

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT on

Database Management Systems (23CS3PCDBM)

Submitted by

CHEETHAN K S (1BM23CS074)

in partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING



B.M.S. COLLEGE OF ENGINEERING
BENGALURU-560019 Sep-2024 to Jan-2025

(Autonomous Institution under VTU)

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 **CHEZHAN K S(1BM23CS074)**, 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 2024-25. 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.

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

Index

Sl. No.	Date	Experiment Title	Page No.
1	4-10-2024	Insurance Database	4
2	18-10-2024	More Queries on Insurance Database	13
3	18-10-2024	Bank Database	16
4	25-10-2024	More Queries on Bank Database	22
5	8-10-2024	Employee Database	25
6	15-11-2024	More Queries on Employee Database	30
7	27-12-2024	Supplier Database	31
8	27-12-2024	NO SQL - Student Database	35
9	27-12-2024	NO SQL - Customer Database	38
10	27-12-2024	NO SQL – Restaurant Database	40

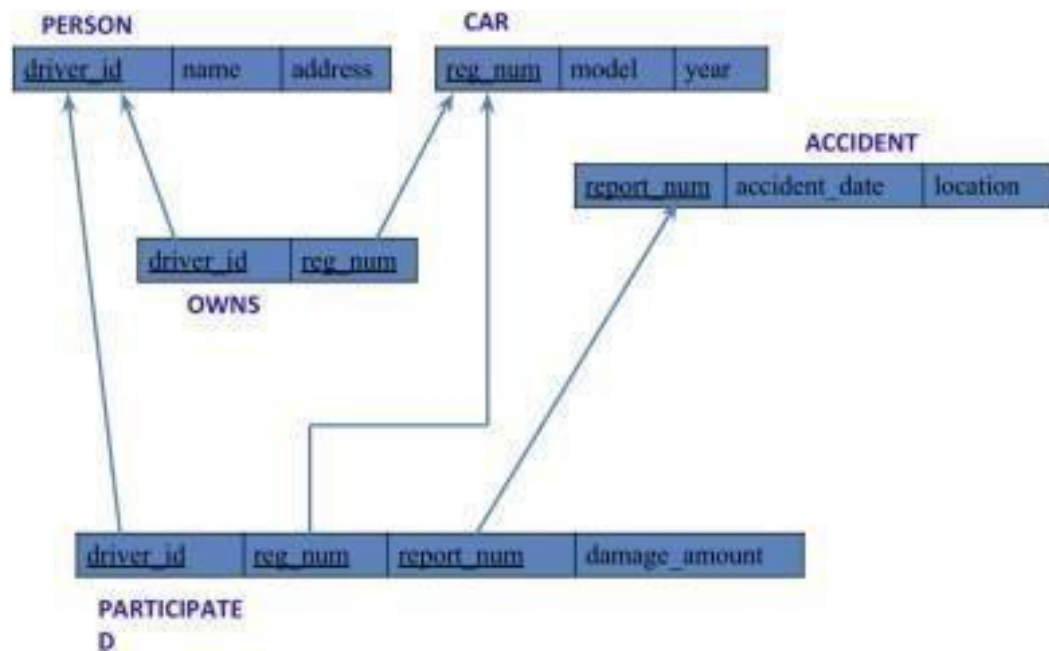
Insurance Database

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
- Find maximum damage amount
- Display driver id who did accident with damage amount greater than or average amount

Schema Diagram



Create database

```
create database insure;
```

```
use insure;
```

Create table

```
create table
```

```
insurance_074( driver_id
```

```
varchar(20), name
```

```
varchar(30), address
```

```
varchar(50),
```

```
PRIMARY KEY(driver_id)
```

```
);
```

```
create table insurance_074.car(
```

```
reg_num varchar(15),
```

```
model varchar(10),
```

```
year int,
```

```
PRIMARY KEY(reg_num)
```

```
);
```

```
create table insurance_074.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_074.accident(
report_num int,
accident_date date,
location varchar(50),
PRIMARY KEY(report_num)
);

Create table insurance_074.participated(
driver_id varchar(20),
reg_num varchar(10),
report_num int, damage_amount
int,
PRIMARY KEY(driver_id,reg_num,report_num),
FOREIGN KEY(driver_id) REFERENCES person(driver_id),
FOREIGN KEY(reg_num) REFERENCES car(reg_num),
FOREIGN KEY(report_num) REFERENCES accident(report_num)
);

```

Structure of the table

desc person;

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

desc accident;

Result Grid

Filter Rows:

Export:

Wrap Cell Contents:

	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 Contents:

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

desc owns;

Result Grid

Filter Rows:

Exports:

Wrap Cell Contents:

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

Inserting Values to the table

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

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

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

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

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

```
select * from person;
```

Result Grid			
Filter Rows:			
	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

```
insert into car values("KA052250","Indica", "1990");
```

```
insert into car values("KA031181","Lancer", "1957");
```

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

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

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

```
select * from car;
```

Result Grid			
Filter Rows:			
	reg_num	model	year
▶	KA031181	Lancer	1957
	KA041702	Audi	2005
	KA052250	Indica	1990
	KA053408	Honda	2008
	KA095477	Toyota	1998

```
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 ;
```



```

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;

```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	driver_id	reg_num			
▶	A02	KA031181			
	A05	KA041702			
	A01	KA052250			
	A04	KA053408			
	A03	KA095477			

```

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);

