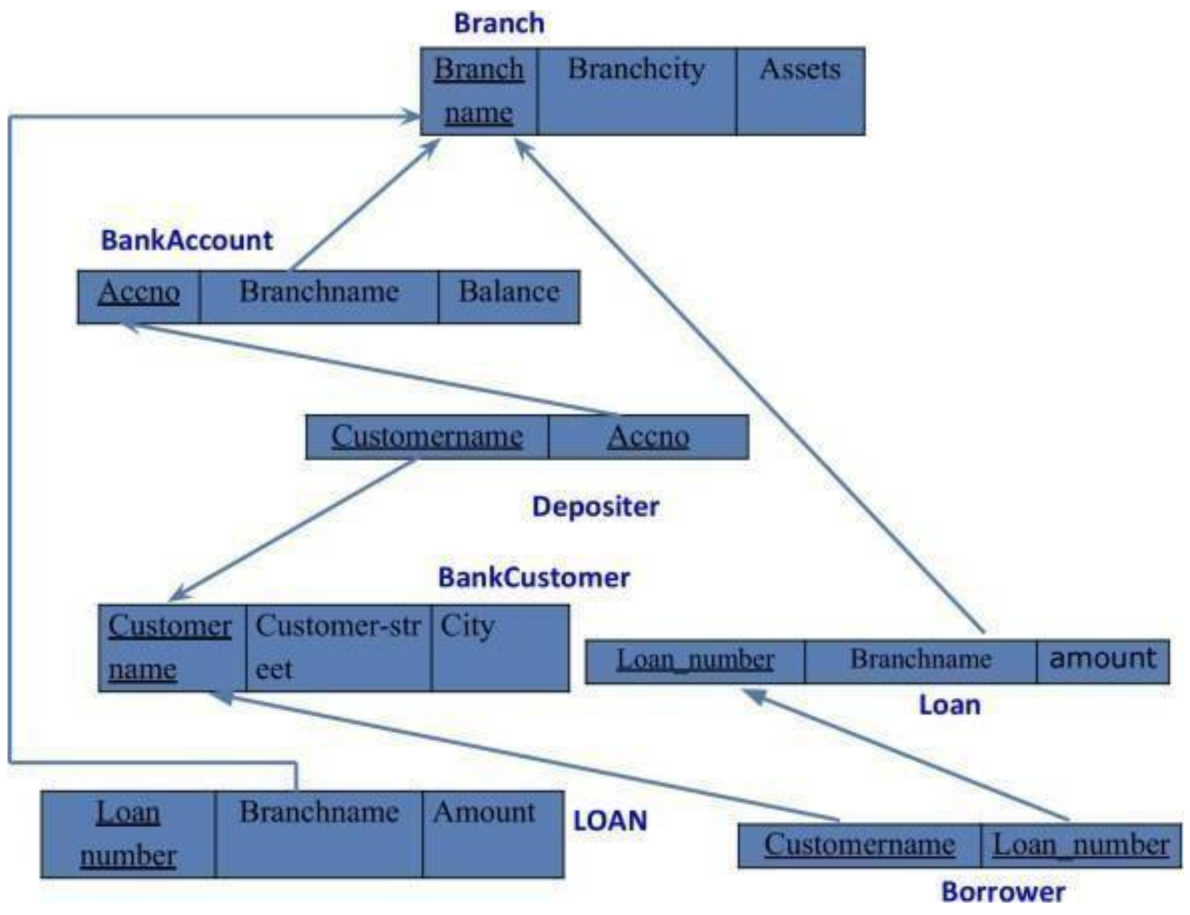
branchname	sum(amount)
SBI_Chamrajpete	1000
SBI_Jantarmantar	5000
SBI_Parliament_road	4000
SBI_Residency_road	2000
SBI_Shivaji_road	3000

More Queries on Bank Database

Questio n (Week 4)

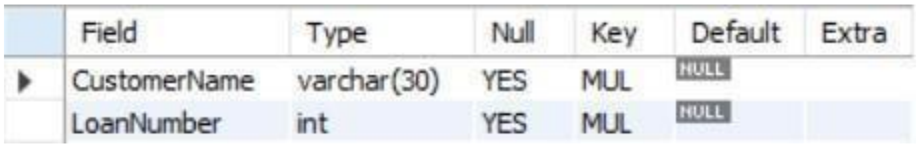
- Branch (branch-name: String, branch-city: String, assets: real)
- BankAccount(accno: int, branch-name: String, balance: real
- BankCustomer (customer-name: String, customer-street: String, customer-city: String)
- Depositer(customer-name: String, accno: int
- Loan (loan-number: int, branch-name: String, amount: real)
- Borrower(customer-name: String, loan-number: int)
- Find all the customers who have an account at all the branches - located in a specific city (Ex. Delhi).
- Find all customers who have a loan at the bank but do not have an account.
- Find all customers who have both an account and a loan at the Bangalore branch
- Find the names of all branches that have greater assets than all branches located in Bangalore
- Update the Balance of all accounts by 5%
- Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).

Schema Diagram:



Create Table:
create table Borrower(CustomerName varchar(30), LoanNumber int,
foreign key(CustomerName) references BankCustomer(CustomerName), foreign key(LoanNumber) references Loan(LoanNumber));

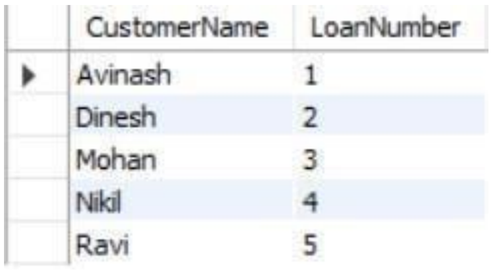
Structure of the Table:
desc Borrower;



	Field	Type	Null	Key	Default	Extra
▶	CustomerName	varchar(30)	YES	MUL	NULL	
	LoanNumber	int	YES	MUL	NULL	

Inserting Values to the tables:

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

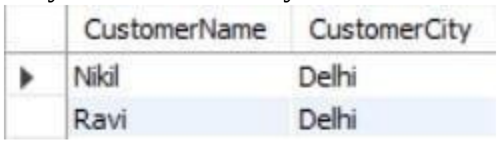


	CustomerName	LoanNumber
▶	Avinash	1
	Dinesh	2
	Mohan	3
	Nikil	4
	Ravi	5

Queries:

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

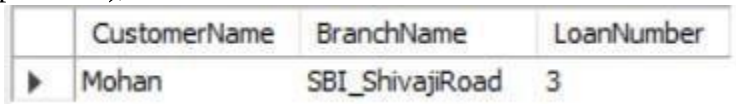
```
select distinct CustomerName, CustomerCity from Branch b, BankCustomer bc where
b.BranchCity=bc.CustomerCity and bc.CustomerCity="Delhi";
```



	CustomerName	CustomerCity
▶	Nikil	Delhi
	Ravi	Delhi

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

```
select distinct bc.CustomerName, l.BranchName, l.LoanNumber from BankCustomer bc, Loan l, Borrower b
where bc.CustomerName= b.CustomerName and l.LoanNumber=b.LoanNumber
and bc.CustomerName NOT IN ( select d.CustomerName
from Depositer d);
```

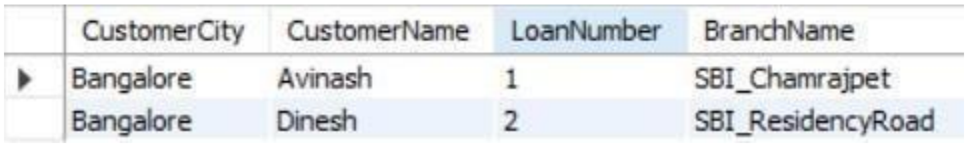


	CustomerName	BranchName	LoanNumber
▶	Mohan	SBI_ShivajiRoad	3

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

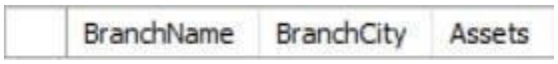
select distinct bc.CustomerCity, b.CustomerName, l.LoanNumber, br.BranchName fromBranch br, Borrower b, Loan l, BankCustomer bc
where
br.BranchCity = bc.CustomerCity AND br.BranchCity = 'Bangalore' AND l.LoanNumber = b.LoanNumbe
AND bc.CustomerName = b.CustomerName and br.BranchName=l.BranchName;

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



	CustomerCity	CustomerName	LoanNumber	BranchName
▶	Bangalore	Avinash	1	SBI_Chamrajpet
	Bangalore	Dinesh	2	SBI_ResidencyRoad

