**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**Database Management Systems (23CS3PCDBM)**

***Submitted by***

**Dhanush K**

**(1BM24CS404)**

***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 **Dhanush K (1BM24CS404),** 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 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.

|  |  |
| --- | --- |
| Lab faculty Incharge **Kayarvizhy N**  Assistant Professor  Department of CSE, BMSCE | Professor  Department of CSE, BMSCE |

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 |  | Insurance Database | 4-11 |
| 2 |  | More Queries on Insurance Database | 12-14 |
| 3 |  | Bank Database | 15-22 |
| 4 |  | More Queries on Bank Database | 23-26 |
| 5 |  | Employee Database | 27-33 |
| 6 |  | More Queries on Employee Database | 34-38 |
| 7 |  | Supplier Database | 39-43 |
| 8 |  | NO SQL - Student Database | 44-47 |
| 9 |  | NO SQL - Customer Database | 48-51 |
| 10 |  | NO SQL – Restaurant Database | 52-57 |

**Insurance Database**

**Question**

**(Week 1)**

**Consider the Insurance database given below.**

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)

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

ii. Enter at least five tuples for each relation

iii. Display Accident date and location

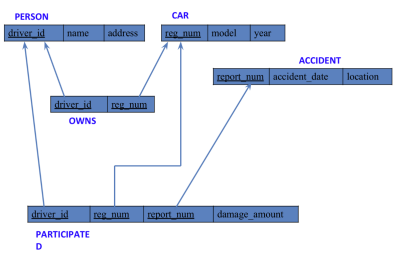
iv. Update the damage amount to 25000 for the car with a specific reg\_num (example 'KA053408' ) for which the accident report number was 12.

iv. Add a new accident to the database.

v. Display Accident date and location

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

**Schema Diagram**

**Create database**

create database insurance\_24BECS405;

use insurance\_24BECS405;

**Create table**

**(Create the above tables by properly specifying the primary keys and the foreign keys)**

create table person (driver\_id varchar(10),

name varchar(20),address varchar(30),primary key(driver\_id));

create table car(reg\_num varchar(10),

model varchar(10),year int ,

primary key(reg\_num));

create table accident(report\_num int,accident\_date date,

location varchar(20),

primary key(report\_num));

create table owns(driver\_id varchar(10),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 participated(driver\_id varchar(10),reg\_num varchar(10),

report\_num int,damage\_amount int,

primary key(driver\_id,reg\_num,report\_num),

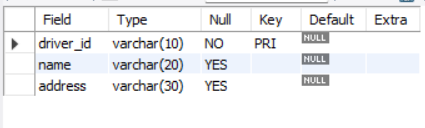
foreign key(driver\_id)references person(driver\_id),

foreign key(reg\_num)references car(reg\_num),

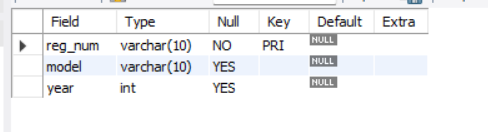
foreign key(report\_num)references accident(report\_num));

**Structure of the table**

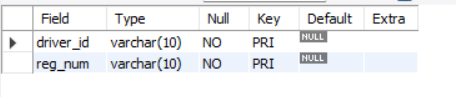
desc person;



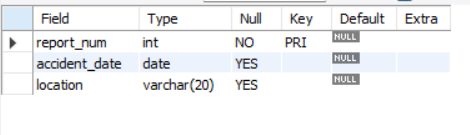
desc car;



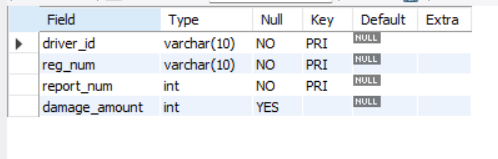
desc owns;



desc accident;



desc participated;



**Inserting Values to the table**

**(Enter at least five tuples for each relation)**

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

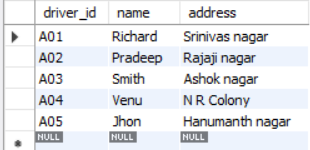
("A02","Pradeep","Rajaji nagar"),

("A03","Smith","Ashok nagar"),

("A04","Venu","N R Colony"),

("A05","Jhon","Hanumanth nagar");

select \* from person;