```

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	report_num	accident_date	location		
▶	11	2003-01-01	Mysore Road		
	12	2004-02-02	South end Circle		
	13	2003-01-21	Bull temple Road		
	14	2008-02-17	Mysore Road		
	15	2004-03-05	Kanakpura Road		

```

select * from participated;

```

driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	30000
A02	KA053408	12	25000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000

Queries

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

```
update participated set
```

```
damage_amount=25000
```

```
where reg_num='KA053408' and report_num=12;
```

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

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	South end Circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore Road
15	2004-03-05	Kanakpura Road
16	2008-03-08	Domlur

- Display Accident date and location

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

accident_date	location
2003-01-01	mysore road
2004-02-02	south end
2003-01-21	bull temple road
2003-02-17	mysore road
2003-03-15	kanakpura ROAD

To-Do

- **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;

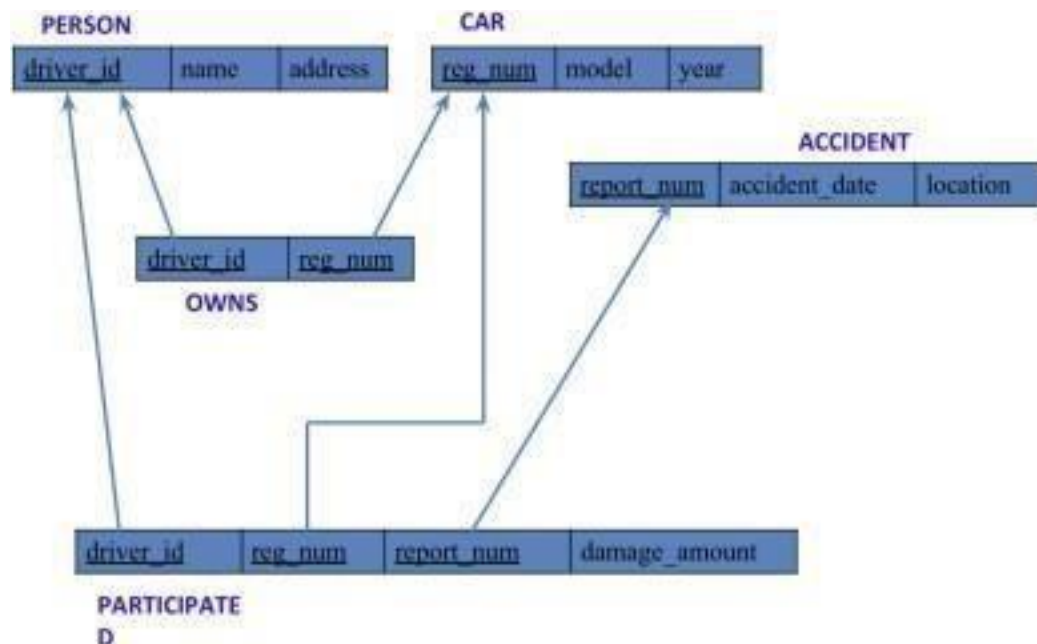
	MAX(DAMAGE_AMOUNT)
▶	25000

More Queries on Insurance Database

(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)
- 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).
- 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_074 order by year asc;
```

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

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

```
select model, count(model) from participated_074 car_074 where  
participated_074.reg_no = car_074.reg_no group by model;
```

model	count(mod...
Lancer	1
Audi	1
Indica	1
Honda	1
Toyota	1

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

```
select participated_18.driver_id as driver_id from accident_074,  
participated_074 where  
accident_074.report_no = participated_074.report_no and  
participated_074.damage_amt >= 25000;
```

driver_id
A02
A03

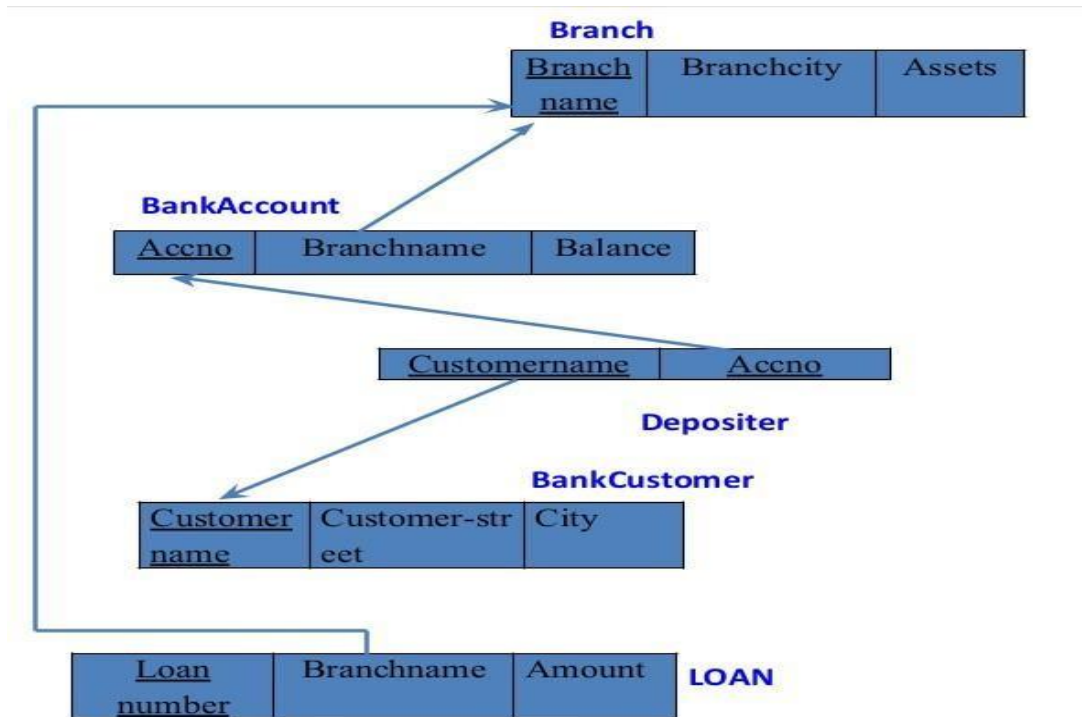
Bank Database

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



DATABASE :

