**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**Database Management Systems (23CS3PCDBM)**

***Submitted by***

**Dhruva S Rao(1BM23CS092)**

***in partial fulfilment 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 **Dhruva S Rao(1BM23CS092)** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024. 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  Professor  Department of CSE, BMSCE | Dr. Kavitha Sooda  Professor HOD  Department of CSE, BMSCE |

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 09/10/2024 | Insurance Database | 04 |
| 2 | 09/10/2024 | More Queries on Insurance Database | 11 |
| 3 | 16/10/2024 | Bank Database | 14 |
| 4 | 23/10/2024 | More Queries on Bank Database | 19 |
| 5 | 30/10/2024 | Employee Database | 23 |
| 6 | 13/11/2024 | More Queries on Employee Database | 28 |
| 7 | 20/11/2024 | Supplier Database | 30 |
| 8 | 27/11/2024 | NO SQL - Student Database | 34 |
| 9 | 04/12/2024 | NO SQL - Customer Database | 37 |
| 10 | 04/12/2024 | NO SQL – Restaurant Database | 42 |

Insurance Database

**Question**

**(Week 1)**

**-** PERSON (driver\_id: String, name: String, address: String)

**-** CAR (reg\_num: String, model: String, year: int)

**-** ACCIDENT (report\_num: int, accident\_date: date, location: String)

**-** OWNS (driver\_id: String, reg\_num: String)

**-** PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)

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

**-** Display Accident date and location

**-** Update the damage amount to 25000 for the car with a specific reg\_num (example 'K A053408' ) for which the accident report number was 12.

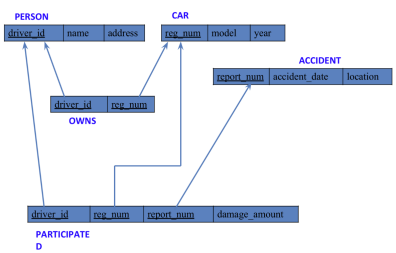
**-** Add a new accident to the database.

**-** To Do

**-** Display Accident date and location

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

**Schema Diagram**

**Create database**

create database insurance;

use insurance;

**Create table**

create table insurance.person(

driver\_id varchar(20),

name varchar(30),

address varchar(50),

PRIMARY KEY(driver\_id)

);

create table car(

reg\_num varchar(15),

model varchar(10),

year int,

PRIMARY KEY(reg\_num)

);

create table insurance.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.accident(

report\_num int,

accident\_date date,

location varchar(50),

PRIMARY KEY(report\_num)

);

create table insurance.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),

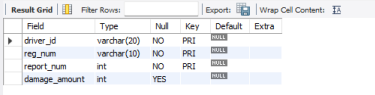
FOREIGN KEY(reg\_num) REFERENCES car(reg\_num),

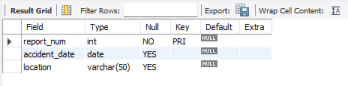
FOREIGN KEY(report\_num) REFERENCES accident(report\_num)

);

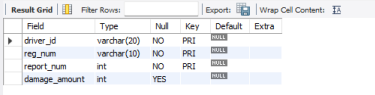
**Structure of the table**

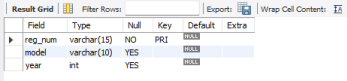
desc person;

desc accident;

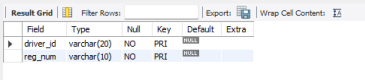


desc participated;

desc car;



desc owns;



**Inserting Values to the table**

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

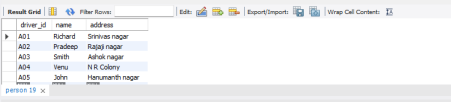
insert into person values("A02","Pradeep", "Rajaji nagar");

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;

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

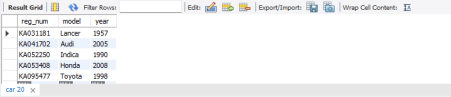
insert into car values("KA031181","Lancer", "1957");

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

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

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

select \* from car;



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

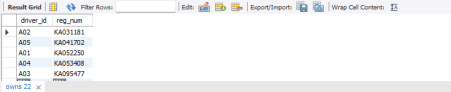
insert into owns values("A02","KA031181");

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");

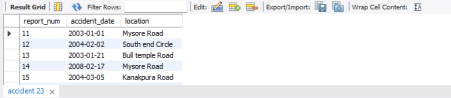
insert into accident values(12,'2004-02-02',"South end Circle");

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;

insert into participated values("A01","KA052250",11,10000);

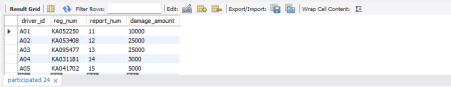
insert into participated values("A02","KA053408",12,50000);

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;



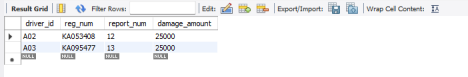
**Queries**

**- 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;

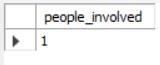


**● Find the total number of people who owned cars that were involved in accidents in 2008.**

select count(distinct driver\_id) CNT

from participated a, accident b

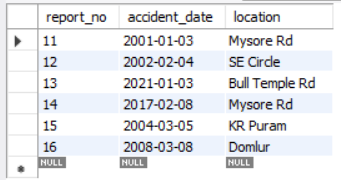
where a.report\_num=b.report\_num and b.accident\_date like '2008%';



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

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

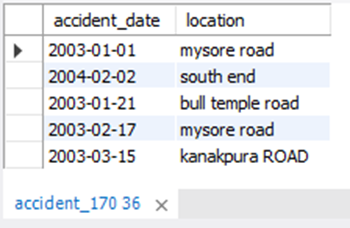
select \* from accident;



TO DO:

● DISPLAY ACCIDENT DATE AND LOCATION

select accident\_date as date, location from accident;



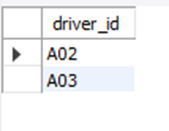
● DISPLAY DRIVER ID WHO DID ACCIDENT WITH DAMAGE

AMOUNT GREATER THAN OR EQUAL TO RS.25000

select participated\_204.driver\_id as driver\_id from accident\_204, participated\_204 where

accident\_204.report\_no = participated\_204.report\_no and participated\_204.damage\_amt >=

25000;



**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)

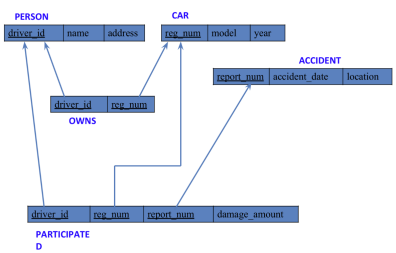
**-** PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)