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

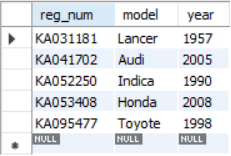
("KA031181","Lancer",1957),

("KA095477","Toyote",1998),

("KA053408","Honda",2008),

("KA041702","Audi",2005);

Select \* from car;



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

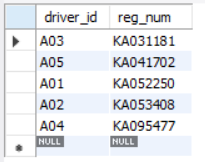
("A02","KA053408"),

("A03","KA031181"),

("A04","KA095477"),

("A05","KA041702");

select \* from owns;



insert into accident values(11,"2003-01-01","Mysore Road"),

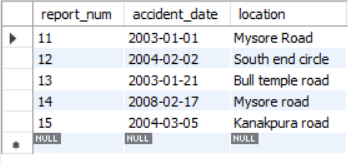
(12,"2004-02-02","South end circle"),

(13,"2003-01-21","Bull temple road"),

(14,"2008-02-17","Mysore road"),

(15,"2004-03-05","Kanakpura road");

select \* from accident;



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

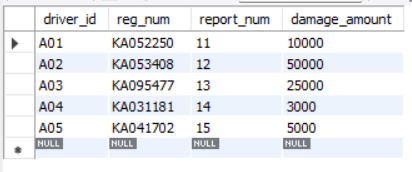
("A02","KA053408",12,50000),

("A03","KA095477",13,25000),

("A04","KA031181",14,3000),

("A05","KA041702",15,5000);

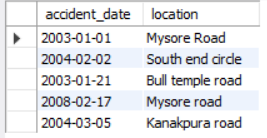
select \* from participated;



**Queries**

**1.Display Accident date and location update participated**

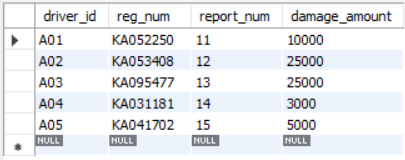
select accident\_date, location from accident;



**2. 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;



**3.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

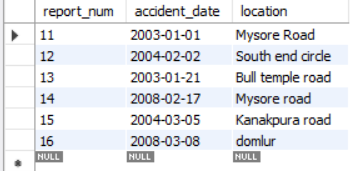
where a.report\_num=b.report\_num and b.accident\_date like'%2008%';



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

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

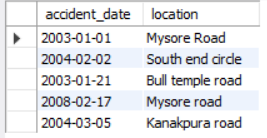
select \* from accident;



**TO DO:**

**1.Display Accident date and location update participated**

select accident\_date, location from accident;

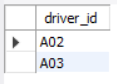


**2.Display driver id who did accident with damage Amount greater than or equal to rs.25000**

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

accident.report\_num = participated.report\_num and participated.damage\_amount >=

25000;



**More Queries on Insurance Database**

**Questions**

**(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 as done in “Program1”week’s lab and Enter at least five tuples for each relation.

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

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

iii. Find the total number of people who owned cars that involved in accidents in 2008.

iv. List the Entire Participated Relation in the Descending Order Of Damage Amount.

v.Find the Average Damage Amount

vi. Delete the Tuple Whose Damage Amount is below the Average Damage Amount

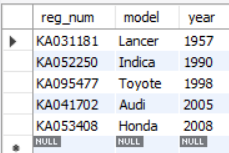
vii. List the Name Of Drivers Whose Damage is Greater than The Average Damage Amount.

viii. Find Maximum Damage Amount.

**Queries**

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

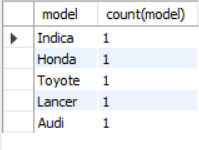
select \* from car order by year asc;



**2. 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\_num =

car.reg\_num group by model;



**3. Find the Average Damage Amount**

select avg(damage\_amount) as average from participated;



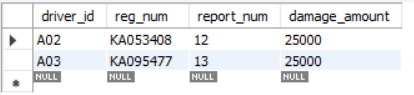
**4. Delete the Tuple Whose Damage Amount is below the Average Damage Amount**

select create temporary table tempavgdamage as

select avg(damage\_amount) as avgdamageamount from participated;

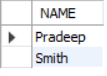
delete from participated

where damage\_amount < (select avgdamageamount from tempavgdamage);