```
create database Bank_Database; show databases; use Bank_Database; create table Branch(Name
varchar(20),City varchar(20), Assets varchar(20), primary key(Name));
```

```
create table BankAccount(accno varchar(20),Name varchar(20), Balance varchar(20), primary
key(accno,Name),foreign key(Name) references Branch(Name)) ; create table Customer(name
varchar(20),Street varchar(20), City varchar(20),primary key(name));
```

```
create table Depositer(name varchar(20),accno varchar(20),
primary key(name,accno), foreign key(name) references Customer(name),foreign key(accno)
references
BankAccount(accno));
```

```
create table Depositer(name varchar(20),accno varchar(20),
primary key(name,accno), foreign key(name) references Customer(name),foreign key(accno)
references
BankAccount(accno));
```

```
create table Loan(loan_no varchar(20),Name varchar(20), Amount varchar(20), primary
key(Name), foreign key(Name) references Branch(Name));
```

```
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_Jantarmanatar", "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_Jantarmanatar", 8000); insert into
BankAccount values(6, "SBI_ShivajiRoad", 8000); insert into
BankAccount values(8, "SBI_ResidencyRoad", 8000); insert into
BankAccount values(9, "SBI_ParlimentRoad", 8000); insert into
BankAccount values(10, "SBI_ResidencyRoad", 8000); insert into
BankAccount values(11, "SBI_Jantarmanatar", 8000); insert into
Customer values("Avinash", "Bull temple road","Bangalore" );
insert into Customer values("Dinesh", "Bannerghatta
Road","Bangalore" ); insert into Customer values("Mohan",
"NationalCollegeRoad","Bangalore" ); insert into Customer
values("Nikhil", "Akbar Road","Delhi" ); insert into Customer
values("Ravi", "Prithviraj Road","Delhi" );
```

```
insert into Depositer values("Avinash", 1);
insert into Depositer values("Dinesh", 2);
insert into Depositer values("Mohan", 3);
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 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_Jantarmanatar", 5000);
```