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

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

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

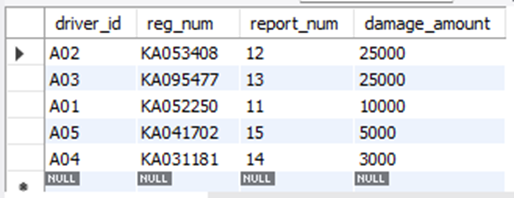
**Schema Diagram**

****

**Queries**

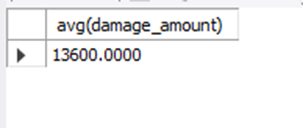
**LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER OF DAMAGE AMOUNT.**

SELECT \* FROM PARTICIPATED ORDER BY DAMAGE\_AMOUNTT DESC;



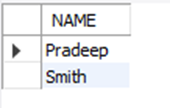
**FIND THE AVERAGE DAMAGE AMOUNT**

SELECT AVG(DAMAGE\_AMOUNTT) FROM PARTICIPATED;



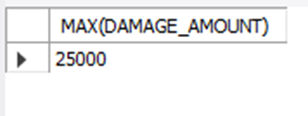
**LIST THE NAME OF DRIVERS WHOSE DAMAGE IS GREATER THAN THE AVERAGE DAMAGE AMOUNT.**

SELECT NAME FROM PERSON A, PARTICIPATED B WHERE A.DRIVER\_ID = B.DRIVER\_ID AND DAMAGE\_AMOUNT > (SELECT AVG(DAMAGE\_AMOUNT) FROM PARTICIPATED);



**FIND MAXIMUM DAMAGE AMOUNT.**

SELECT MAX(DAMAGE\_AMOUNT) FROM PARTICIPATED;



**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)

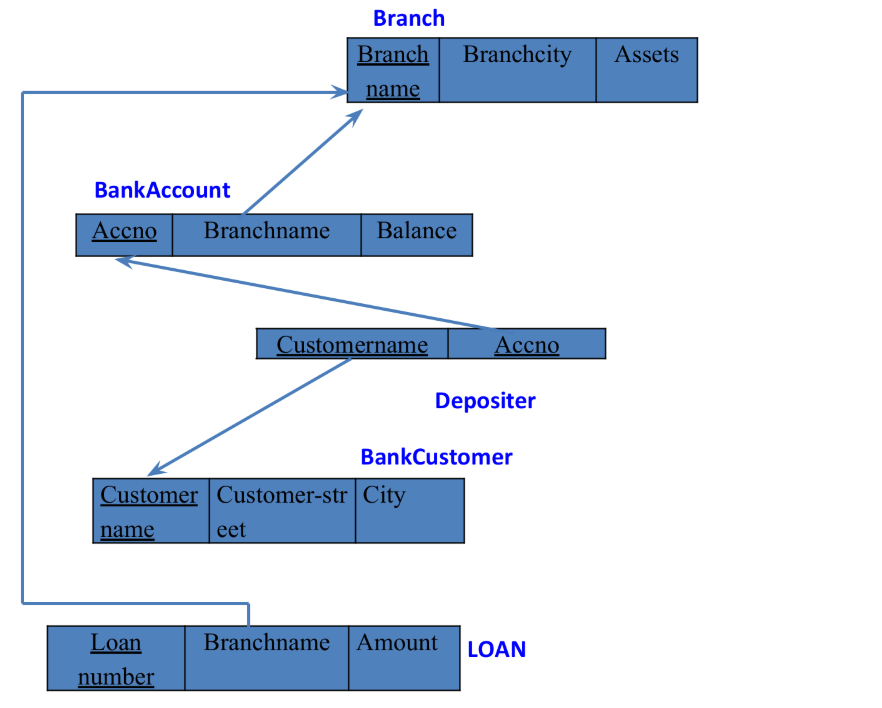
- Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation.

- 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 DB:**

create database bank170;

show databases;

use bank170;

**create tables:**

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 LOAN(

LOAN\_NO VARCHAR(20),

NAME VARCHAR(20),

AMOUNT VARCHAR(20),

PRIMARY KEY (NAME ),

foreign key (NAME ) references BRanch(name));

**inserting values:**

insert into branch values ("SBI\_chmarajpet","banglore",50000);

insert into branch values ("SBI\_residencyroad","banglore",10000);

insert into branch values ("SBI\_shivajiroad","bombay",20000);

insert into branch values ("SBI\_parlimentroad","delhi",10000);

insert into branch values ("SBI\_jantarmantar","delhi",20000);

insert into bankaccount values (1,"SBI\_chmarajpet",2000);

insert into bankaccount values (2,"SBI\_residencyroad",5000);

insert into bankaccount values (3,"SBI\_shivajiroad",6000);

insert into bankaccount values (4,"SBI\_parlimentroad",9000);

insert into bankaccount values (5,"SBI\_jantarmantar",8000);

insert into bankaccount values (6,"SBI\_shivajiroad",4000);

insert into bankaccount values (8,"SBI\_residencyroad",4000);

insert into bankaccount values (9,"SBI\_parlimentroad",3000);

insert into bankaccount values (10,"SBI\_residencyroad",5000);

insert into bankaccount values (11,"SBI\_jantarmantar",2000);

insert into customer values ("avinash","bull temple road","banglore");

insert into customer values ("dinesh","bannerghatta road","banglore");

insert into customer values ("mohan","national clg road","banglore");

insert into customer values ("nikhil","akbar road ","delhi");

insert into customer values ("ravi","prithvi raj 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\_chmarajpet",1000);

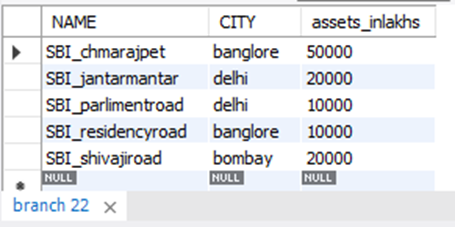
insert into loan values (2,"SBI\_residencyroad",2000);

insert into loan values (3,"SBI\_shivajiroad",3000);

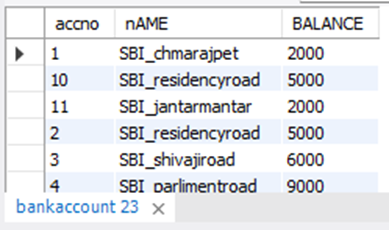
insert into loan values (4,"SBI\_parlimentroad",4000);

insert into loan values (5,"SBI\_jantarmantar",5000);

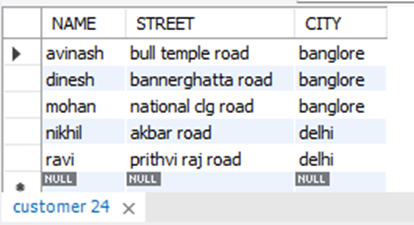
select \* from branch;



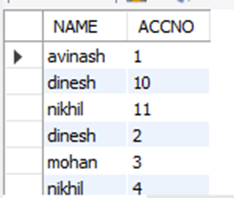
select \* from bankaccount;