**5. 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);



**6. 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)

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

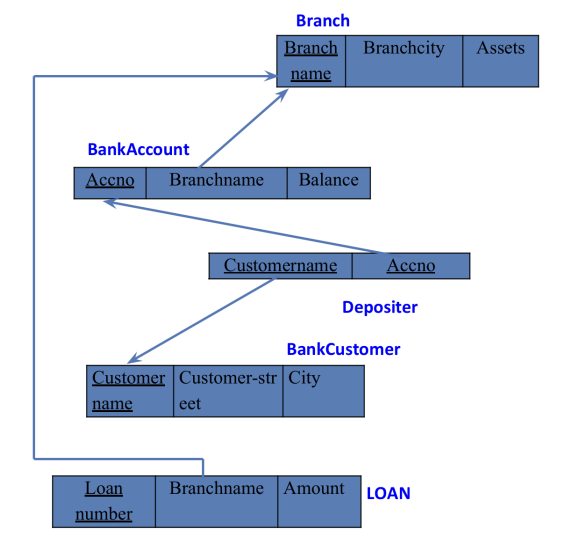
ii. Enter at least five tuples for each relation.

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

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

v. 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\_24becs405;

use bank\_24becs405;

**Create table**

**(Create the above tables by properly specifying the primary keys and the foreign keys)**

create table branch (branch\_name varchar(20), branchcity varchar(20), assets int (10),

primary key(branch\_name));

create table bank\_Account ( accno int (10), branch\_name varchar (20), balance int(20),

primary key(accno),

foreign key(branch\_name)references branch(branch\_name));

create table depositer (customername varchar(20) , accno int(10),

foreign key(customername)references bank\_customer(customername),

foreign key(accno)references bank\_account(accno), primary key (accno,customername));

create table bank\_customer (customername varchar(20) primary key, customer\_street varchar (20),

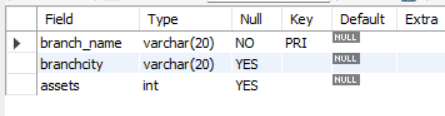
city varchar(20));

create table loan (loan\_number int(10) primary key, branch\_name varchar(20), amount int (10),

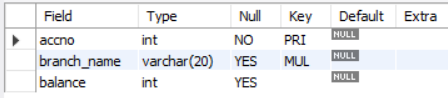
foreign key(branch\_name)references branch(branch\_name));

**Structure of the table**

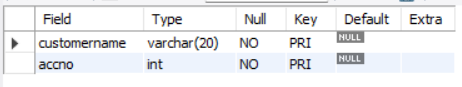
desc branch;



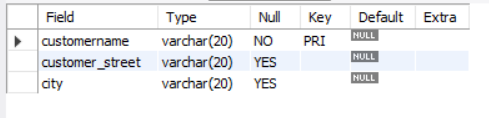
desc bank\_account;



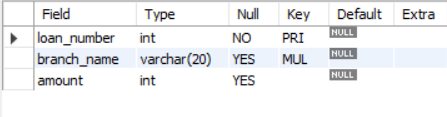
desc depositer;



desc bank\_customer;



desc loan;



**Inserting Values to the table**

**(Enter at least five tuples for each relation)**

insert into branch values ("SBI\_chamrajpet","Bangalore",50000),

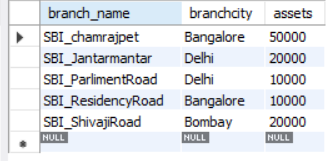
("SBI\_ResidencyRoad","Bangalore",10000),

("SBI\_ShivajiRoad","Bombay",20000),

("SBI\_ParlimentRoad","Delhi",10000),

("SBI\_Jantarmantar","Delhi",20000);

select \* from branch;



insert into bank\_account values

(1,"SBI\_chamrajpet",2000),

(2,"SBI\_ResidencyRoad",5000),

(3,"SBI\_ShivajiRoad",6000),

(4,"SBI\_ParlimentRoad",9000),

(5,"SBI\_Jantarmantar",8000),

(6,"SBI\_ShivajiRoad",4000),

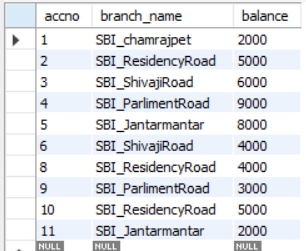
(8,"SBI\_ResidencyRoad",4000),

(9,"SBI\_ParlimentRoad",3000),

(10,"SBI\_ResidencyRoad",5000),

(11,"SBI\_Jantarmantar",2000);

select \* from bank\_account;



insert into bank\_customer values

("Avinash","Bull\_Temple\_Road","Bangalore"),

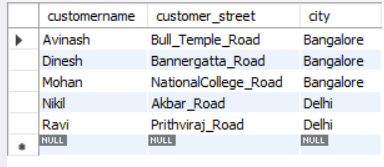
("Dinesh","Bannergatta\_Road","Bangalore"),