```
select * from Branch;
```


	Name	City	Assets
▶	SBI_Chamrajpet	Bangalore	50000
	SBI_Jantarmantar	Delhi	20000
	SBI_ParlimentRoad	Delhi	10000
	SBI_ResidencyRoad	Bangalore	10000
	SBI_ShivajiRoad	Bombay	20000

select * from BankAccount;

	accno	Name	Balance
▶	1	SBI_Chamrajpet	2000
	10	SBI_ResidencyRoad	8000
	11	SBI_Jantarmantar	8000
	2	SBI_ResidencyRoad	5000
	3	SBI_ShivajiRoad	6000
	4	SBI_ParlimentRoad	9000
	5	SBI_Jantarmantar	8000
	6	SBI_ShivajiRoad	8000
	8	SBI_ResidencyRoad	8000
	9	SBI_ParlimentRoad	8000

select * from Customer;

	name	Street	City
▶	Avinash	Bull temple road	Bangalore
	Dinesh	Bannerghatta Road	Bangalore
	Mohan	NationalCollegeRoad	Bangalore
	Nikhil	Akbar Road	Delhi
	Ravi	Prithviraj Road	Delhi
•	NULL	NULL	NULL

select * from Depositer;

	name	accno
▶	Avinash	1
	Dinesh	10
	Nikhil	11
	Dinesh	2
	Mohan	3
	Nikhil	4
	Ravi	5
	Avinash	8
	Nikhil	9
*	HULL	HULL

select * from Loan;

	loan_no	Name	Amount
▶	1	SBI_Chamrajpet	1000
	5	SBI_Jantarmanatar	5000
	4	SBI_ParliamentRoad	4000
	2	SBI_ResidencyRoad	2000
	3	SBI_ShivajiRoad	3000
*	HULL	HULL	HULL

Queries

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

alter table branch

change assets assents_inlakhs real;

	branch_name	branch_city	assents_inlakhs
▶	SBI_Chamrajpet	Bangalore	50000
	SBI_Jantarmanatar	Delhi	20000
	SBI_MantriMarg	Delhi	200000
	SBI_ParliamentRoad	Delhi	10000
	SBI_ResidencyRoad	Bangalore	10000
	SBI_ShivajiRoad	Bombay	20000
*	HULL	HULL	HULL

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;

	customer_name
►	Dinesh

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

create view br

as

select branch_name, sum(amount)

from loan group by branch_name;

select * from br

Result Grid			Filter Rows:
	branch_name	sum(amount)	
►	SBI_Chamrajpet	1000	
	SBI_Jantarmantar	5000	
	SBI_ParliamentRoad	4000	
	SBI_ResidencyRoad	2000	
	SBI_ShivajiRoad	3000	

More Queries on Bank Database

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

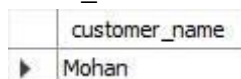
```
select distinct d.customer_name
from Depositer d, BranchAccount ba, Branch b where
d.accno=ba.accno and ba.branch_name=b.branch_name and
b.branch_city="Delhi"
group by d.customer_name having count(b.branch_name)>1;
```



The screenshot shows a 'Result Grid' with a 'Filter Rows' input field. Below the grid, a table displays the results of the query.

customer_name
Niki

2. Find all customers who have a loan at the bank but do not have an account. select b.customer_name from borrower b where b.loan_number not in(select d.accno from depositer d where b.loan_number=d.accno);



The screenshot shows a table with the results of the query.

customer_name
Mohan

3. Find all customers who have both an account and a loan at the Bangalore branch select b.customer_name from borrower b where b.loan_number in(select d.accno from depositer d,branchaccount ba, branch b where b.loan_number=d.accno and d.accno=ba.accno and ba.branch_name=b.branch_name and b.branch_city="Bangalore");



The screenshot shows a table with the results of the query.

customer_name
Avinash
Dinesh



4. Find the names of all branches that have greater assets than all branches located in Bangalore. select branch_name from branch where assents_inlakhs > all (select assents_inlakhs from branch where branch_city="Bangalore");

Result Grid		Filter Rows:
	branch_name	
▶	SBI_MantriMarg	
•	NULL	

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

delete from branchaccount



where ba.branch_name=(select b.branch_name from branch b where branch_city="Bombay"); select * from branchaccount;

Result Grid			 Filter Rows:
	accno	branch_name	balance
▶	1	SBI_Chamrajpet	2100
	2	SBI_ResidencyRoad	5250
	4	SBI_ParliamentRoad	9450
	5	SBI_Jantarmantra	8400
	8	SBI_ResidencyRoad	4200
	9	SBI_ParliamentRoad	3150
	10	SBI_ResidencyRoad	5250
	11	SBI_Jantarmantra	2100
	12	SBI_MantriMarg	2100
•	NULL	NULL	NULL

6. Update the Balance of all accounts by 5%

update BranchAccount

set balance=balance+((5*balance)/100) where accno in(1,2,4,5,8,9,10,11,12);

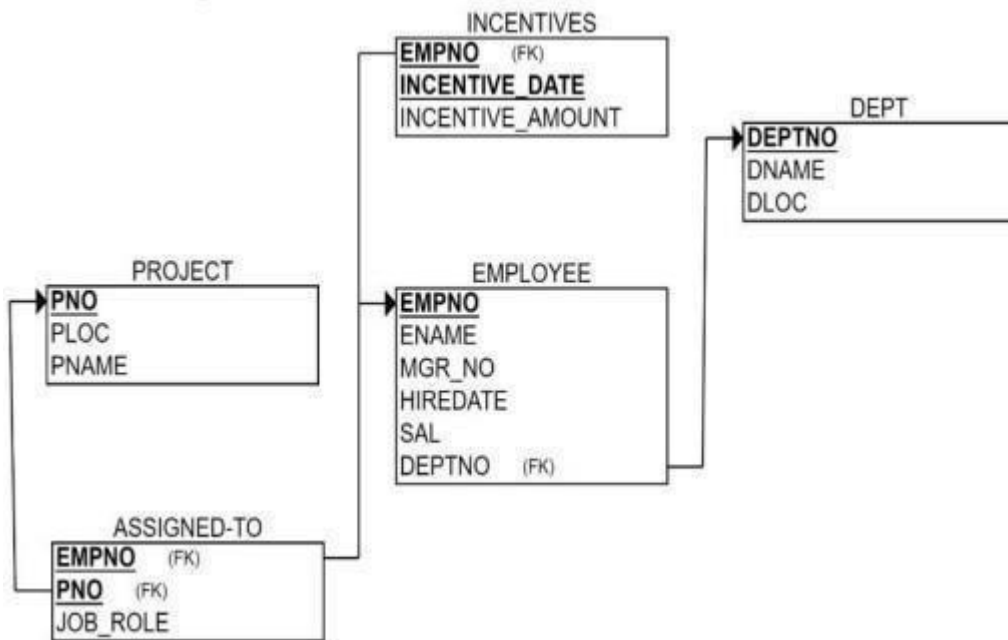
Result Grid			 Filter Rows:	
	accno	branch_name	balance	
▶	1	SBI_Chamrajpet	2205	
	2	SBI_ResidencyRoad	5512.5	
	4	SBI_ParliamentRoad	9922.5	
	5	SBI_Jantarmantra	8820	
	8	SBI_ResidencyRoad	4410	
	9	SBI_ParliamentRoad	3307.5	
	10	SBI_ResidencyRoad	5512.5	
	11	SBI_Jantarmantra	2205	
	12	SBI_MantriMarg	2205	
•	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.

Schema diagram



DATABASE :

```
create database Employee_Database; use
Employee_Database;
```

```
create table dept(no varchar(20) primary key, dname varchar(20), dloc varchar(20));
```

```
create table employee(empno int,ename varchar(20),mgr_no int,hiredate varchar(20), sal float, no
varchar(20),primary key(empno,no), foreign key(no) references dept(no));
```

```
create table incentives(empno int, date VARCHAR(20), amt float,primary key(empno,date),foreign
key(empno) references employee(empno));
```

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

```
create table Assingnedto(empno int, pno int,job_role text, primary key(empno,pno), foreign  
key(empno)  
references employee(empno), foreign key(pno) references project(pno));
```