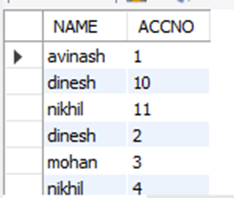
select \* from customer;



select \* from depositer;



select \* from loan;

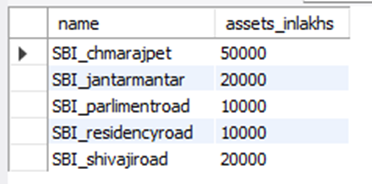


**Queries:**

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

select name,assets\_inlakhs from branch;

alter table branch rename column assets to assets\_inlakhs;

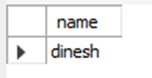


**● Find all the customers who have at least two accounts at the same branch (ex.SBI\_ResidencyRoad).**

select d.name from Depositer d, BankAccount b where

b.name='SBI\_ResidencyRoad' and d.Accno=b.Accno group by d.name having

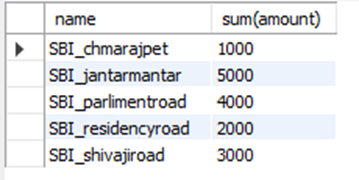
count(d.Accno)>=2;



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

create view br as select name,sum(amount) from loan group by name;

select \* from br;



**More Queries on Bank Database**

**Question**

**(Week 4)**

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

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

**-** BankCustomer (customer-name: String, customer-street: String, customer-city: String) **-** Depositer(customer-name: String, accno: int)

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

- Find all the customers who have an account at all the branches

- located in a specific city (Ex. Delhi).

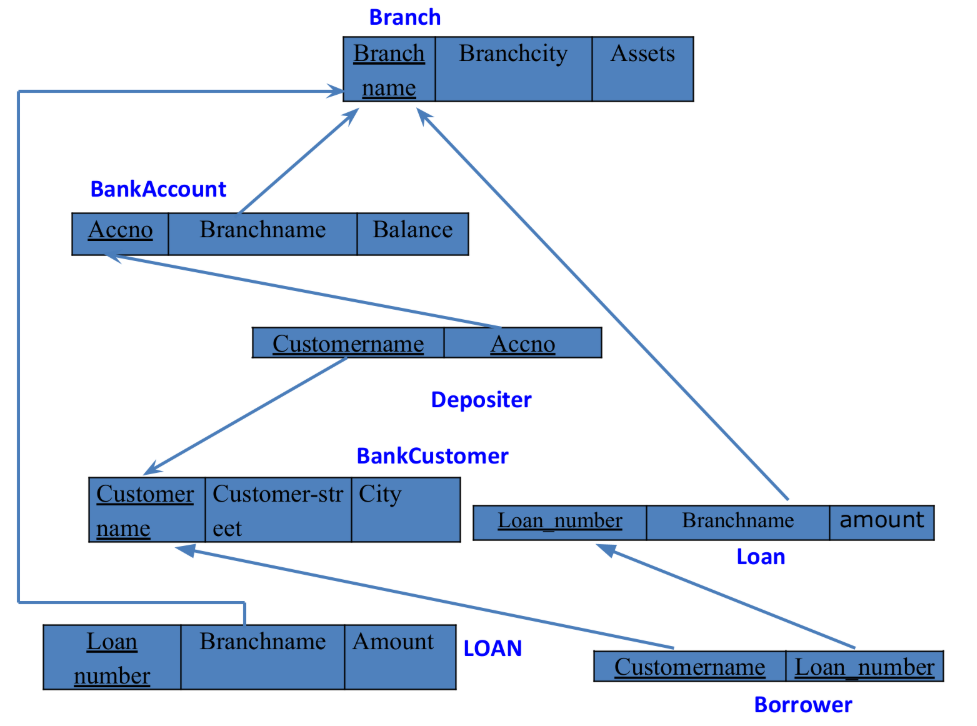
- Find all customers who have a loan at the bank but do not have an account. - Find all customers who have both an account and a loan at the Bangalore branch

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

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

- Update the Balance of all accounts by 5%

**Schema Diagram**



**Creating Table:**

**create table** borrower(

customer\_name varchar(20),

loan\_no int,

foreign **key**(customer\_name) **references** bank\_customer\_204(customer\_name), foreign **key**(loan\_no) **references** loans\_204(loan\_no)

);

**Inserting values:**

**insert into** branch **values** (**"SBI\_MantriMarg"**, **"Delhi"**, 200000);

**insert into** bank\_account **values** (12, **"SBI\_MantriMarg"**, 2000);

**insert into** deposits **values**(**"Nikhil"**, 12);

**insert into** borrower **values(**

(**"Avinash"**, 1),

(**"Dinesh"**, 2),

(**"Mohan"**, 3),

(**"Nikhil"**, 4),

(**"Ravi"**, 5));

**Queries**

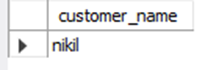
**● Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).**

select distinct d.customer\_name

from depositer d,bankAccount ba,branch b

where d.accno=ba.accno and ba.branch\_name=b.branch\_name and b.city='delhi'

group by d.customer\_name having count(b.branch\_name)>1;

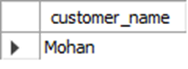


**● Find all customers who have a loan at the bank but do not have an account.**

select b.customer\_name

from borrower b

where b.loan\_no not in(select d.accno from depositer d where b.loan\_no=d.accno);



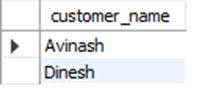
**● Find all customers who have both an account and a loan at the Bangalore branch.**

select b.customer\_name

from borrower b

where b.loan\_no in (select d.accno from depositer d,bankAccount ba,branch b

where b.loan\_no=d.accno and d.accno=ba.accno and ba.branch\_name=b.branch\_name and b.city='Bangalore');

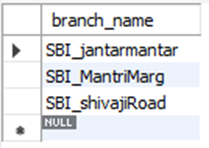


**● Find the names of all branches that have greater assets than all branches located in Bangalore.**

select branch\_name

from branch

where assets\_inlakhs>all(select assets\_inlakhs from branch where city='Bangalore');

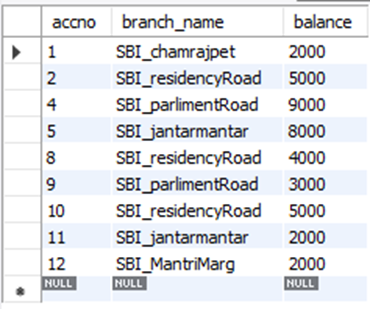


**● Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).**

delete from bankAccount ba

where ba.branch\_name=(select b.branch\_name from branch b where city='Bombay');

select \* from bankAccount;