("Mohan","NationalCollege\_Road","Bangalore"),

("Nikil","Akbar\_Road","Delhi"),

("Ravi","Prithviraj\_Road","Delhi");

select \* from bank\_customer;;



insert into depositer values

("Avinash",1),

("Dinesh",2),

("Nikil",4),

("Ravi",5),

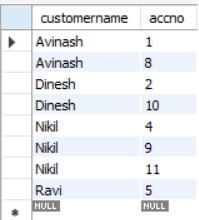
("Avinash",8),

("Nikil",9),

("Dinesh",10),

("Nikil",11);

select \* from depositer;



insert into loan values

(1,"SBI\_chamrajpet",1000),

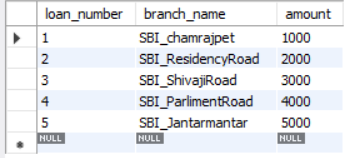
(2,"SBI\_ResidencyRoad",2000),

(3,"SBI\_ShivajiRoad",3000),

(4,"SBI\_ParlimentRoad",4000),

(5,"SBI\_Jantarmantar",5000);

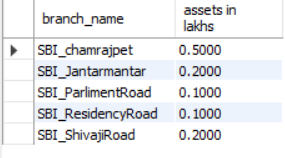
select \* from loan;



**Queries**

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

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



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

select d.customername from depositer d, bank\_account b where

b.branch\_name='SBI\_ResidencyRoad' and d.accno=b.accno group by d.customername having

count(d.accno)>=2;

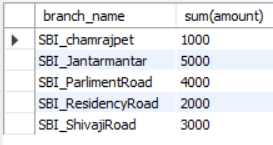
****

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

create view loansum as (

select branch\_name, sum(amount) from loan group by branch\_name);

select \* from loansum;

****

**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)

1. Retrieve all branches and their respective total assets

2. List all customers who live in a particular city

3. List all customers with their account numbers

4. List all customers with their loan amounts

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

6. Find all customers who have accounts with a balance greater than a specified amount (100000)

7. List all customers who have both a loan and an account at the same branch

8. Get the number of accounts held at each branch

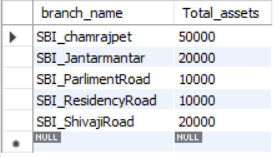
9. Find all branches that have no loans issued

10. Retrieve the branch with the smallest total loan amount

**Queries**

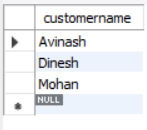
**1. Retrieve all branches and their respective total assets**

select branch\_name , assets as Total\_assets from branch;



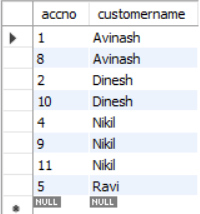
**2. List all customers who live in a particular city**

select customername from bank\_customer where city="Bangalore";



**3. List all customers with their account numbers**

select accno ,customername from depositer;



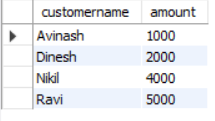
**4. List all customers with their loan amounts**

select c.customername, l.amount

from bank\_customer c

join depositer d on c.customername = d.customername

join loan l on d.accno = l.loan\_number;



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

Select distinct d.customername

from depositer d,bank\_account ba, branch b

where d.accno=ba.accno and ba.branch\_name=b.branch\_name and b.branchcity="Delhi"

group by d.customername having

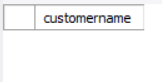
count(b.branch\_name )>1;



**6. Find all customers who have accounts with a balance greater than a specified amount (100000)**

select d.customername from bank\_account b ,depositer d

where b.accno = d.accno and b.balance >100000;

****

**7. List all customers who have both a loan and an account at the same branch**

select distinct c.customername

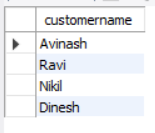
from bank\_customer c

join depositer d on c.customername = d.customername

join bank\_account b on d.accno = b.accno

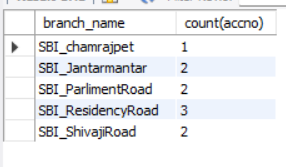
join loan l on b.branch\_name = l.branch\_name

where d.accno = l.loan\_number;