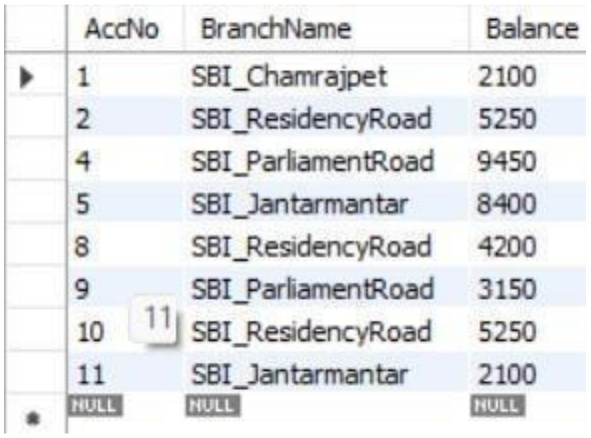
select distinct br.BranchName, br.BranchCity, br.Assets from Branch br where
br.Assets> all(select max(br.Assets) where br.BranchCity="Bangalore"); select max(Assets), BranchName from Branch
group by BranchName;



	BranchName	BranchCity	Assets

Update the Balance of all accounts by 5%
update BankAccount set Balance= Balance+ 0.05*Balance;

Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).
delete from BankAccount ba where ba.BranchName in(
select br.BranchName from Branch br where br.BranchCity="Bombay"); select * from BankAccount;



	AccNo	BranchName	Balance
▶	1	SBI_Chamrajpet	2100
	2	SBI_ResidencyRoad	5250
	4	SBI_ParliamentRoad	9450
	5	SBI_Jantarmantar	8400
	8	SBI_ResidencyRoad	4200
	9	SBI_ParliamentRoad	3150
	10	SBI_ResidencyRoad	5250
	11	SBI_Jantarmantar	2100
*	NULL	NULL	NULL

Employee Database

Question

(Week 5)

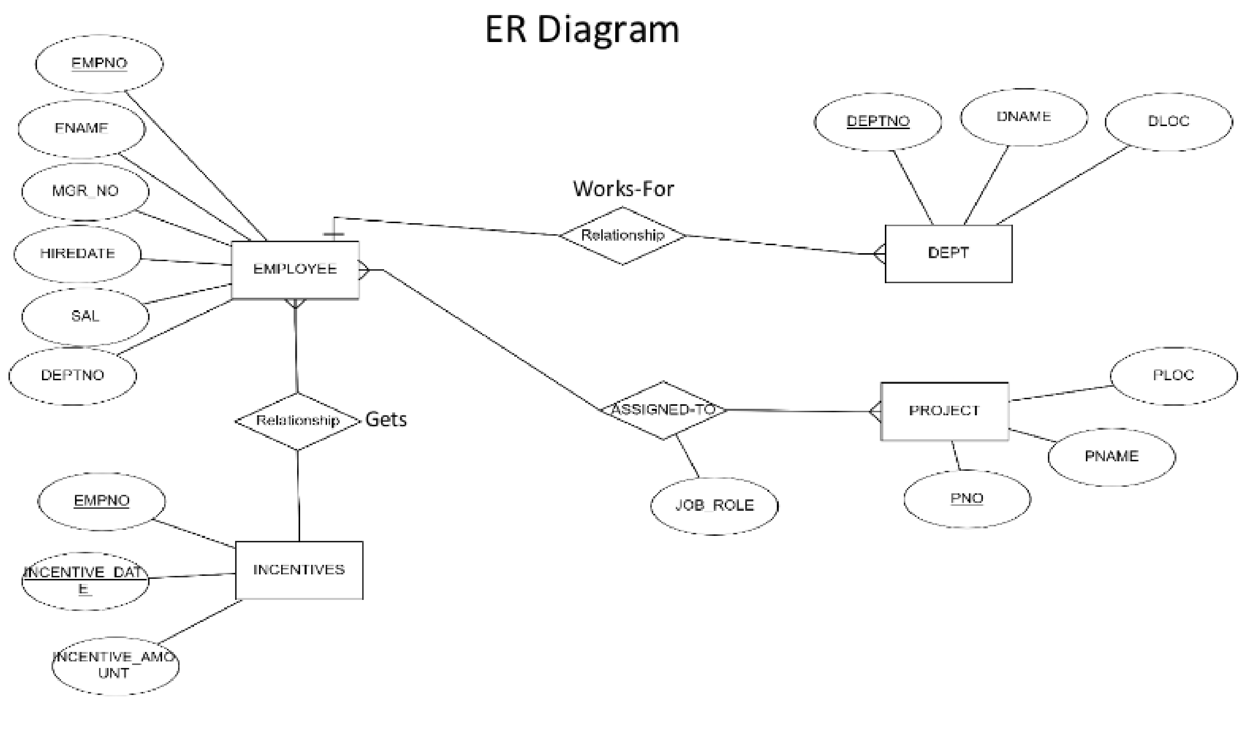
1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.

2. Enter greater than five tuples for each table.

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

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

5. Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.



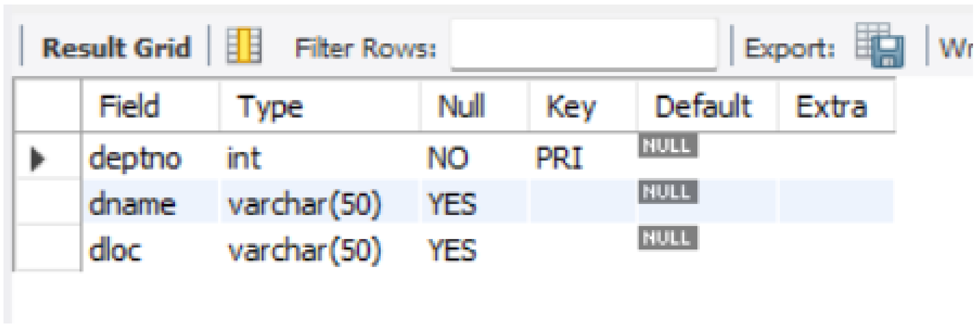
Create database

```
create database employee_421;
 use employee_421;
```

Create tables

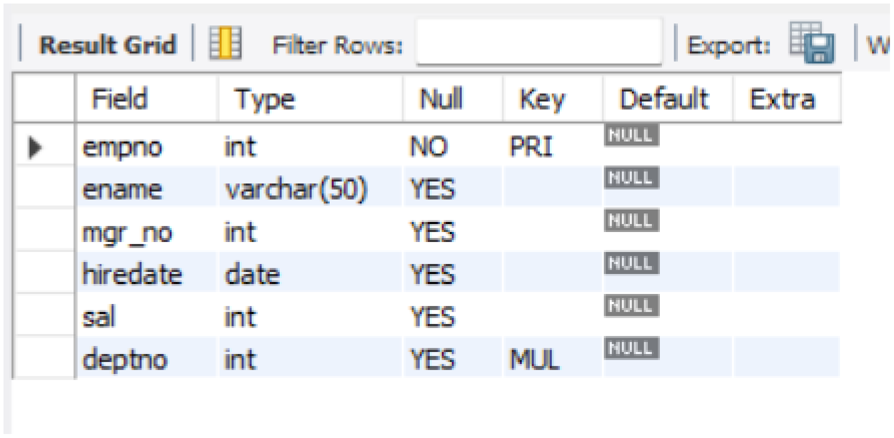
```
create table dept (
 deptno int primary key,
 dname varchar(50),
 dloc varchar(50)
  );

  create table employee (
  empno int primary key,
  ename varchar(50),
  mgr_no int,
  hiredate date,
  sal int,
  deptno int,
  foreign key (deptno) references dept(deptno)
  );

  create table project (
  pno int primary key,
  ploc varchar(50),
  pname varchar(50)
  );

  create table assigned_to (
  empno int,
  pno int,
  job_role varchar(50),
  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 int,
  foreign key (empno) references employee(empno));
```

Department table :



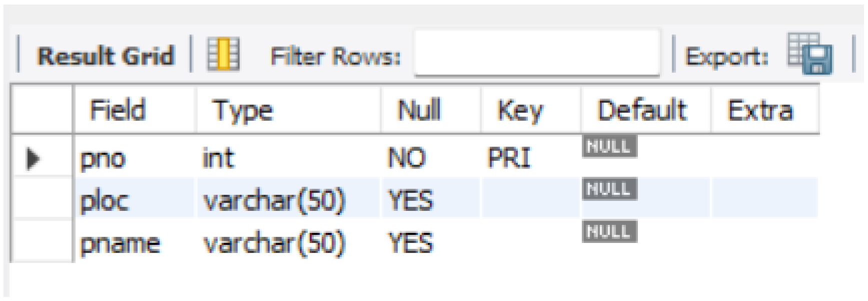
	Field	Type	Null	Key	Default	Extra
▶	deptno	int	NO	PRI	NULL	
	dname	varchar(50)	YES		NULL	
	dloc	varchar(50)	YES		NULL	