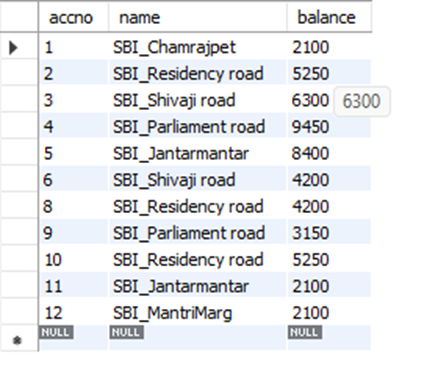


**● Update the Balance of all accounts by 5%**

update bankAccount

set balance=balance+((5\*balance)/100);

select \* from bankAccount;



**Employee Database**

**Question**

**(Week 5)**

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

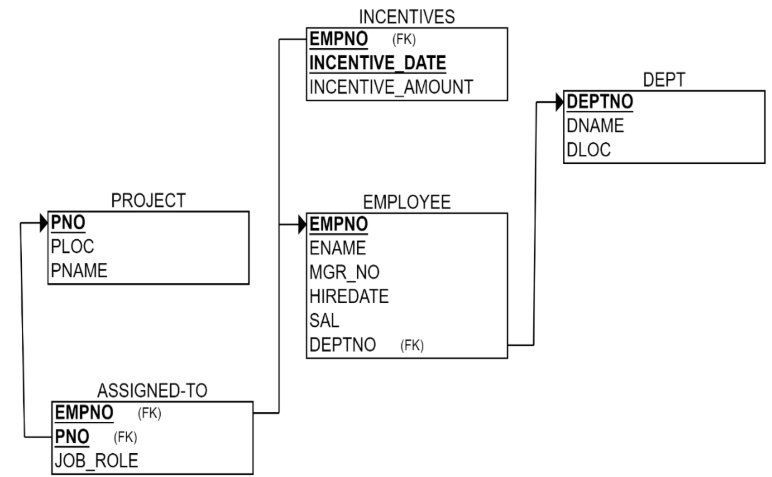
2. Enter greater than five tuples for each table.

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

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

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

**Schema Diagram**



**Create database**

**create database** employee\_database;

**use** employee\_database;

**Create table**

**create table** project(

pno int primary **key**,

ploc varchar(20),

pname varchar(20)

);

**create table** dept(

deptno int primary **key**,

dname varchar(30),

dloc varchar(30)

);

**create table** employee(

empno int primary **key**,

ename varchar(20),

mgr\_no int,

hiredate date,

sal **double**,

deptno int,

foreign **key**(deptno) **references** dept(deptno)

);

**create table** assigned\_to(

empno int primary **key**,

pno int,

job\_role varchar(20),

foreign **key**(empno) **references** employee(empno),

foreign **key**(pno) **references** projec(pno)

);

**create table** incentives(

empno int,

incentive\_date date primary **key**,

incentive\_amount **double**,

foreign **key**(empno) **references** employee(empno));

**insert values:**

**insert into** project **values(**

(1,**"bengaluru"**,**"abcd"**),

(2,**"hyderabad"**,**"bcda"**),

(3,**"bengaluru"**,**"abab"**),

(4,**"bengaluru"**,**"baba"**),

(5,**"hyderabad"**,**"cdcd"**),

(6, **"mysuru"**, **"efef")**);

**insert into** dept **values (**

(1,**"cse"**,**"bengaluru"**),

(2,**"ise"**,**"hyderabad"**),

(3,**"ece"**,**"bengaluru"**),

(4,**"ete"**,**"hyderabad"**),

(5,**"ime"**,**"bengaluru"**),

(6, **"mech"**, **"mysuru")**);

**insert into** employee **values (**

(1,**"a"**,**null**,**"2023-11-9"**,70000,1),

(2,**"b"**,2,**"2023-8-9"**,70000,1),

(3,**"c"**,3,**"2023-6-8"**,70000,2),

(4,**"d"**,**null**,**"2023-8-6"**,70000,2),

(5,**"e"**,**null**,**"2023-5-4"**,70000,3),

(6, **"f"**, **null**, **"2023-6-1"**, 90000, 6));

**insert into** incentives **values (**

(1,**"2023-12-9"**,10000),

(2,**"2023-8-9"**,10000),

(3,**"2023-6-8"**,10000),

(4,**"2023-5-4"**,10000),

(5,**"2023-12-8"**,10000));

**insert into** assigned\_to **values (**

(1,1, **"employee"**),

(2,1, **"manager"**),

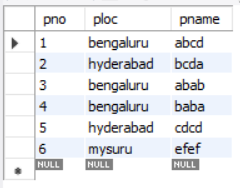
(3,2, **"manager"**),

(4,3, **"employee"**),

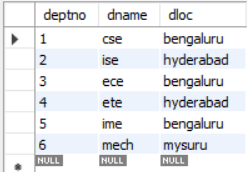
(5,4, **"employee"**),

(6, 6, **"employee")**);

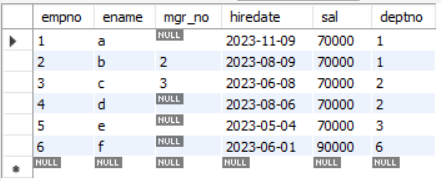
select \* from project;



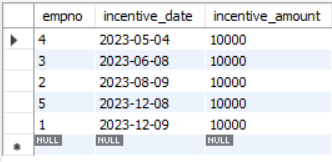
select \* from dept;



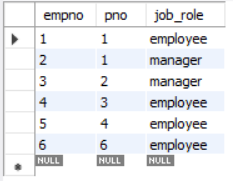
select \* from employee;



select \* from incentivest;



select \* from project;



**Queries**

**● Retrieve the employee numbers of all employees who work on**

**project located in Bengaluru, Hyderabad, or Mysuru.**

**select** assigned\_to.empno **from** assigned\_to, project

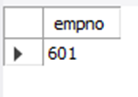
**where** assigned\_to.pno = project.pno **and** project.ploc **in** (**"bengaluru"**, **"mysuru"**, **"hyderabad"**);



**● Get Employee ID’s of those employees who didn’t receive incentives**

select empno from employee where empno not in (select empno

from incentives);

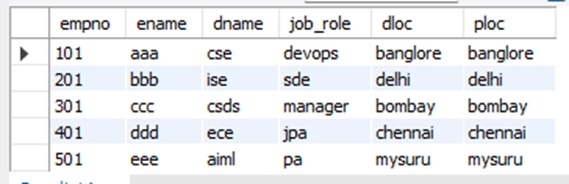


**● Write a SQL query to find the employees name, number, dept, job\_role, department location and project location who are working for a project location same as his/her department location.**