```
insert into dept values(1,"cse","pj");  
insert into dept values(2,"ise","pj");  
insert into dept values(3,"csds","pg");  
insert into dept values(4,"ece","pg");  
insert into dept values(5,"aiml","pj");
```

```
insert into employee values(101,"mdr",100,"12/01/1999",100000,1);  
insert into employee values(201,"sak",200,"17/01/2020",50000,2);  
insert into employee values(301,"grp",100,"01/09/2004",30000,3);  
insert into employee values(401,"sws",101,"03/08/2000",10000,4);  
insert into employee values(501,"sks",101,"29/2/2008",90000,5);
```

```
insert into incentives values(101,"12/03/2004",50000);  
insert into incentives values(201,"17/03/2024",25000);  
insert into incentives values(301,"01/12/2019",15000);  
insert into incentives values(401,"03/11/2019",5000);  
insert into incentives values(501,"29/4/2019",45000);
```

```
insert into project values(10,"bng","chatbot");  
insert into project values(40,"delhi","ml model");  
insert into project values(50,"bombay","blockchain");  
insert into project values(30,"chennai","stocks");  
insert into project values(80,"mysore","android app");  
insert into Assingnedto values(101,10,"devops");  
insert into Assingnedto values(201,40,"sde");  
insert into Assingnedto values(301,50,"manager");  
insert into Assingnedto values(401,30,"jpa");  
insert into Assingnedto values(501,80,"pa");
```

```
select * from dept;
```

	no	dname	dloc
▶	1	cse	pj
	2	ise	pj
	3	csds	pg
	4	ece	pg
	5	aiml	pj
•	NULL	NULL	NULL

select * from employee;

	empno	ename	mgr_no	hiredate	sal	no
▶	101	mdr	100	12/01/1999	100000	1
	201	sak	200	17/01/2020	50000	2
	301	grp	100	01/09/2004	30000	3
	401	sws	101	03/08/2000	10000	4
	501	sks	101	29/2/2008	90000	5
•	NULL	NULL	NULL	NULL	NULL	NULL

select * from incentives;

	empno	date	amt
▶	101	12/03/2004	50000
	201	17/03/2024	25000
	301	01/12/2019	15000
	401	03/11/2019	5000
	501	29/4/2019	45000
•	NULL	NULL	NULL

select * from project;

	pno	ploc	pname
▶	10	bng	chatbot
	30	chennai	stocks
	40	delhi	ml model
	50	bombay	blockchain
	80	mysore	android app
•	NULL	NULL	NULL

select * from Assignedto;



	empno	pno	job_role
▶	101	10	devops
	201	40	sde
	301	50	manager
	401	30	jpa
	501	80	pa
•	NULL	NULL	NULL

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

```
select a.empno
```

```
from assignedto a
```

```
where a.pno=any(select pno from project where ploc in('Bengaluru','Hyderabad','Mysuru'));
```

Result Grid			Filter Rows:	
	empno			
▶	104			
	106			

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

```
select e.empno from employee e where e.empno != all(select i.empno from incentives i);
```

Result Grid			 Filter Rows: <input type="text"/>
	empno		
	106		
	102		
	NULL		

3. 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.empno,e.ename,d.dname,a.job_role,d.dloc,p.ploc
```

```
from employee e, dept d, assignedto a, project p
```

```
where e.deptno=d.deptno and e.empno=a.empno and a.pno=p.pno
```

Result Grid						
		Filter Rows:		Export:		Wrap Cell Cor
	empno	ename	dname	job_role	dloc	ploc
▶	101	Raj	HR	Developer	New Delhi	New Delhi
	103	Adi	HR	Analyst	New Delhi	New Delhi
	102	Priyam	Finance	Manager	Mumbai	Mumbai
	104	Asha	IT	Tester	Bangalore	Bangalore

More Queries on Employee Database

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

select ename from employee where mgr_no = (select max(mgr_no) from employee);

	ename
▶	sak

2. Display those managers name whose salary is more than average salary of his employee select ename from employee where sal > (select avg(sal) from employee);

	ename
▶	mdr
	sks

3. Find the name of the second top level managers of each department. select ename from employee where sal = (select max(sal) from employee where sal < (select max(sal) from employee));

	ename
▶	sks

4. Find the employee details who got second maximum incentive in January 2019. select * from employee where empno = (select empno from incentives where amt = (select max(amt) from incentives where amt < (select max(amt) from incentives)));

	empno	ename	mgr_no	hiredate	sal	no
▶	501	sks	101	29/2/2008	90000	5
*	NULL	NULL	NULL	NULL	NULL	NULL

Supplier Database

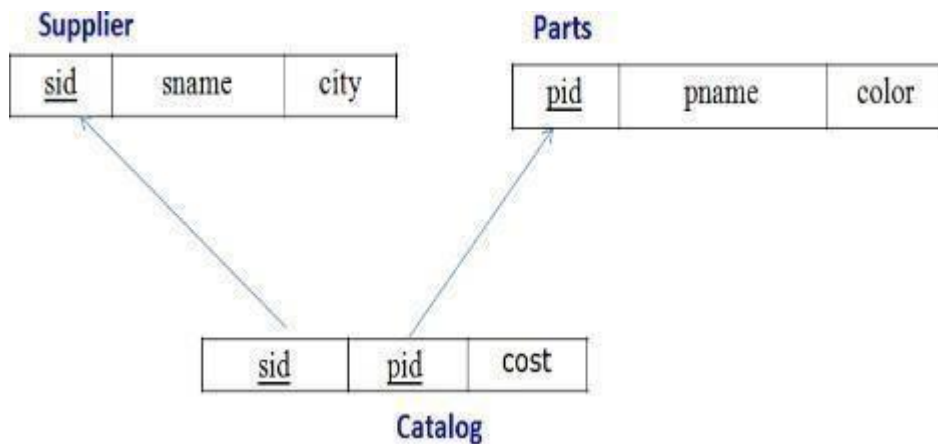
Question

(Week 7)

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2. Insert appropriate records in each table.
3. Find the pnames of parts for which there is some supplier.
4. Find the snames of suppliers who supply every part.