Employee Table :



	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	ename	varchar(50)	YES		NULL	
	mgr_no	int	YES		NULL	
	hiredate	date	YES		NULL	
	sal	int	YES		NULL	
	deptno	int	YES	MUL	NULL	

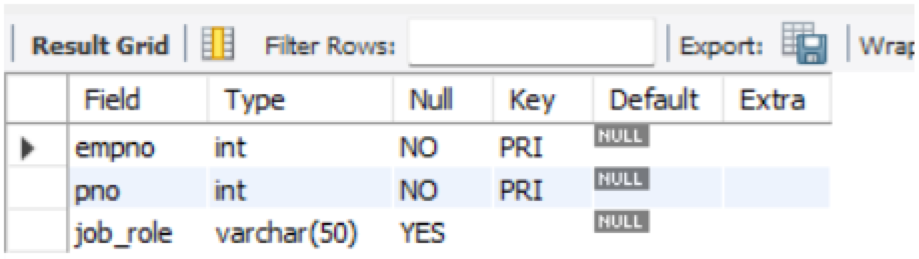
Project table :



Result Grid | Filter Rows: | Export:

Field	Type	Null	Key	Default	Extra
pno	int	NO	PRI	NULL	
ploc	varchar(50)	YES		NULL	
pname	varchar(50)	YES		NULL	

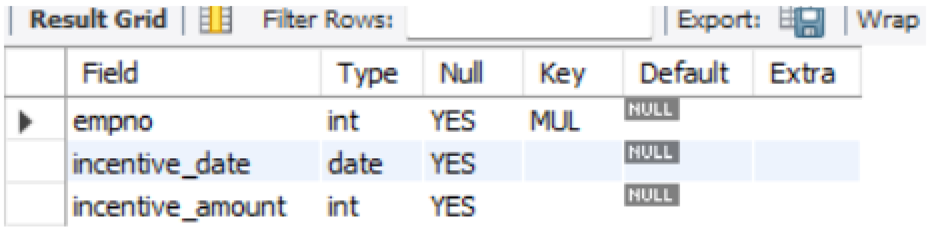
Assigned_to table :



Result Grid | Filter Rows: | Export: | Wrap

Field	Type	Null	Key	Default	Extra
empno	int	NO	PRI	NULL	
pno	int	NO	PRI	NULL	
job_role	varchar(50)	YES		NULL	

Incentive table:

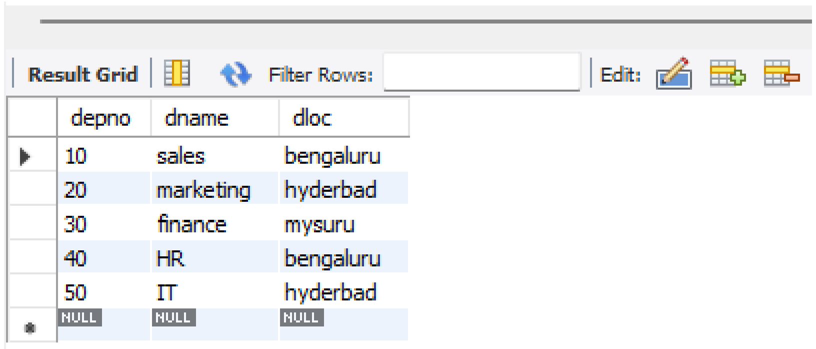


Result Grid | Filter Rows: | Export: | Wrap

Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
incentive_date	date	YES		NULL	
incentive_amount	int	YES		NULL	

Inserting the values to the tables

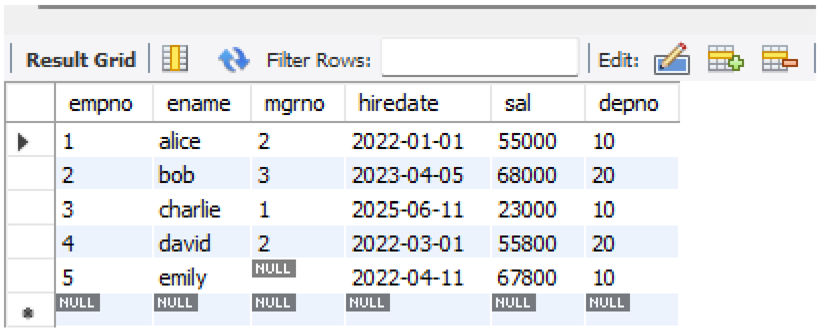
```
insert into dept values(10,'sales','bengaluru');
insert into dept values(20,'marketing','hyderbad');
insert into dept values(30,'finance','mysuru');
insert into dept values(40,'HR','bengaluru');
insert into dept values(50,'IT','hyderbad');
select * from dept;
```



Result Grid | Filter Rows: | Edit:

	depno	dname	dloc
▶	10	sales	bengaluru
	20	marketing	hyderbad
	30	finance	mysuru
	40	HR	bengaluru
	50	IT	hyderbad
●	NULL	NULL	NULL

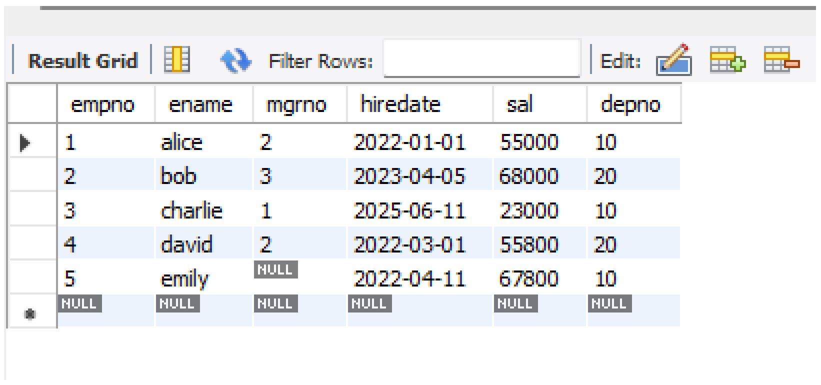
insert into employee values(1,'alice',2,'2022-01-01',55000,10);
insert into employee values(2,'bob',3,'2023-04-05',68000,20);
insert into employee values(3,'charlie',1,'2025-06-11',23000,10);
insert into employee values(4,'david',2,'2022-03-01',55800,20);
insert into employee values(5,'emily',null,'2022-04-11',67800,10);
select * from employee;



Result Grid | Filter Rows: | Edit:

	empno	ename	mgrno	hiredate	sal	depno
▶	1	alice	2	2022-01-01	55000	10
	2	bob	3	2023-04-05	68000	20
	3	charlie	1	2025-06-11	23000	10
	4	david	2	2022-03-01	55800	20
	5	emily	NULL	2022-04-11	67800	10
●	NULL	NULL	NULL	NULL	NULL	NULL

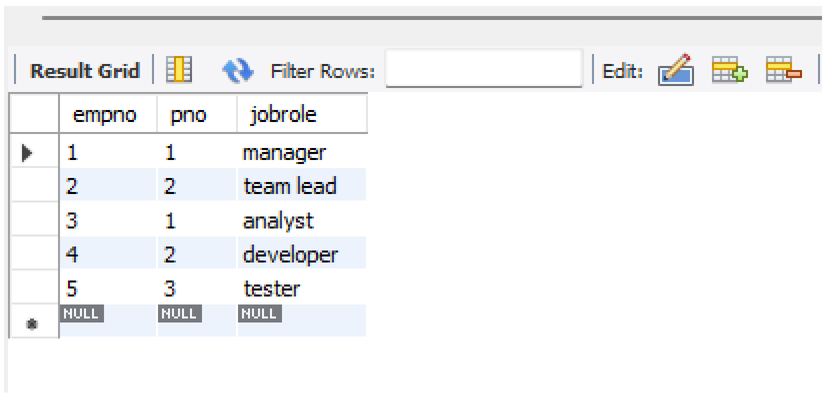
insert into project values(1,'e-learning','bengaluru');
insert into project values(2,'hostel management','hyderbad');
insert into project values(3,'hotel management','bengaluru');
insert into project values(4,'face recognition','chennai');
insert into project values(5,'face emotion recognition','mysuru');
select * from project;



Result Grid | Filter Rows: | Edit:

empno	ename	mgrno	hiredate	sal	depno
1	alice	2	2022-01-01	55000	10
2	bob	3	2023-04-05	68000	20
3	charlie	1	2025-06-11	23000	10
4	david	2	2022-03-01	55800	20
5	emily	NULL	2022-04-11	67800	10
NULL	NULL	NULL	NULL	NULL	NULL