**select** employee.empno, ename, dname, job\_role, dloc, ploc

**from** employee, assigned\_to, project, dept

**where** ploc = dloc **and** assigned\_to.empno = employee.empno

**and** employee.deptno = dept.deptno **and** project.pno = assigned\_to.pno; 

**More Queries on Employee Database**

**Question**

**(Week 6)**

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

2. Enter greater than five tuples for each table.

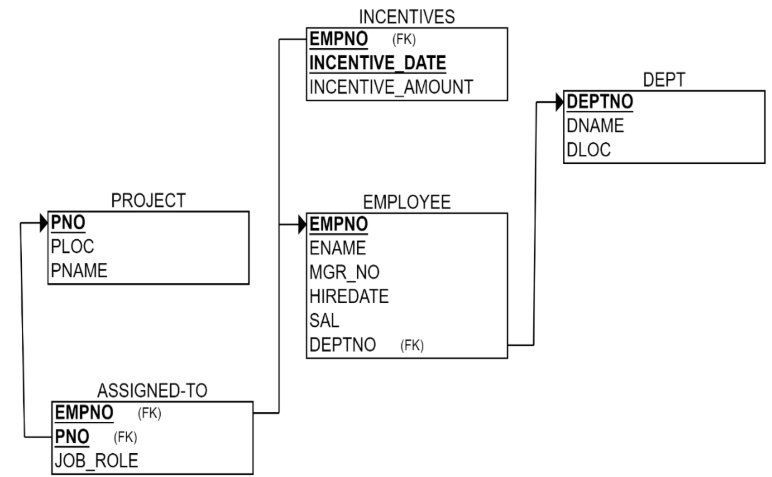
3. List the name of the managers with the maximum employees

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

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

6. Find the employee details who got the second maximum incentive in January 2019. 7. Display those employees who are working in the same department where his the manager is working.

**Schema Diagram**

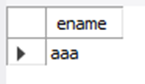


**Queries**

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

select e.ename from employee e where e.empno in(select mgr\_no from employee group by mgr\_no having count(\*)=

(select max(emp\_count) from (select count(\*) as emp\_count from employee group by mgr\_no)AS SUBQUERY));

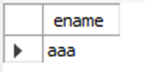


**● Display those managers name whose salary is more than average salary of his employee**

select e.ename from employee e

where e.sal>(select avg(sub.sal) from employee sub

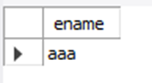
where sub.mgr\_no=e.empno);



**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));

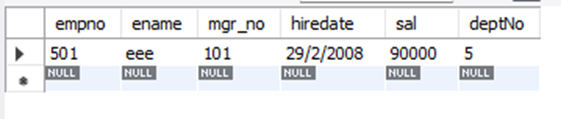


**● Find the employee details who got second maximum incentive in January 2019.**

select \* from employee where empno=(select empno from incentives where amount=

(select max(amount) from incentives where amount<

(select max(amount) from incentives)));



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

select E.ename from employee e , employee m WHERE e.mgr\_no=m.empno aND e.deptno=m.deptno;



**Supplier Database**

**Question**

**(Week 7)**

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

keys and the foreign keys.

2. Insert appropriate records in each table.

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

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

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

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

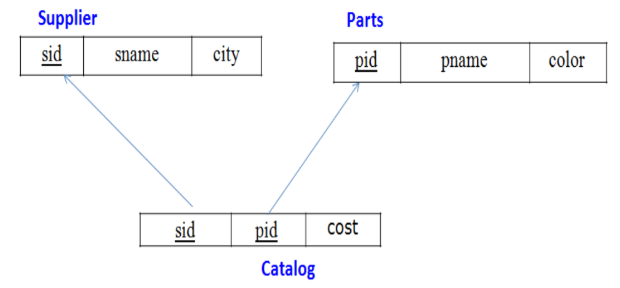
one else.

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

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

that part.

**Schema Diagram**



**Create DB:**

create database sup\_170;

use sup\_170;

**Create Table:**

create table supplier(

sid int,

sname varchar(20),

city varchar(20),

primary key(sid));

create table parts(

pid int,

pname varchar(20),

color varchar(20),

primary key(pid));

create table catalog(

sid int,pid int, cost int,

foreign key(sid) references supplier(sid),

foreign key(pid) references parts(pid));

**Inserting values:**

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);

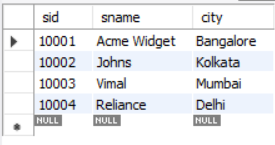
insert into Catalog values (10002, 20001 , 10);

insert into Catalog values (10002, 20002 , 20);

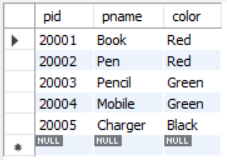
insert into Catalog values (10003, 20003 , 30);

insert into Catalog values (10004, 20003 , 40);

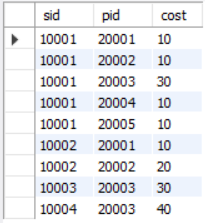
select \* from supplier;



select \* from supplier;



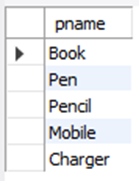
select \* from supplier;



**Queries**

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

select distinct p.pname from parts p , catalog c where p.pid=c.pid;

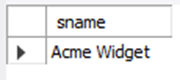


**● Find the snames of suppliers who supply every part.**

select s.sname from supplier s

where NOT exists(select p.pid from

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



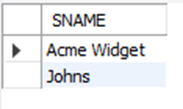
**● Find the snames of suppliers who supply every red part.**

SELECT S.SNAME FROM SUPPLIER S

WHERE NOT EXISTS( SELECT P.PID FROM PARTS P

WHERE P.COLOR="RED" AND NOT EXISTS (

SELECT C.SID FROM CATALOG C WHERE C.SID=S.SID AND C.PID=P.PID));

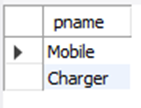
****

**● 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

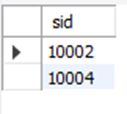
p.pid=c.Pid and c.sid=s.sid and s.sname="acme widget" and not exists (

select \* from catalog c1,supplier s1 where p.pid=c1.Pid AND C1.SID=s1.sid and s1.sname!="acme widget");

****

**● 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);

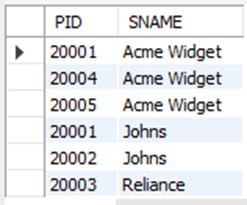


**● For each part, find the sname of the supplier who charges the most for that part.**

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);



**NoSQL Lab 1**

**Question**

**(Week 8)**

Perform the following DB operations using MongoDB.

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

2. Insert appropriate values

3. Write query to update Email-Id of a student with rollno 10.

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

5. Export the created table into local file system

6. Drop the table

7. Import a given csv dataset from local file system into

mongodb collection.