5. Find the snames of suppliers who supply every red part.
6. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
7. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).
8. For each part, find the sname of the supplier who charges the most for that part.

Schema diagram



1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys. create database s; use s;

```

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

```

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

```

create table Catalog(
sid int, pid
int,
cost int,
  
```

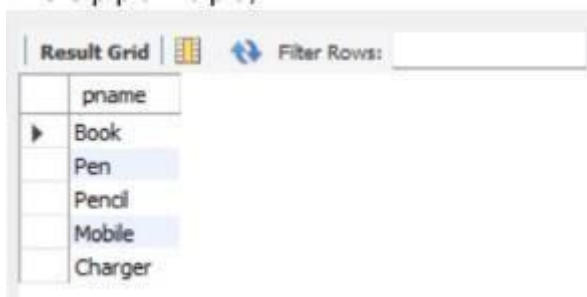
foreign key(sid) references Supplier(sid), foreign
key(pid) references Parts(pid));

Insert appropriate records in each table. insert into Supplier
values (10001, 'Acme Widget','Bangalore'); insert into Supplier
values (10002, 'Johns','Kolkata'); insert into Supplier values
(10003, 'Vimal','Mumbai'); insert into Supplier values (10004,
'Reliance','Delhi');

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

insert into Catalog values (10001, 20001 , 10);
insert into Catalog values (10001, 20002 , 10);
insert into Catalog values (10001, 20003 , 30);
insert into Catalog values (10001, 20004 , 10);
insert into Catalog values (10001, 20005 , 10);
insert into Catalog values (10002, 20001 , 10);
insert into Catalog values (10002, 20002 , 20);
insert into Catalog values (10003, 20003 , 30);
insert into Catalog values (10004, 20003 , 40);

3. Find the pname of parts for which there is some supplier. select distinct
p.pname from Parts p, Catalog c where p.pid = c.pid;



	pname
▶	Book
	Pen
	Pencil
	Mobile
	Charger

4. Find the snames of suppliers who supply every part. select distinct s.sname
from Catalog c , Supplier s where c.sid = s.sid and
NOT EXISTS(select p.pid from Parts p where
NOT EXISTS(select c1.sid from Catalog c1 where
c1.sid=c.sid and c1.pid =c.pid));

	sname
▶	Acme Widget
	Johns
	Vimal
	Reliance

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

```
select distinct s.sname
from Catalog C, Supplier s
where C.sid=s.sid and
NOT EXISTS (select P.pid from Parts P
where P.color="Red" and NOT EXISTS (select C1.sid from Catalog C1 where
C1.sid = C.sid and C1.pid = P.pid and P.color="Red"));
```

	sname
▶	Acme Widget
	Johns

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

```
select p.pname
from Parts p, Catalog c, Supplier s where p.pid=c.pid and
c.sid=s.sid and s.sname="Acme Widget" and NOT EXISTS
(select * from Catalog c1, Supplier s1 where p.pid=c1.pid and
c1.sid=s1.sid and s1.sname != "Acme Widget");
```

	pname
▶	Mobile
	Charger

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).

```
select distinct C.sid from Catalog C where C.cost > (select AVG(C1.cost)
from Catalog C1 where C1.pid = C.pid);
```

	sid
▶	10002
	10004

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

```
select P.pid, S.sname from Parts P,  
Supplier S, Catalog C where C.pid  
= P.pid and C.sid = S.sid and  
C.cost = (select max(C1.cost)  
from Catalog C1 where  
C1.pid = P.pid);
```

	sname
▶	Acme Widget
	Johns
	Reliance

No SQL – STUDENT DATABASE

Question

(Week 8)

Perform the following DB operations using MongoDB.

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

STRUCTURE OF THE COLLECTION

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

QUERIES

- Create a database “Student” with the following attributes Rollno, age, contactNo, Email-Id.

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

```
show dbs
```

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

```
Atlas atlas-mozg5o-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
Atlas atlas-mozg5o-shard-0 [primary] test> show dbs
Student   72.00 KiB
test      8.00 KiB
admin     328.00 KiB
local     88.62 GiB
Atlas atlas-mozg5o-shard-0 [primary] test> |
```