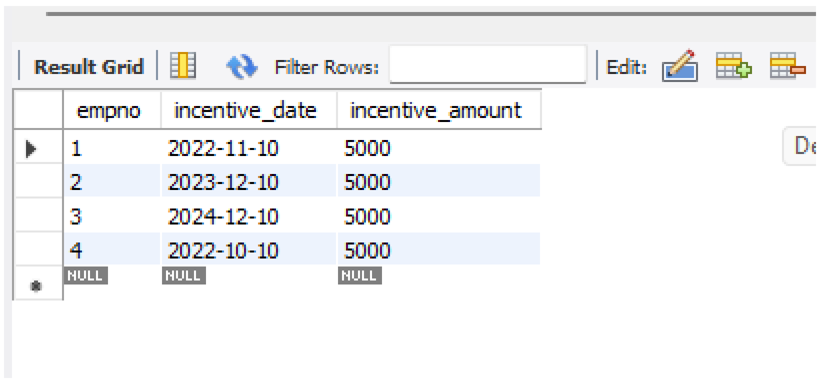
insert into assignment values(1,1,'manager');
insert into assignment values(2,2,'team lead');
insert into assignment values(3,1,'analyst');
insert into assignment values(4,2,'developer');
insert into assignment values(5,3,'tester');
select * from assignment;



Result Grid | Filter Rows: | Edit:

empno	pno	jobrole
1	1	manager
2	2	team lead
3	1	analyst
4	2	developer
5	3	tester
NULL	NULL	NULL

insert into incentives values(1,'2022-11-10',5000);
insert into incentives values(2,'2023-12-10',5000);
insert into incentives values(3,'2024-12-10',5000);
insert into incentives values(4,'2022-10-10',5000);
select * from incentives;



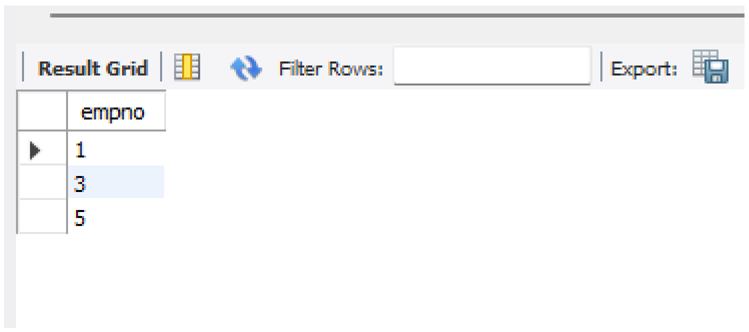
Result Grid | Filter Rows: | Edit:

	empno	incentive_date	incentive_amount
▶	1	2022-11-10	5000
	2	2023-12-10	5000
	3	2024-12-10	5000
	4	2022-10-10	5000
✱	NULL	NULL	NULL

Queries

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

select empno **from** assignment **where** pno in(select pno **from** project **where** ploc in('bengaluru','mysuru'));

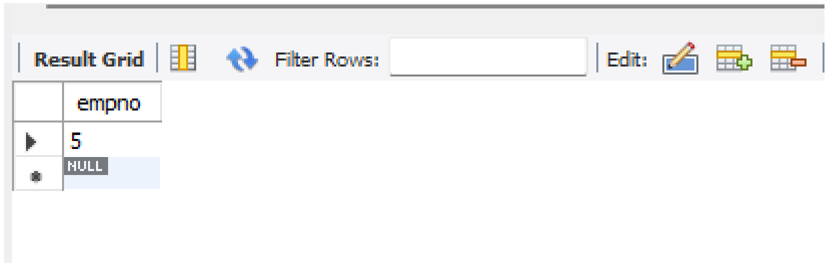


Result Grid | Filter Rows: | Export:

	empno
▶	1
	3
	5

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

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



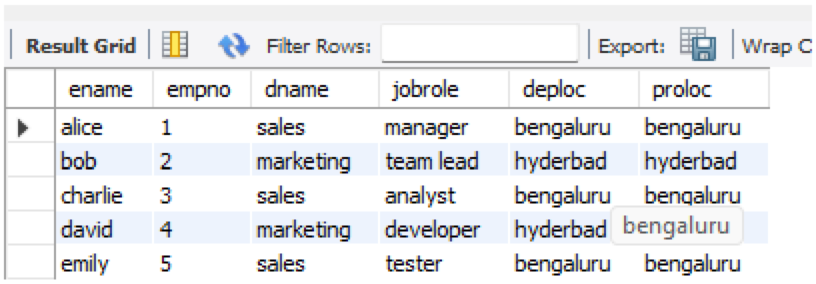
Result Grid | Filter Rows: | Edit:

	empno
▶	5
✱	NULL

● Write a SQL query to find the employees name, number, dept, job_role, departmen

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.jobrole, d.dloc as deploc, p.ploc as proloc **from** employee e **join** dept d on e.depno = d.depno join assignment a **on** e.empno = a.empno **joi** project p on a.pno=p.pno **where** d.dloc = p.ploc;



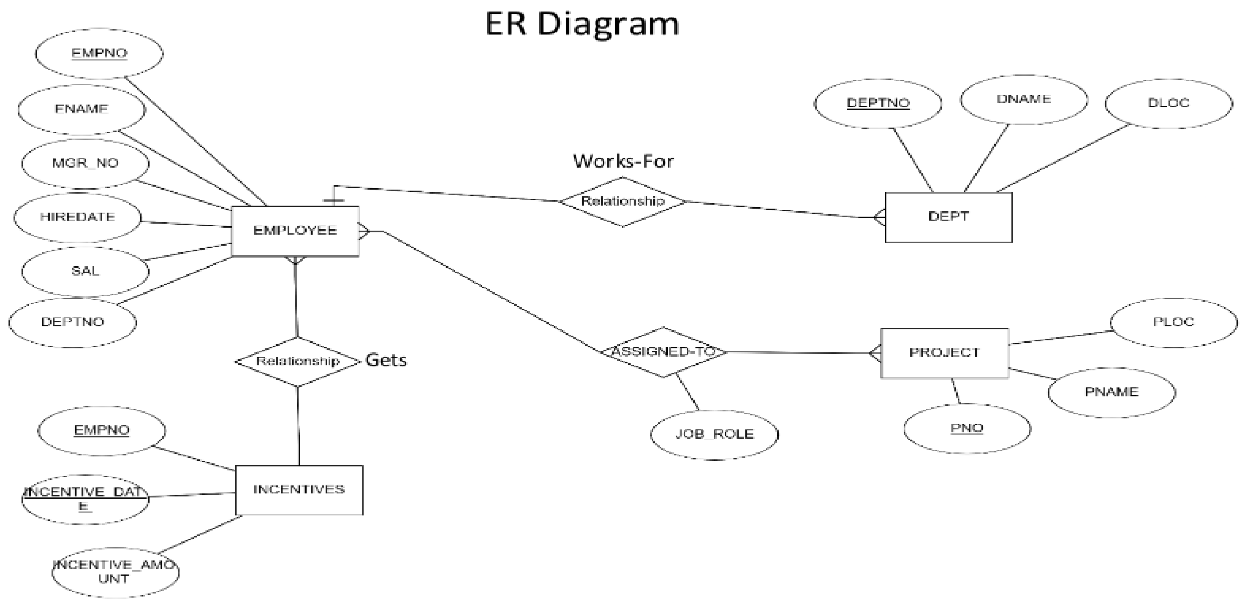
ename	empno	dname	jobrole	deploc	proloc
alice	1	sales	manager	bengaluru	bengaluru
bob	2	marketing	team lead	hyderbad	hyderbad
charlie	3	sales	analyst	bengaluru	bengaluru
david	4	marketing	developer	hyderbad	bengaluru
emily	5	sales	tester	bengaluru	bengaluru

More Queries on Insurance Database

Question

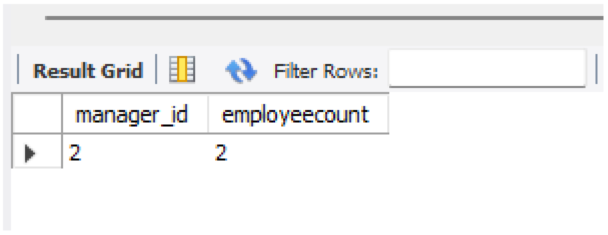
(Week 6)

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2. Enter greater than five tuples for each table.
3. List the name of the managers with the maximum employees
4. Display those managers name whose salary is more than average salary of his employee.
5. Find the name of the second top level managers of each department.
6. Find the employee details who got second maximum incentive in January 2019.
7. Display those employees who are working in the same department where his manager is working.