**Create database**

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

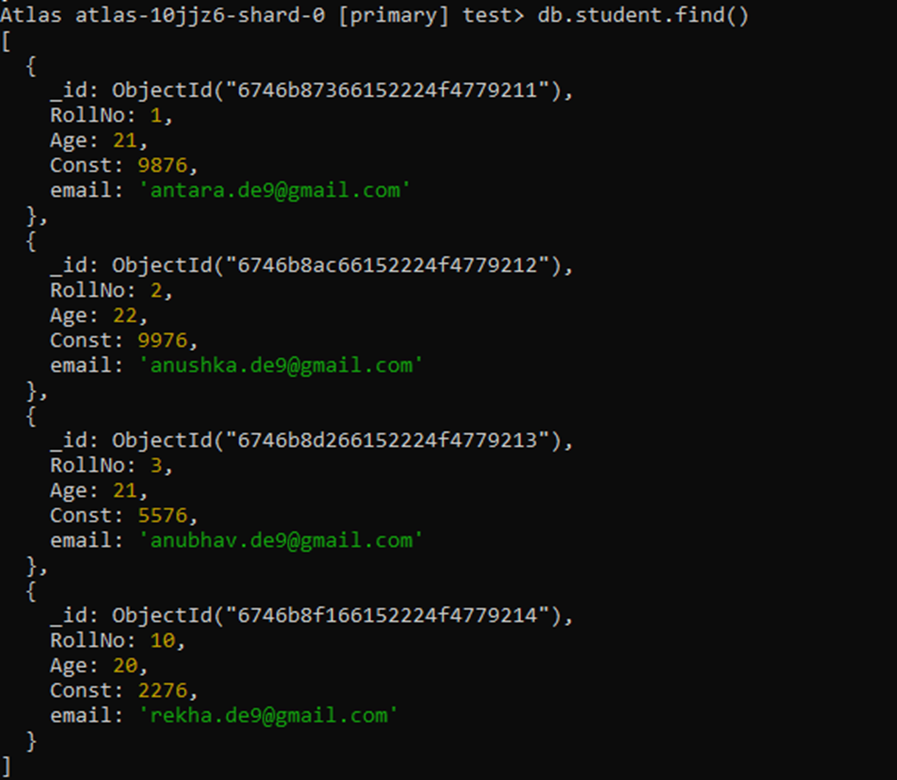
**Create table & Inserting Values to the table**

db.Student.insertMany([{**rollno:**1,**age:**21,**cont:**9876,**email:"prannay@gmail.com"**},{**rollno:**2,**a ge:**22,**cont:**9976,**email:"sohan@gmail.com"**},

{**rollno:**3,**age:**21,**cont:**5576,**email:"farhan@gmail.com"**},

{**rollno:**4,**age:**20,**cont:**4476,**email:"sakshi@gmail.com"**},{**rollno:**5,**age:**23,**cont:**2276,**email:"sa nika@gmail.com"**}]);

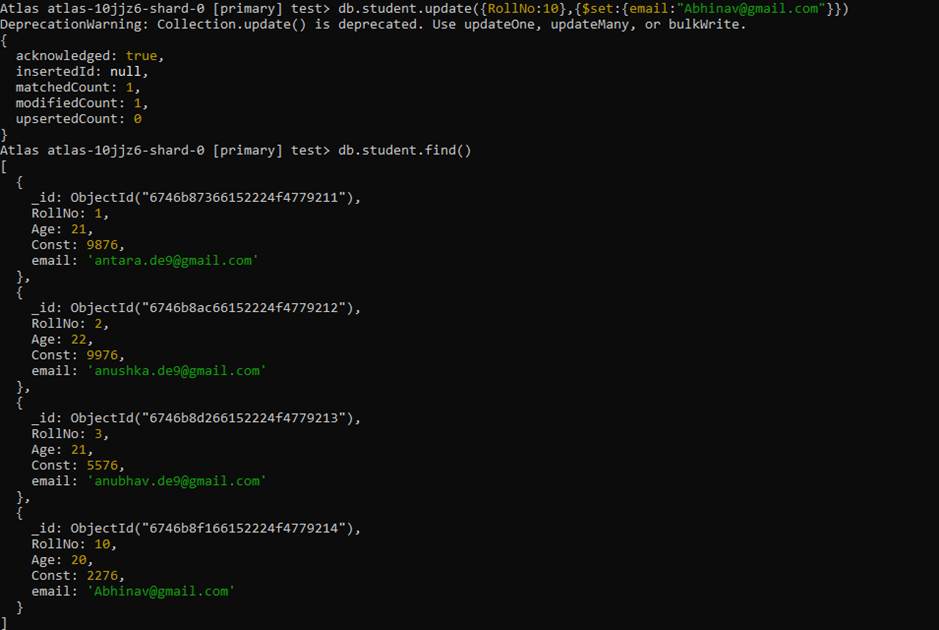
db.student.find()



**Queries**

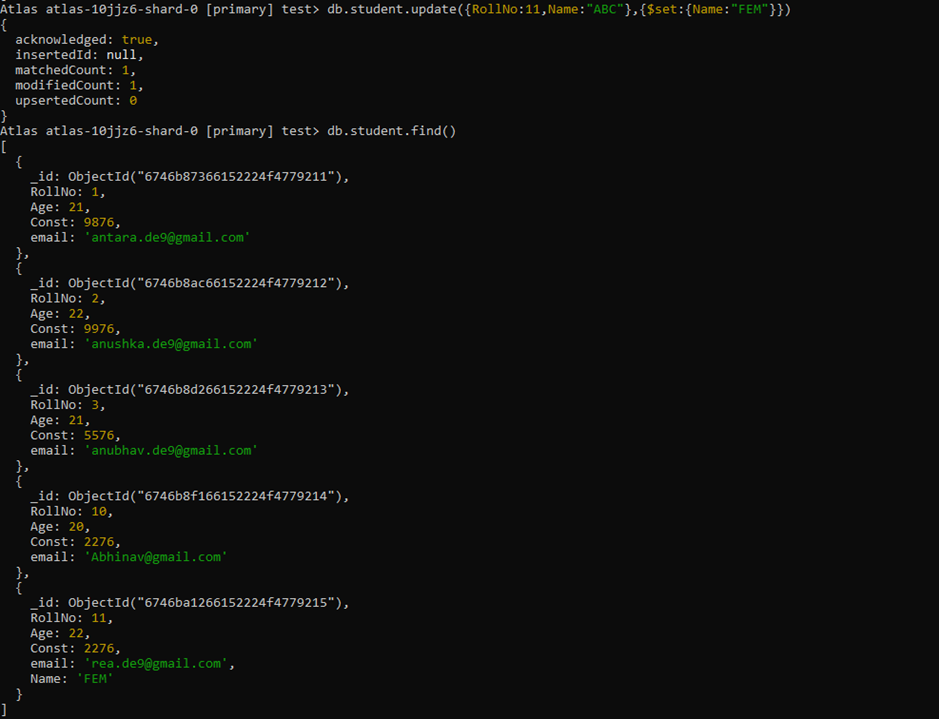
46

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



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

db.Student.insert({rollno:11,age:22,name:"ABC",cont:2276,email:"madhura@gmail.com"}); db.Student.update({rollno:11,name:"ABC"},{$set:{name:"FEM"}})



**NoSQL Lab 2**

**Question**

**(Week 9)**

Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes.