**8. Get the number of accounts held at each branch**

select branch\_name ,count(accno) from bank\_account group by branch\_name ;



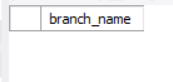
**9. Find all branches that have no loans issued**

select b.branch\_name

from branch b

left join Loan l on b.branch\_name = l.branch\_name

where l.loan\_number is null;



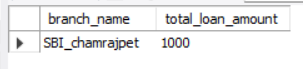
**10. Retrieve the branch with the smallest total loan amount**

select b.branch\_name, sum(l.amount)as total\_loan\_amount from branch b

join loan l on b.branch\_name = l.branch\_name group by b.branch\_name

order by total\_loan\_amount asc

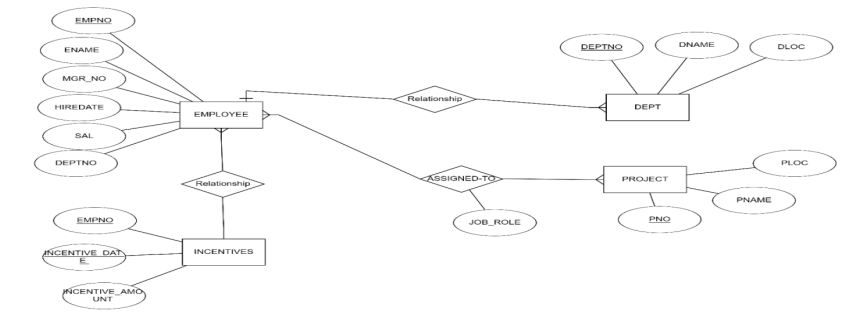
limit 1;



**Employee Database**

**Question**

**(Week 5)**

****

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

ii. Enter greater than five tuples for each table.

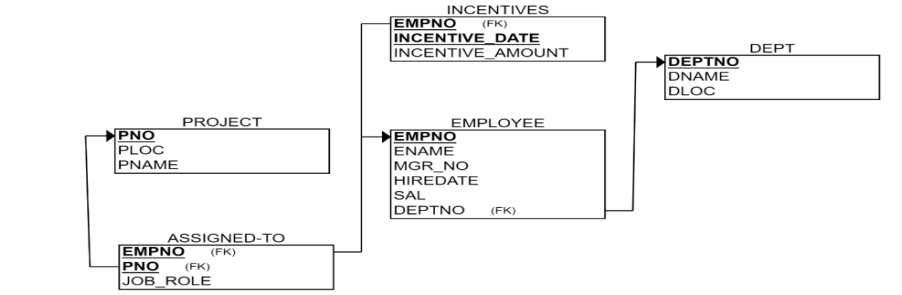
iii. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad,

or Mysuru

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

v. 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 dhanush\_employee;

use dhanush\_employee;

**Create table**

**(Create the above tables by properly specifying the primary keys and the foreign keys)**

create table dept(

deptno int primary key,

dname varchar(20),

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 project(

pno int primary key,

ploc varchar(20),

pname varchar(20));

create table assigned\_to(

empno int , pno int ,

jobrole varchar(20),

primary key(pno,empno),

foreign key(empno) references employee(empno),

foreign key(pno) references project(pno));

create table incentives(

empno int ,

incentive\_date date ,

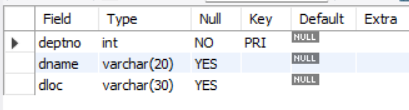
incentive\_amount double ,

primary key(empno,incentive\_date),

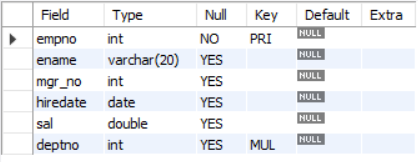
foreign key(empno) references employee(empno));

**Structure of the table**

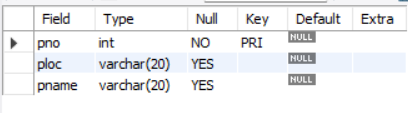
desc dept;



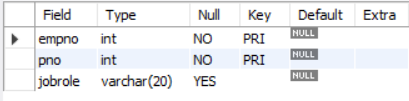
desc employee;