● **List the name of the managers with the maximum employees**

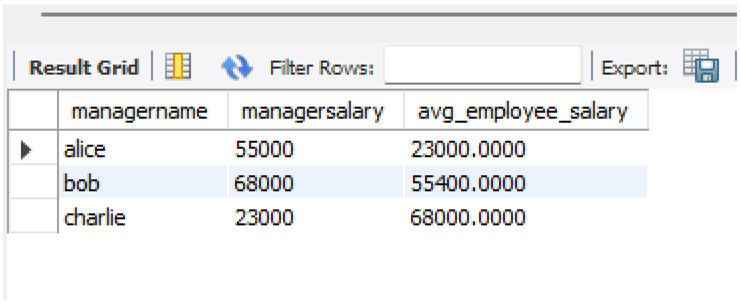
select mgrno as manager_id, **count(**empno) as employeecount **from** employee **group by** mgrno **order by** employeecount **desc** limit 1;



```
manager_id  employeecount
2           2
```

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

select m.ename as managername,m.sal **as** managersalary,emp_avg.avg_employee_salary **from** employee m **join** (select mgrno,avg(sal)as avg_employee_salary from employee **group by** mgrno) as emp_avg **on** m.empno=emp_avg.mgrno;

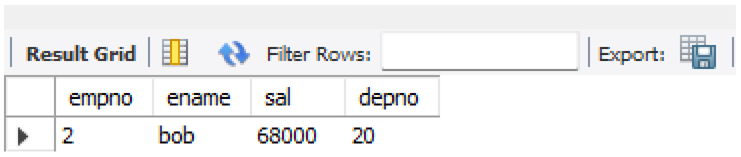


```
managername  managersalary  avg_employee_salary
alice        55000          23000.0000
bob          68000          55400.0000
charlie      23000          68000.0000
```

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

select ename as secondtopmanager **from**(select m.empno,d.depno,row_number() over(partition by d.depno **order by** m.sal desc) as rank1 **from** employee m **join** dept d on m.depno=d.depno **where** m.mgrno is null) as rankedmanagers **where** rank1=2;

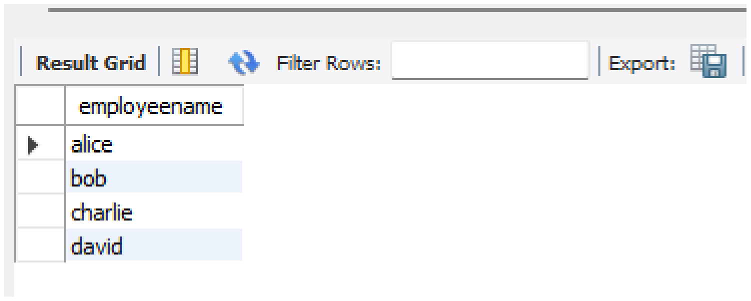
● **Find the employee details who got second maximum incentive in January 2019** select e.empno,e.ename,e.sal, e.depno from employee e join incentives i on e.empno=i.empno where i.incentive_date between '2022-11-10' and '2024-12-10' order by i.incentive_amount desc limit 1 offset 1;


Result Grid | Filter Rows: | Export:

empno	ename	sal	depno
2	bob	68000	20

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

select e.ename as employeename **from** employee e **join** employee m **on** e.mgrno=m.mgrno **where** e.depno=m.depno;


Result Grid | Filter Rows: | Export:

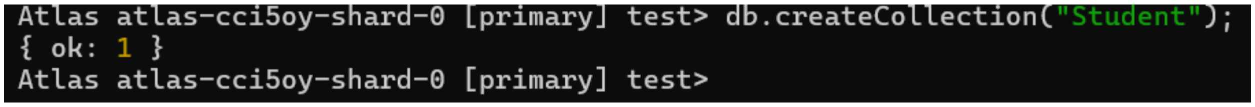
employeename
alice
bob
charlie
david

No SQL Student Database

Questio n (Week 8)

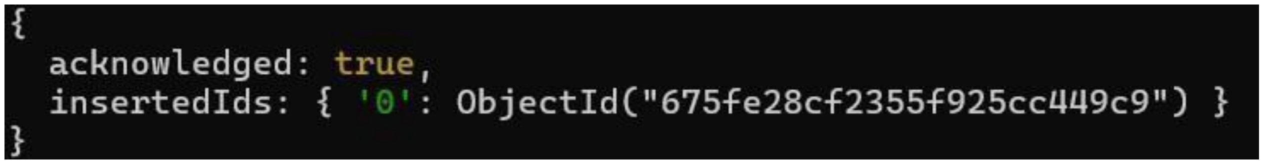
- Perform the following DB operations using MongoDB.
- Create a database "Student" with the following attributes Rollno, Age, ContactNo, Email-Id.
- Insert appropriate values
- Write query to update Email-Id of a student with rollno 10.
- Replace the student name from "ABC" to "FEM" of rollno 11.

Create Database:
db.createCollection("Student");



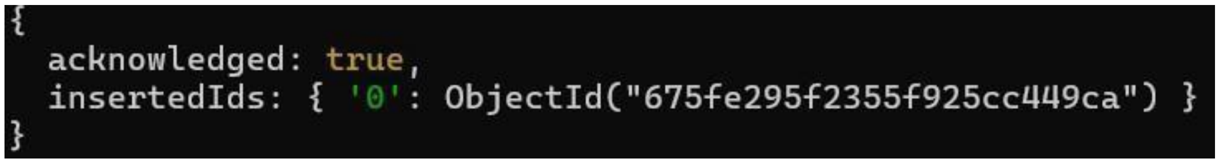
```
Atlas atlas-cci5oy-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
Atlas atlas-cci5oy-shard-0 [primary] test>
```

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



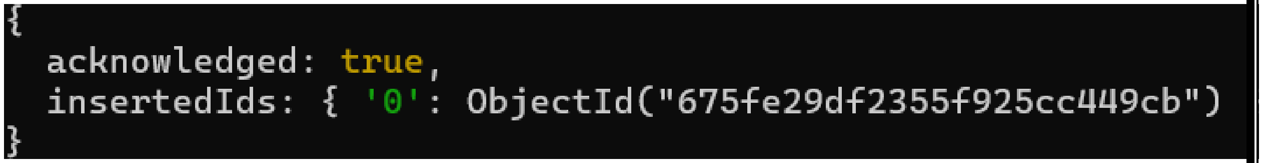
```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe28cf2355f925cc449c9") }
}
```

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



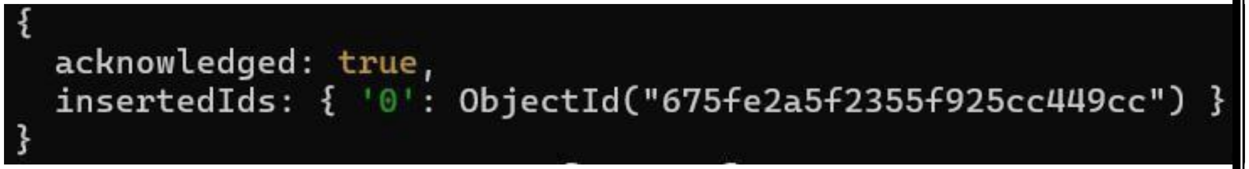
```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe295f2355f925cc449ca") }
}
```

db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});



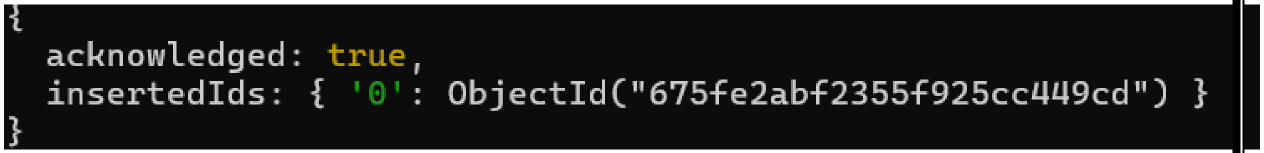
```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe29df2355f925cc449cb")
}
```

db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});



```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe2a5f2355f925cc449cc") }
}
```