Cust\_id, Acc\_Bal, Acc\_Type

2. Insert at least 5 values into the table

3. Write a query to display those records whose total account balance

is greater than 1200 of account type ‘Checking’ for each customer\_id.

4. Determine Minimum and Maximum account balance for each

customer\_id.

5. Export the created collection into local file system

6. Drop the table

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

collection.

**Create Table:**

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

**Inserting Values:**

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,

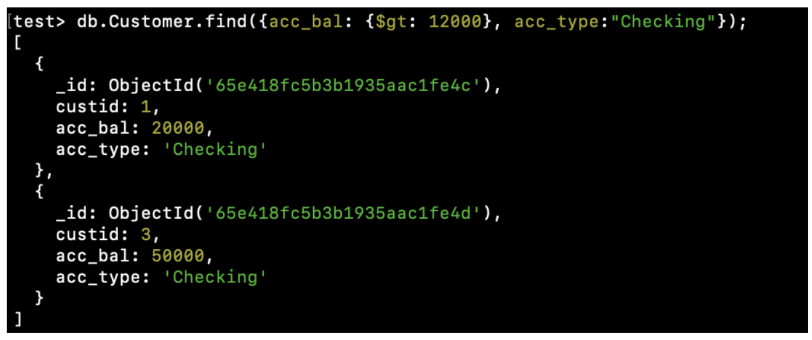
acc\_type: "Saving"}, {custid: 5, acc\_bal:2000, acc\_type: "Checking"}]);

db.student.find();



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

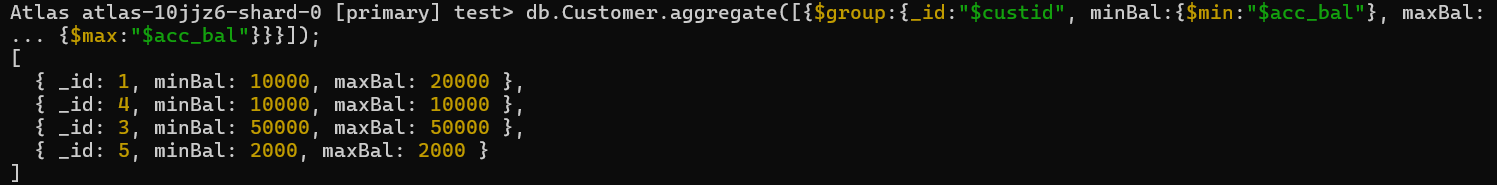
db.Customer.find({acc\_bal: {$gt: 12000}, acc\_type:"Checking"});



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

db.Customer.aggregate([{$group:{\_id:"$custid", minBal:{$min:"$acc\_bal"}, maxBal:

{$max:"$acc\_bal"}}}]);



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

mongoexport mongodb+srv://Likhith:@cluster0.xbmgopf.mongodb.net/test --collection=Customer -- out D:\1BM23CS170\st.json

50

**● Dropping collection “Customer”**

db.Customer.drop();

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

mongoimport mongodb+srv://Likhith:@cluster0.xbmgopf.mongodb.net/test --collection=Customer --file D:\1BM23CS170\st.json



db.Customer.find();



**NoSQL Lab 3**

**Question**

**(Week 10)**

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

collection restaurants.

2. Write a MongoDB query to arrange the name of the restaurants in

descending along with all the columns.

3. Write a MongoDB query to find the restaurant Id, name, town and

cuisine for those restaurants which achieved a score which is not

more than 10.

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

restaurant.

5. Write a MongoDB query to find the name and address of the

restaurants that have a zipcode that starts with '10'.

**Creating Table:**

db.createCollection(**"Restaurant"**);

db.restaurants.insertMany([

{name:"Meghna Foods",town:"Jayanagar", cuisine: "Indian", score: 8, address: { zipcode: "10001", street: "Jayanagar"}},

{ name: "Empire", town: "MG Road", cuisine: "Indian", score: 7, address: { zipcode: "10100", street: "MG Road" } },

{ name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: { zipcode: "20000", street: "Indiranagar"}},

{ name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode: "10300", street: "Majestic" } },

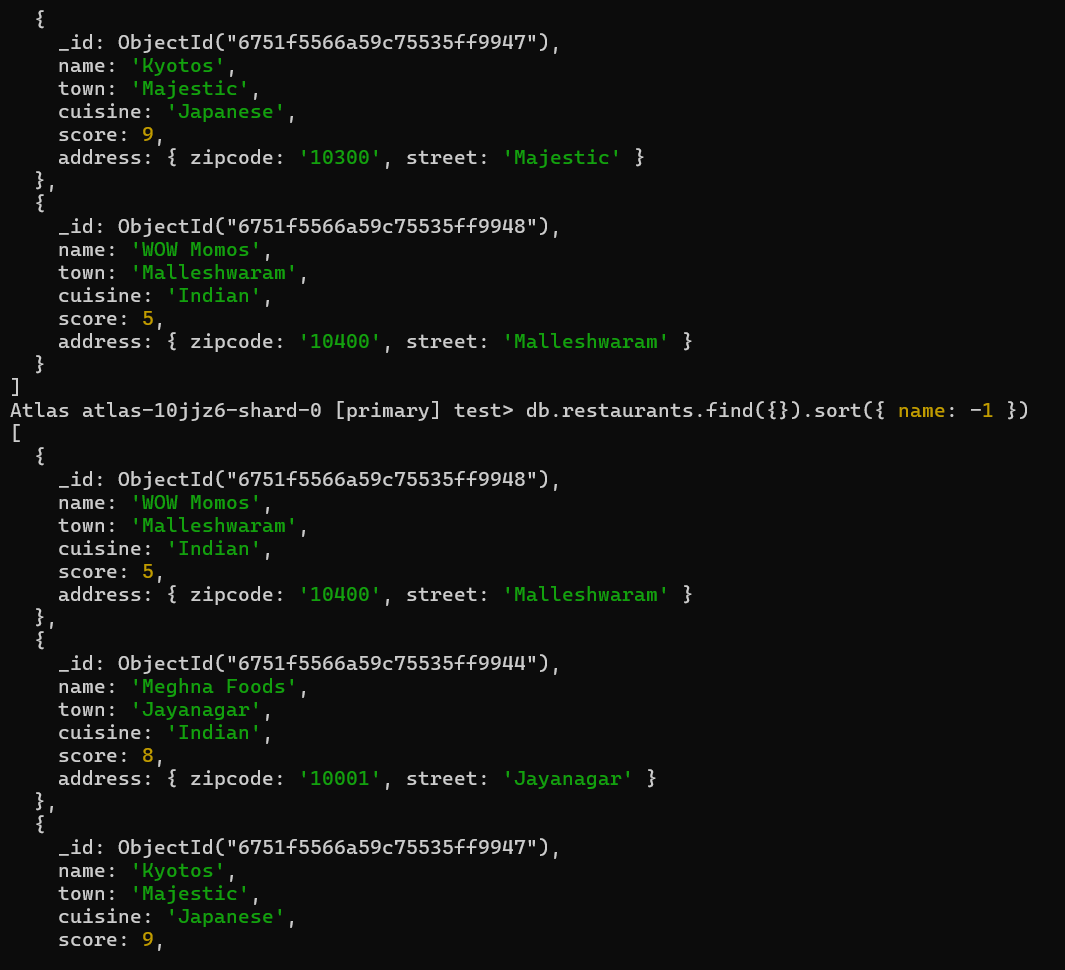
{ name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram" }}])