Insert appropriate values

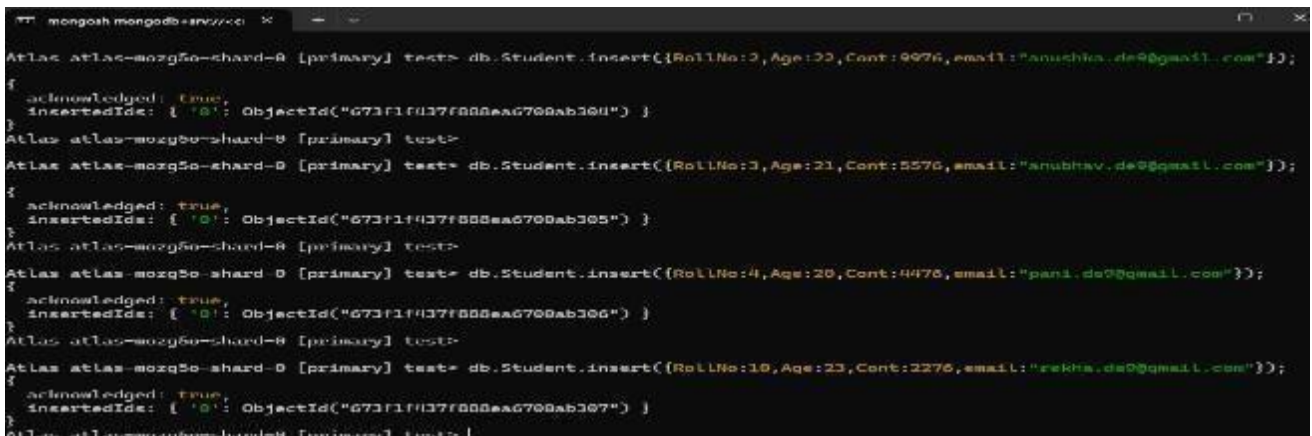
```
db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
```

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

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

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

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



```
Atlas atlas-mozg5o-shard-0 [primary] test> db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
{ acknowledged: true,
  insertedIds: { '_id': ObjectId("6731f1f437f808a6700ab304") } }
Atlas atlas-mozg5o-shard-0 [primary] test>
Atlas atlas-mozg5o-shard-0 [primary] test> db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
{ acknowledged: true,
  insertedIds: { '_id': ObjectId("6731f1f437f808a6700ab305") } }
Atlas atlas-mozg5o-shard-0 [primary] test>
Atlas atlas-mozg5o-shard-0 [primary] test> db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{ acknowledged: true,
  insertedIds: { '_id': ObjectId("6731f1f437f808a6700ab306") } }
Atlas atlas-mozg5o-shard-0 [primary] test>
Atlas atlas-mozg5o-shard-0 [primary] test> db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{ acknowledged: true,
  insertedIds: { '_id': ObjectId("6731f1f437f808a6700ab307") } }
Atlas atlas-mozg5o-shard-0 [primary] test> |
```

- Write a query to update the Email-Id of a student with rollno 5.

```
db.Student.update({RollNo:10},{ $set:
{email:"Abhinav@gmail.com"}})
```



```

mongo> use test
switched to db test
mongo> db.Student.insertMany([
  { RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("673f1f437f888ea6700ab306"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("673f1f437f888ea6700ab307"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  }
])
Atlas atlas-mozg5o-shard-0 [primary] test> db.Student.update({RollNo:10},{set:{email:"Abhinav@gmail.com"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-mozg5o-shard-0 [primary] test>

```

- Replace the student name from “ABC” to “FEM” of rollno 11.
`db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})`

```

{
  _id: ObjectId("63bfd4de56eba0e23c3a5c78"),
  RollNo: 11,
  Age: 22,
  Name: 'FEM',
  Cont: 2276,
  email: 'rea.de9@gmail.com'
}

```

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

_id	RollNo	Age	Cont	email	Name
6746b6c4f73fea43f1	1	21	9876	antara.de9@gmail.com	
6746b6cbf73fea43f1	2	22	9976	anushka.de9@gmail.com	
6746b6d2f73fea43f1	3	21	5576	anubhav.de9@gmail.com	
6746b6d8f73fea43f1	4	20	4476	pani.de9@gmail.com	
6746b6def73fea43f1	10	23	2276	Abhinav@gmail.com	
6746b710f73fea43f1	11	22	2276	rea.de9@gmail.com	FEM

NO SQL – CUSTOMER DATABASE

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

Create Table:

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

Inserting Values:

```
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"}]);
```

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

Atlas atlas-zkql51-shard-0 [primary] test> db.createCollection("Customer");
{ ok: 1 }
Atlas atlas-zkql51-shard-0 [primary] test> 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("674ff20946b4cd1ffe0d55a3"),
    '1': ObjectId("674ff20946b4cd1ffe0d55a4"),
    '2': ObjectId("674ff20946b4cd1ffe0d55a5"),
    '3': ObjectId("674ff20946b4cd1ffe0d55a6"),
    '4': ObjectId("674ff20946b4cd1ffe0d55a7")
  }
}
```

Finding all checking accounts with balance greater than 12000

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

```
Atlas atlas-zkq151-shard-0 [primary] test> db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});
[
  {
    _id: ObjectId("674ff20946b4cd1ffe0d55a4"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("674ff20946b4cd1ffe0d55a5"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  }
]
```

Finding the maximum and minimum balance of each customer

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

```
Atlas atlas-zkq151-shard-0 [primary] test> db.Customer.aggregate([{$group: {_id:"$custid", minBal:{$min:$min:"$acc_bal"}, maxBal:{$max:"$acc_bal"}}}]);
[
  { _id: 5, minBal: 2000, maxBal: 2000 },
  { _id: 3, minBal: 50000, maxBal: 50000 },
  { _id: 4, minBal: 10000, maxBal: 10000 },
  { _id: 1, minBal: 10000, maxBal: 20000 }
]
```

Dropping collection “Customer”