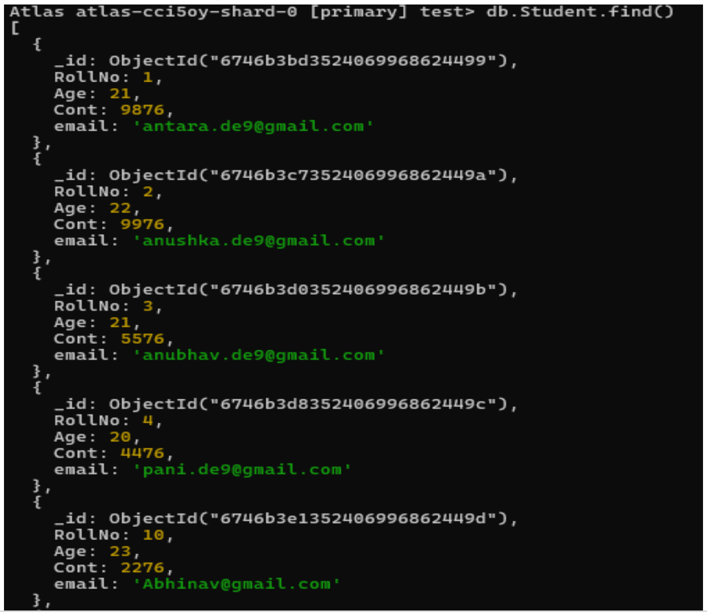
db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});



```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe2abf2355f925cc449cd") }
}
```

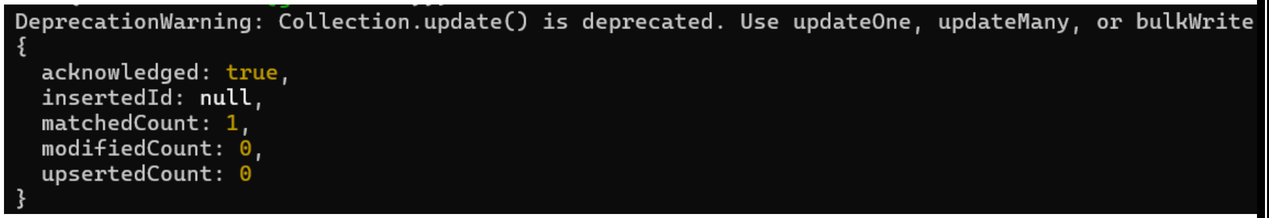
Queries:

db.Student.find()



```
Atlas atlas-cci5oy-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("6746b3bd352406996862449 9"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3c7352406996862449a"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3d0352406996862449b"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3d8352406996862449c"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3e1352406996862449d"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  },
```

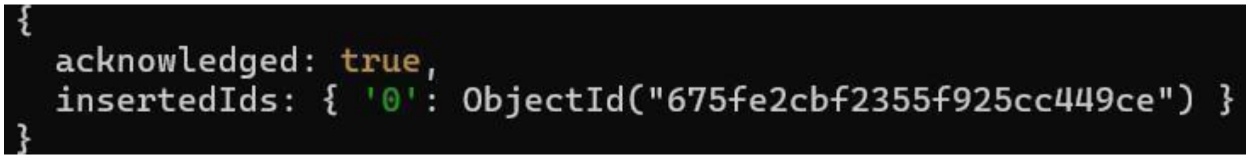
Write query to update Email-Id of a student with rollno 10.
db.Student.update({RollNo:10},{$set:{email:"Abhinav@gmail.com"}})
4e



```
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

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

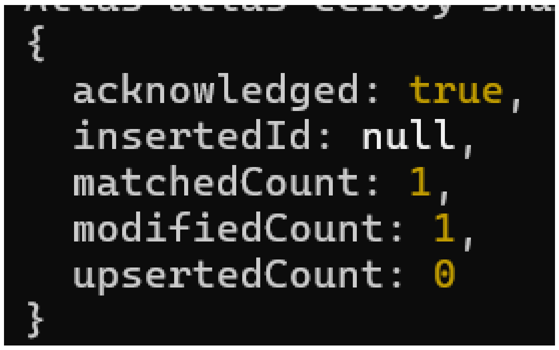
db.Student.insert({RollNo:11,Age:22,Name:"ABC",Cont:2276,email:"rea.de9@gmail. c



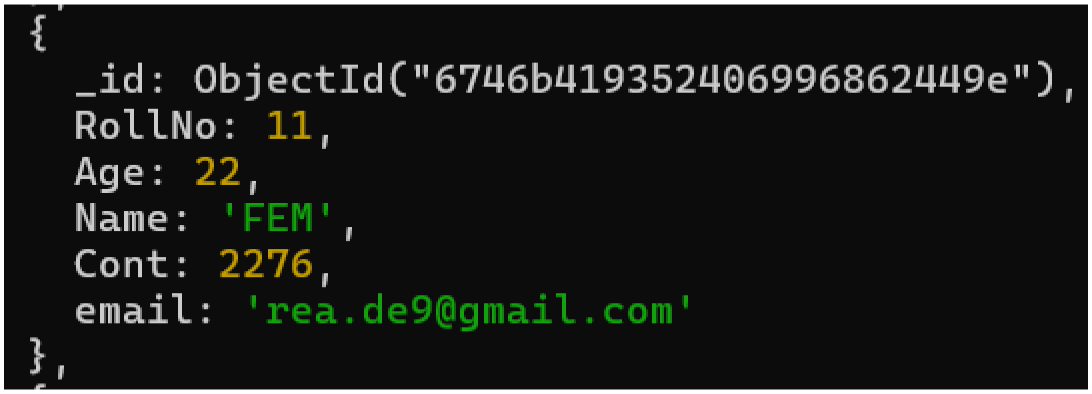
```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe2cbf2355f925cc449ce") }
}
```

om"});

db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})



```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```



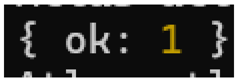
```
{
  _id: ObjectId("6746b41935240699686 2449e"),
  RollNo: 11,
  Age: 22,
  Name: 'FEM',
  Cont: 2276,
  email: 'rea.de9@gmail.com'
},
```

No SQL Customers Database

Questio n (Week 9)

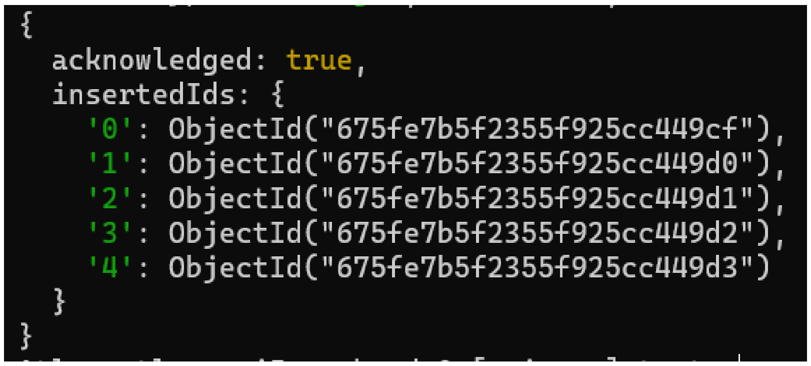
- Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type
- Insert at least 5 values into the table
- Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.
- Determine Minimum and Maximum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

Create Database:
db.createCollection("Customer");



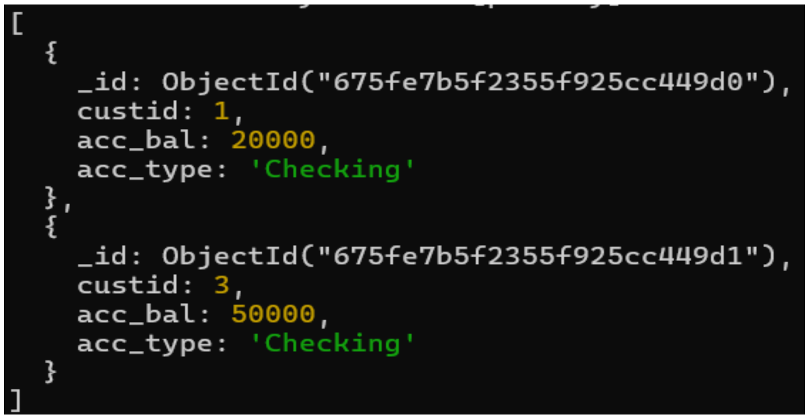
```
{ ok: 1 }
```

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



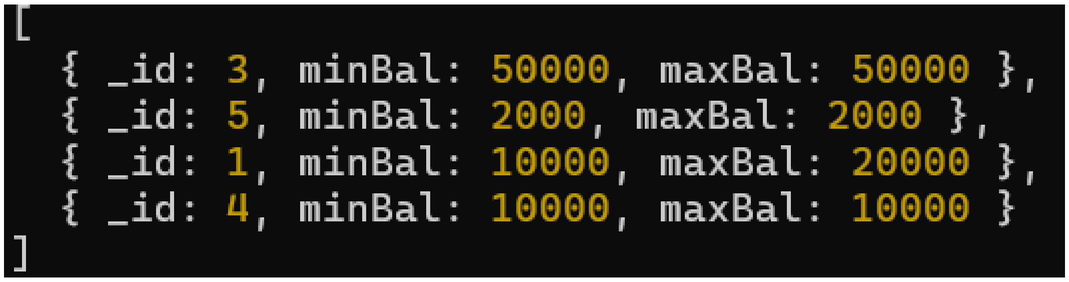
```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("675fe7b5f2355f925cc449cf"),
    '1': ObjectId("675fe7b5f2355f925cc449d0"),
    '2': ObjectId("675fe7b5f2355f925cc449d1"),
    '3': ObjectId("675fe7b5f2355f925cc449d2"),
    '4': ObjectId("675fe7b5f2355f925cc449d3")
  }
}
```