**QUERIES**

1) db.Restraunt.find()



2) db.Restraunt.find().sort({ "name": -1 });

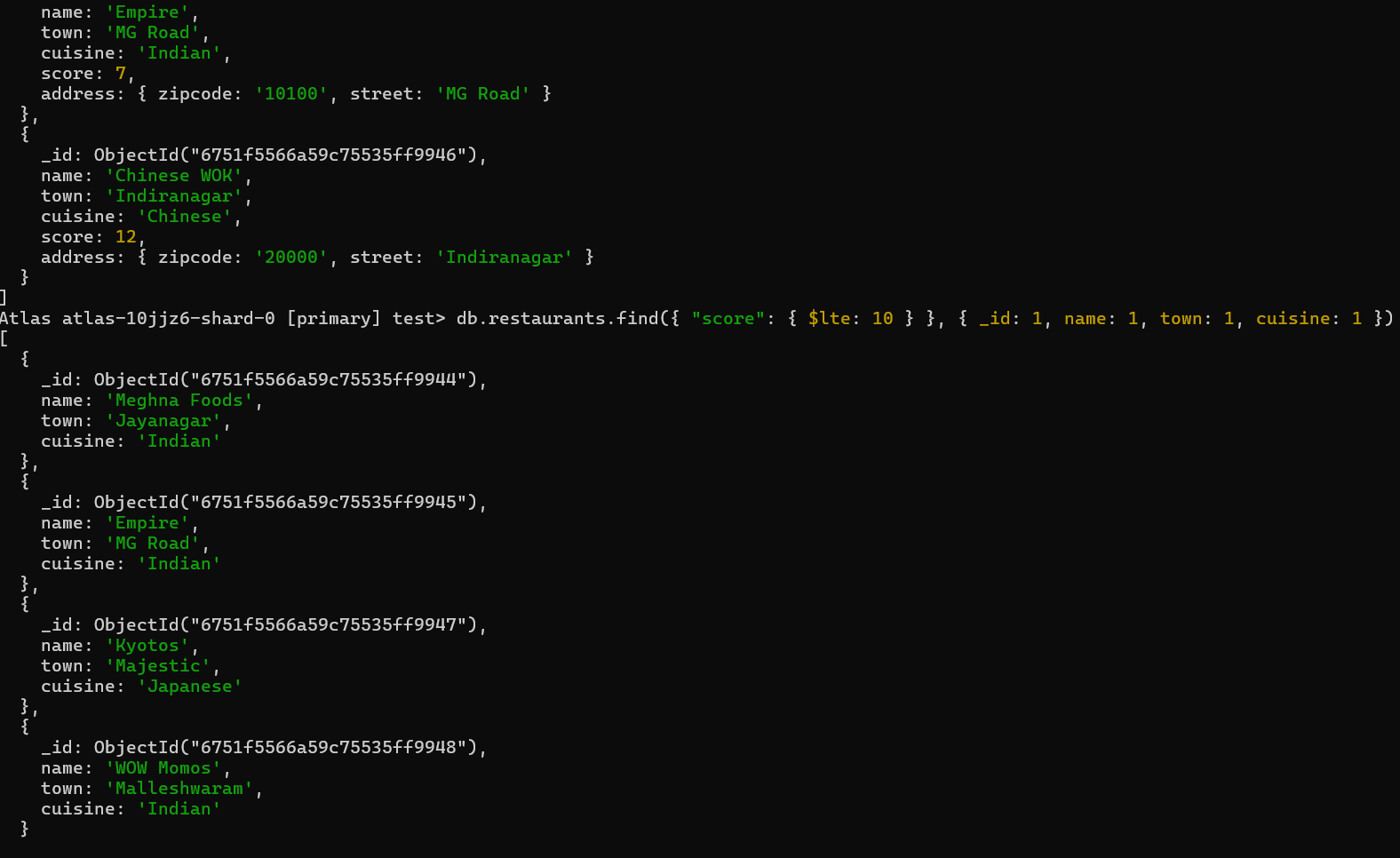


3) db.Restraunt.find(

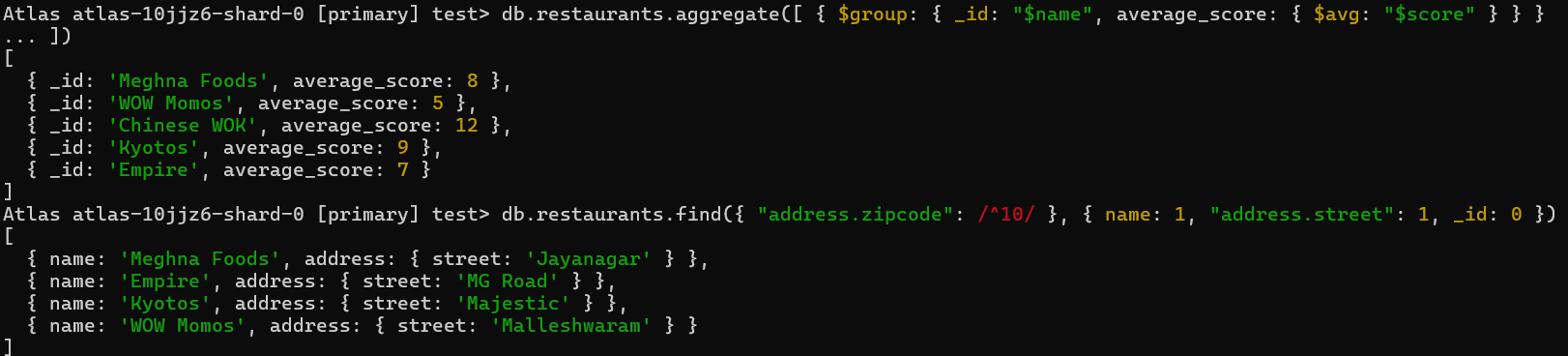
{ "grades.score": { $lte: 10 } },

{ \_id: 1, name: 1, town: 1, cuisine: 1, restaurant\_id: 1 }

);



4)db.restaurants.aggregate([ { $group: { \_id: "$name", average\_score: { $avg: "$score" }}}])



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