```
db.Customer.drop();
```

```
[test> db.Customer.drop();
true]
```

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

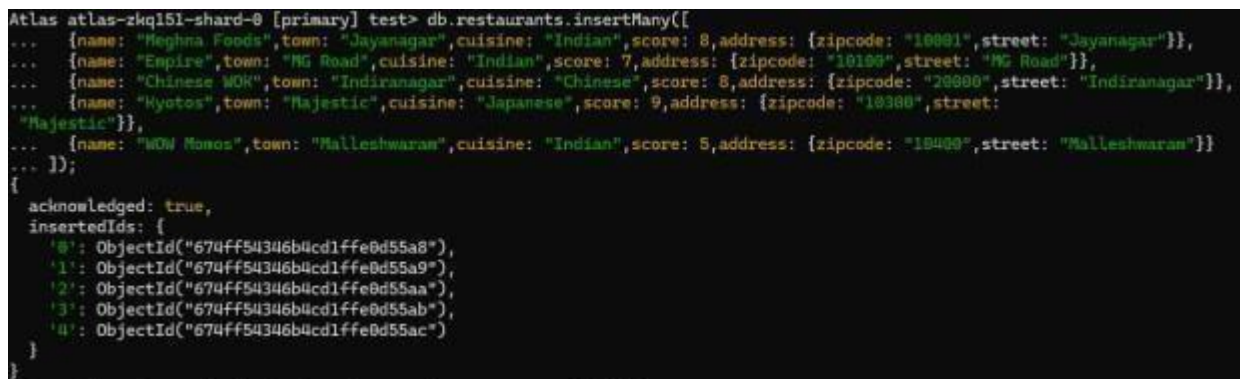
_id	custid	acc_bal	acc_type
674ff20946b4cd1ffe	1	10000	Saving
674ff20946b4cd1ffe	1	20000	Checking
674ff20946b4cd1ffe	3	50000	Checking
674ff20946b4cd1ffe	4	10000	Saving
674ff20946b4cd1ffe	5	2000	Checking

NO SQL – 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'.

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

```
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" }
} ])
```



```
Atlas atlas-zkq151-shard-0 [primary] test> 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("674ff54346b4cd1ffe0d55a8"),
    '1': ObjectId("674ff54346b4cd1ffe0d55a9"),
    '2': ObjectId("674ff54346b4cd1ffe0d55aa"),
    '3': ObjectId("674ff54346b4cd1ffe0d55ab"),
    '4': ObjectId("674ff54346b4cd1ffe0d55ac")
  }
}
```

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

```
db.restaurants.find({})
```

```

Atlas atlas-zkq151-shard-0 [primary] test> db.restaurants.find({})
[
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55a8"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55a9"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55aa"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 8,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55ab"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  }
]

```

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

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

```

Atlas atlas-zkql5l-shard-0 [primary] test> db.restaurants.find({}).sort({ name: -1 })
[
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55ac"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55a8"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55ab"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55a9"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("674ff54346b4cd1ffe0d55aa"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 8,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]

```

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

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

```

atlas atlas-zkql5l-shard-0 [primary] test> db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })
{
  _id: ObjectId("674ff54346b4cd1ffe0d55a8"),
  name: 'Meghna Foods',
  town: 'Jayanagar',
  cuisine: 'Indian'
},
{
  _id: ObjectId("674ff54346b4cd1ffe0d55a9"),
  name: 'Empire',
  town: 'MG Road',
  cuisine: 'Indian'
},
{
  _id: ObjectId("674ff54346b4cd1ffe0d55aa"),
  name: 'Chinese WOK',
  town: 'Indiranagar',
  cuisine: 'Chinese'
},
{
  _id: ObjectId("674ff54346b4cd1ffe0d55ab"),
  name: 'Kyotos',
  town: 'Majestic',
  cuisine: 'Japanese'
},
{
  _id: ObjectId("674ff54346b4cd1ffe0d55ac"),
  name: 'WOW Momos',
  town: 'Malleshwaram',
  cuisine: 'Indian'
}

```

Query to find the average score for each restaurant

```

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

```

```

atlas atlas-zkql5l-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } }
... ])
{ _id: 'Chinese WOK', average_score: 8 },
{ _id: 'Kyotos', average_score: 9 },
{ _id: 'Meghna Foods', average_score: 8 },
{ _id: 'WOW Momos', average_score: 5 },
{ _id: 'Empire', average_score: 7 }

```

Query to find the name and address of the restaurants that have a zipcode that starts with '10'.

```

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

```

```

atlas atlas-zkql5l-shard-0 [primary] test> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
{ name: 'Meghna Foods', address: { street: 'Jayanagar' } },
{ name: 'Empire', address: { street: 'MG Road' } },
{ name: 'Kyotos', address: { street: 'Majestic' } },
{ name: 'WOW Momos', address: { street: 'Malleshwaram' } }

```


_id	name	town	cuisine	score	address.zipcode	address.street
674ff54346b4cd1ffe	Meghna Foods	Jayanagar	Indian	8	10001	Jayanagar
674ff54346b4cd1ffe	Empire	MG Road	Indian	7	10100	MG Road
674ff54346b4cd1ffe	Chinese WOK	Indiranagar	Chinese	8	20000	Indiranagar
674ff54346b4cd1ffe	Kyotos	Majestic	Japanese	9	10300	Majestic
674ff54346b4cd1ffe	WOW Momos	Malleswaram	Indian	5	10400	Malleswaram