Queries:
Write a query to display those records whose total account balance is greater than 1200 of account type ‘Z’ for each customer_id.
db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});



```
[
  {
    _id: ObjectId("675fe7b5f2355f925cc449d0"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("675fe7b5f2355f925cc449d1"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  }
]
```

Determine Minimum and Maximum account balance for each customer_id.
db.Customer.aggregate([{$group:{_id:"$custid", minBal:{$min:"$acc_bal"}, maxBal:{$max:"$acc_bal"}}}]):

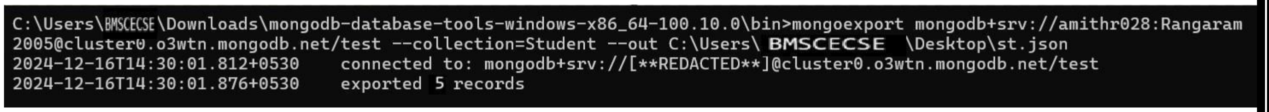


```
[
  { _id: 3, minBal: 50000, maxBal: 50000 },
  { _id: 5, minBal: 2000, maxBal: 2000 },
  { _id: 1, minBal: 10000, maxBal: 20000 },
  { _id: 4, minBal: 10000, maxBal: 10000 }
]
```

db.Customers.drop()

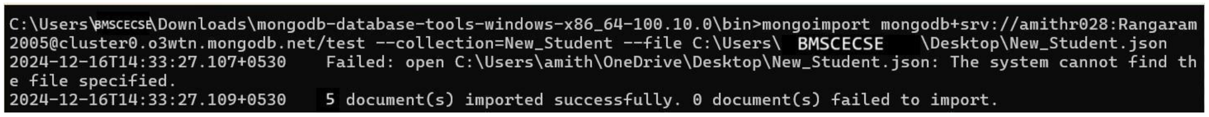
```
true
```

mongoexport mongodb+srv://dbms:@cluster0.xmdk9.mongodb.net/test
--collection=Student --out C:\Users\BMSCECSE\Desktop\st.json



```
C:\Users\BMSCECSE\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\bin>mongoexport mongodb+srv://amithr028:Rangaram
2005@cluster0.o3wtn.mongodb.net/test --collection=Student --out C:\Users\ BMSCECSE  \Desktop\st.json
2024-12-16T14:30:01.812+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.o3wtn.mongodb.net/test
2024-12-16T14:30:01.876+0530    exported 5 records
```

mongoimport mongodb+srv://dbms:@cluster0.xmdk9.mongodb.net/test
--collection=New_Student --file C:\Users\BMSCECSE\Desktop\New_Student.json



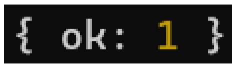
```
C:\Users\BMSCECSE\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\bin>mongoimport mongodb+srv://amithr028:Rangaram
2005@cluster0.o3wtn.mongodb.net/test --collection=New_Student --file C:\Users\ BMSCECSE    \Desktop\New_Student.json
2024-12-16T14:33:27.107+0530    Failed: open C:\Users\amith\OneDrive\Desktop\New_Student.json: The system cannot find th
e file specified.
2024-12-16T14:33:27.109+0530    5 document(s) imported successfully. 0 document(s) failed to import.
```

No SQL Restaurants Database

Question (Week 10)

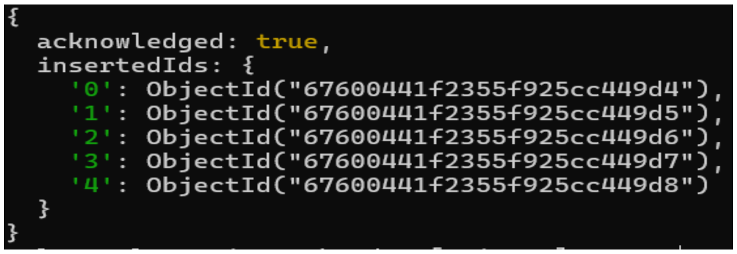
- Write a MongoDB query to display all the documents in the collection restaurants.
- Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.
- Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10.
- Write a MongoDB query to find the average score for each restaurant.
- Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

Create Database:
db.createCollection("restaurants");



```
{ ok: 1 }
```

Inserting Values to the tables:
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"}}])

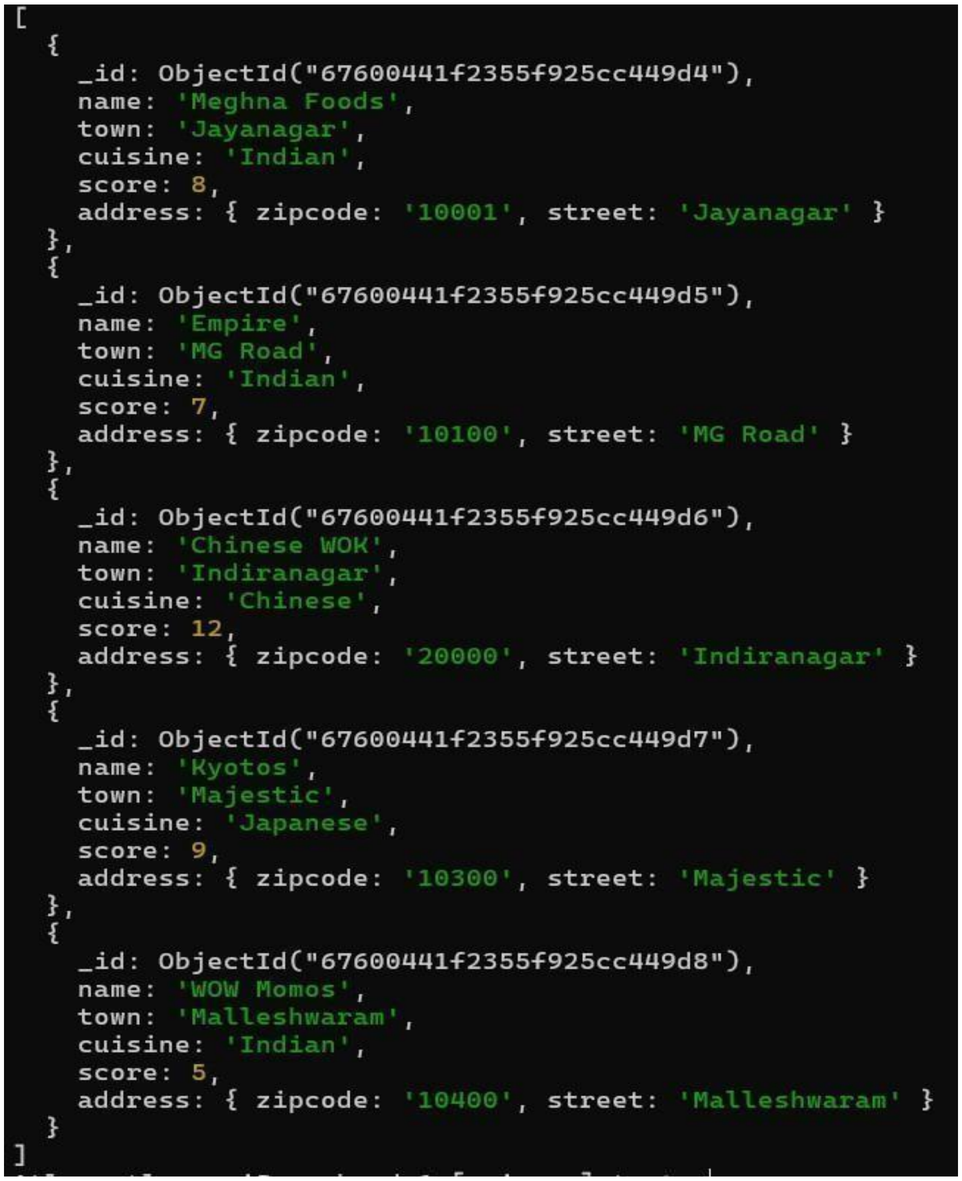


```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("67600441f2355f925cc449d4"),
    '1': ObjectId("67600441f2355f925cc449d5"),
    '2': ObjectId("67600441f2355f925cc449d6"),
    '3': ObjectId("67600441f2355f925cc449d7"),
    '4': ObjectId("67600441f2355f925cc449d8")
  }
}
```

Queries:

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

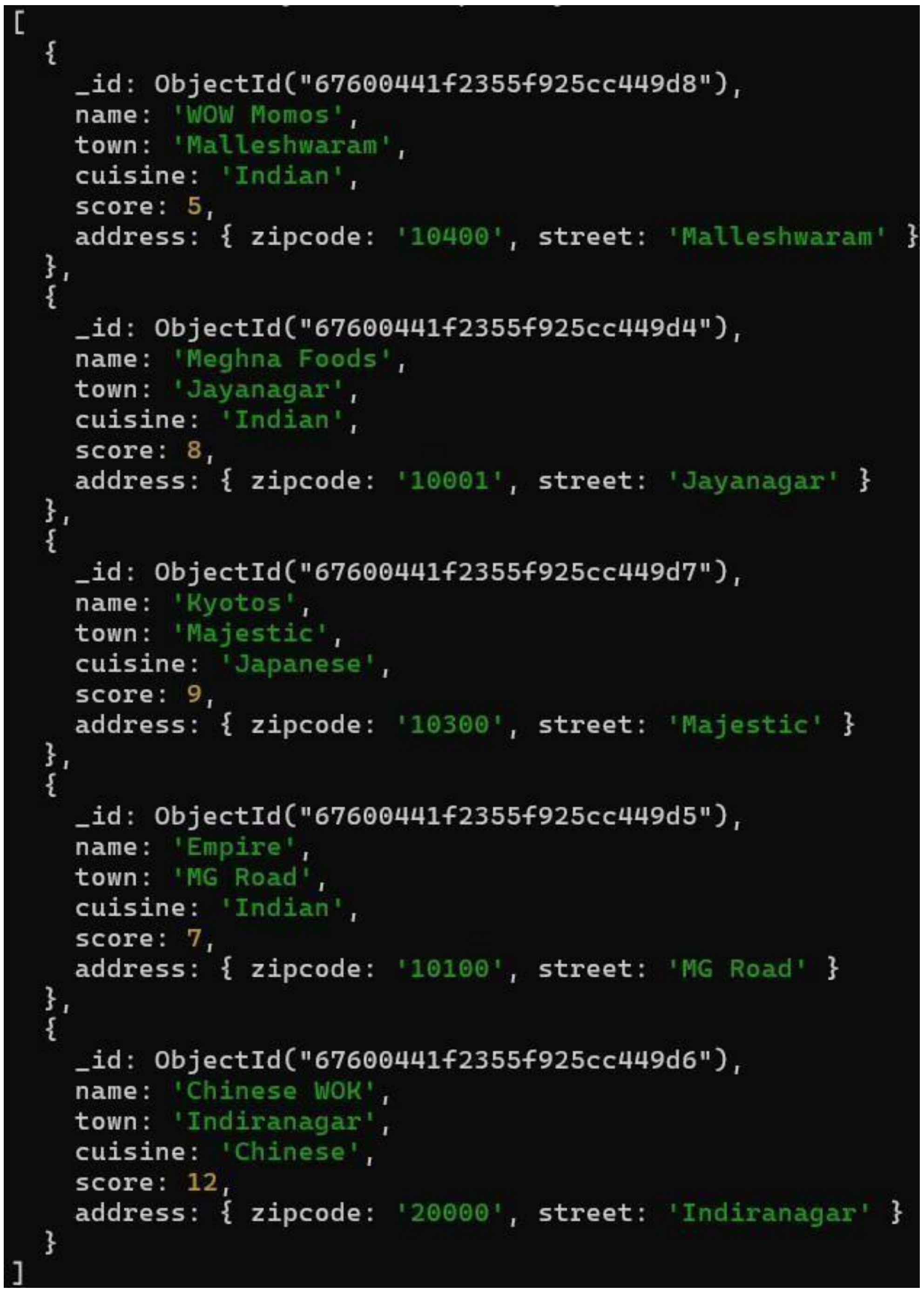
db.restaurants.find({})



```
[
  {
    _id: ObjectId("67600441f2355f925cc449d4"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d5"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d6"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d7"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d8"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

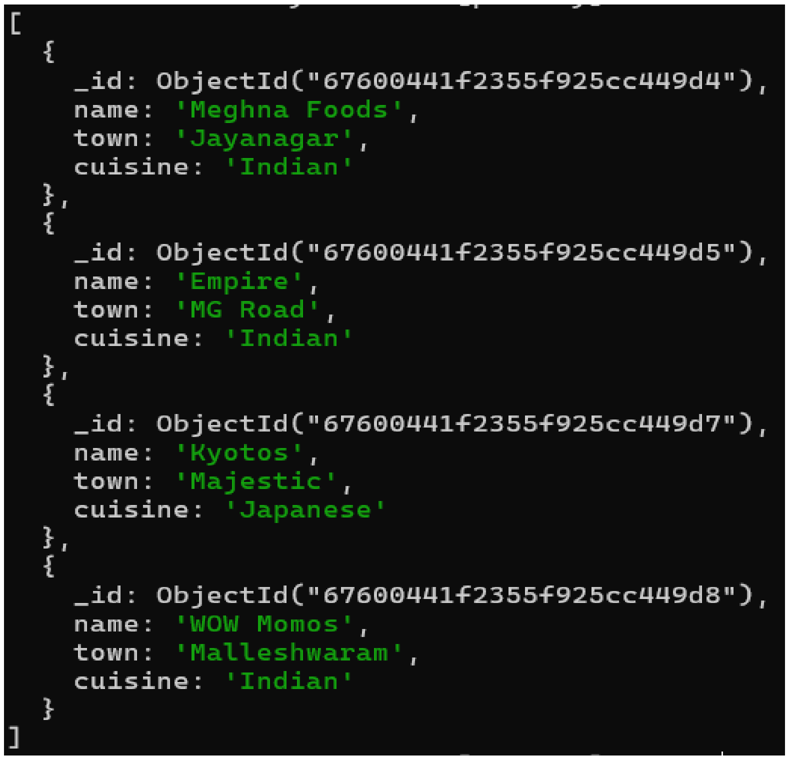
Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns

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



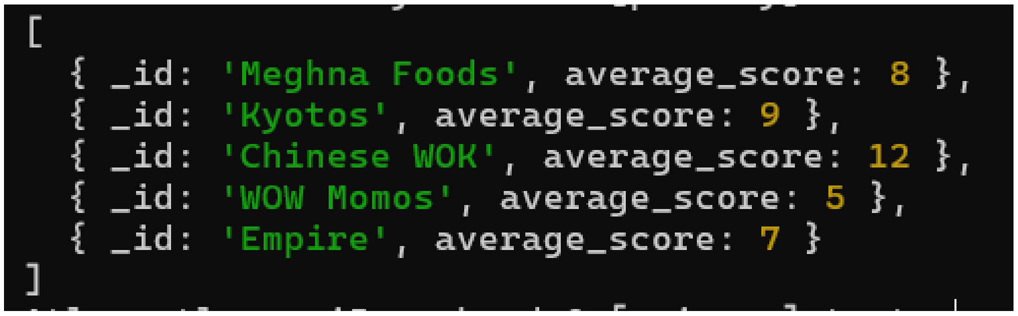
```
[
  {
    _id: ObjectId("67600441f2355f925cc449d8"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d4"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d7"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d5"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d6"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

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("67600441f2355f925cc449d4"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67600441f2355f925cc449d5"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67600441f2355f925cc449d7"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId("67600441f2355f925cc449d8"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
]
```

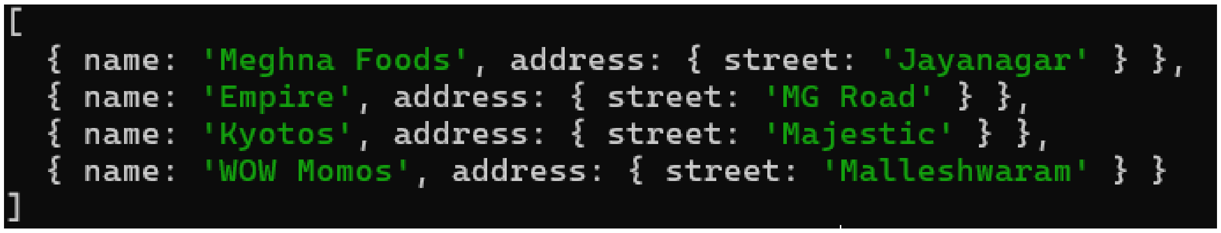
Write a MongoDB query to find the average score for each restaurant.
db.restaurants.aggregate([{ $group: { _id: "$name", average_score: { $avg:"$score"}}}])



```
[
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Empire', average_score: 7 }
]
```

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' } }
]
```