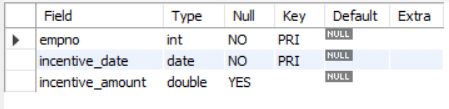
desc project;



desc assigned\_to;



desc incentives;



**Inserting Values to the table**

**(Enter at least five tuples for each relation)**

insert into dept values

(1,"CSE","Bengaluru"),

(2,"ISE","Mysuru"),

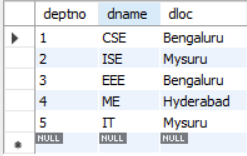
(3,"EEE","Bengaluru"),

(4,"ME","Hyderabad"),

(5,"IT","Mysuru");

select \* from branch;

Select \* from dept;



insert into employee values

(1,"Dhanush",5,"2006-02-24",100000,1),

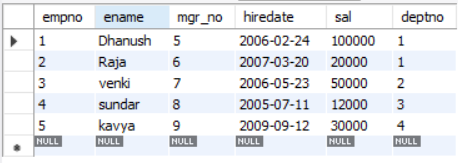
(2,"Raja",6,"2007-03-20",20000,1),

(3,"venki",7,"2006-05-23",50000,2),

(4,"sundar",8,"2005-07-11",12000,3),

(5,"kavya",9,"2009-09-12",30000,4);

Select \* from employee;



insert into project values

(001,"Bengaluru","Expence tracker"),

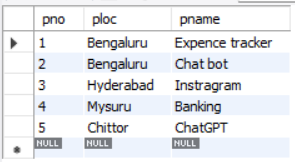
(002,"Bengaluru","Chat bot"),

(003,"Hyderabad","Instragram"),

(004,"Mysuru","Banking"),

(005,"Chittor","ChatGPT");

Select \* from project;



insert into assigned\_to values

(1,001,"HR"),

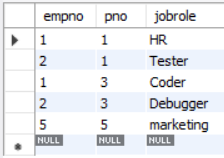
(1,003,"Coder"),

(2,001,"Tester"),

(2,003,"Debugger"),

(5,005,"marketing");

Select \* from assigned\_to;



insert into incentives values

(1,"2006-02-24",20000),

(2,"2006-03-25",100000),

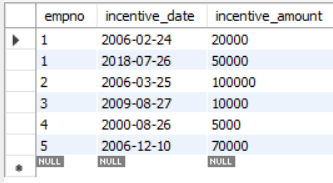
(1,"2018-07-26",50000),

(3,"2009-08-27",10000),

(4,"2000-08-26",5000),

(5,"2006-12-10",70000);

Select \* from incentives;

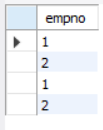


**Queries**

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

select empno from assigned\_to a, project p

where a.pno=p.pno and ploc in ("Bengaluru","Mysuru","Hyderabad");

****

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

select empno from employee where empno not in

(select empno from incentives);

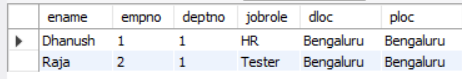


**3. Write a SQL query to find the employees name, number, dept, job\_role, department location and project location who are working for a project location same as his/her department location.**

select e.ename,e.empno,d.deptno,a.jobrole, d.dloc, p.ploc from

employee e, dept d, project p, assigned\_to a

where (e.deptno=d.deptno and a.empno=e.empno and p.pno=a.pno and p.ploc=d.dloc);



**More Queries on Employee Database**

**Question**

**(Week 6)**

1. List all employees along with their project details (if assigned)
2. Find all employees who received incentives, along with the total incentive amount
3. Retrieve the project names and locations of projects with employees assigned as 'Manager'
4. List departments along with the number of employees in each department
5. Find employees who have not been assigned to any project
6. List all employees along with their department names and location
7. Retrieve the details of employees who work under a specific manager (e.g., manager with empno = 101)
8. List all projects that have employees assigned and the number of employees on each project:
9. Find employees with the same manager and list their department details
10. List the total number of incentives given to each employee and the sum of incentives for each:
11. Retrieve all employees who have the role of 'Developer' on any project:
12. Display the department-wise average salary of employees:

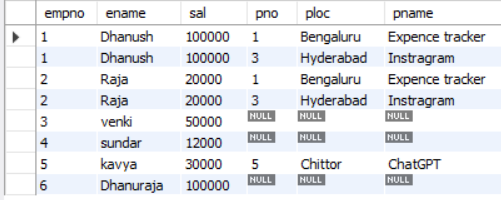
**Queries**

**1. List all employees along with their project details (if assigned)**

select e.empno, e.ename,e.sal,p.pno,p.ploc,p.pname from employee e

left join assigned\_to a on a.empno=e.empno

left join project p on a.pno=p.pno;



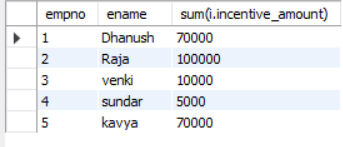
**2. Find all employees who received incentives, along with the total incentive amount**

select e.empno, e.ename ,sum(i.incentive\_amount)

from employee e

join incentives i on e.empno=i.empno

group by empno,e.ename;



**3. Retrieve the project names and locations of projects with employees assigned as 'HR'**

select p.pname,p.ploc from project p

join assigned\_to a on a.pno=p.pno

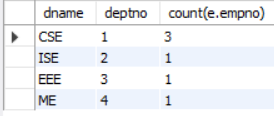
where a.jobrole=”HR";



**4. List departments along with the number of employees in each department**select d.dname,d.deptno ,count(e.empno) from employee e, dept d

where e.deptno=d.deptno

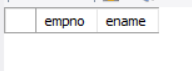
group by d.deptno;

****

**5. Find employees who have not been assigned to any project**

select e.empno ,e.ename from employee e

left join assigned\_to a on e.empno=a.empno where a.pno=Null;

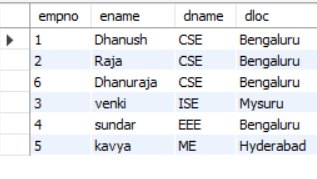
****

**6. List all employees along with their department names and location**

select e.empno,e.ename, d.dname, d.dloc

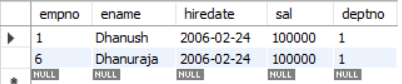
from employee e ,dept d

where e.deptno=d.deptno;

****

**7. Retrieve the details of employees who work under a specific manager (e.g., manager with empno = 5)**

select empno,ename,hiredate,sal,deptno from employee where mgr\_no =5;



**8. List all projects that have employees assigned and the number of employees on each project:**

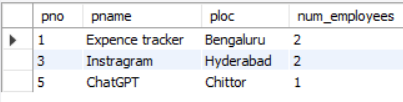
select p.pno, p.pname, p.ploc,

count(a.empno) as num\_employees

from project p

join assigned\_to a on p.pno = a.pno

group by p.pno, p.pname, p.ploc;



**9. Find employees with the same manager and list their department details**

select e1.empno as emp1\_no, e1.ename as emp1\_name,

e2.empno as emp2\_no, e2.ename as emp2\_name,

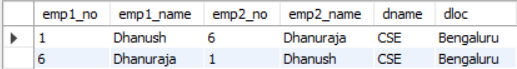
d.dname, d.dloc

from employee e1

join employee e2 on e1.mgr\_no = e2.mgr\_no and e1.empno != e2.empno

join dept d on e1.deptno = d.deptno

order by e1.mgr\_no;



**10 .List the total number of incentives given to each employee and the sum of incentives for each:**

select i.empno, e.ename,

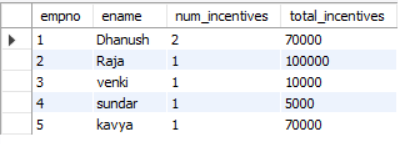
count(i.incentive\_amount) as num\_incentives,

sum(i.incentive\_amount) as total\_incentives

from incentives i

join employee e on i.empno = e.empno

group by i.empno, e.ename;



**11. Retrieve all employees who have the role of 'HR' on any project:**

select e.empno, e.ename, e.sal, e.deptno,

a.jobrole, p.pname, p.ploc from employee e join assigned\_to a on e.empno = a.empno join project p on a.pno = p.pno where a.jobrole = 'hr';

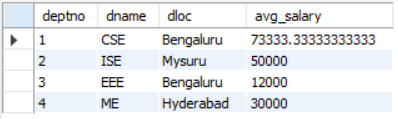


**12. Display the department-wise average salary of employees:**

select d.deptno, d.dname, d.dloc, avg(e.sal) as avg\_salaryfrom dept d

join employee e on d.deptno = e.deptno

group by d.deptno, d.dname, d.dloc;

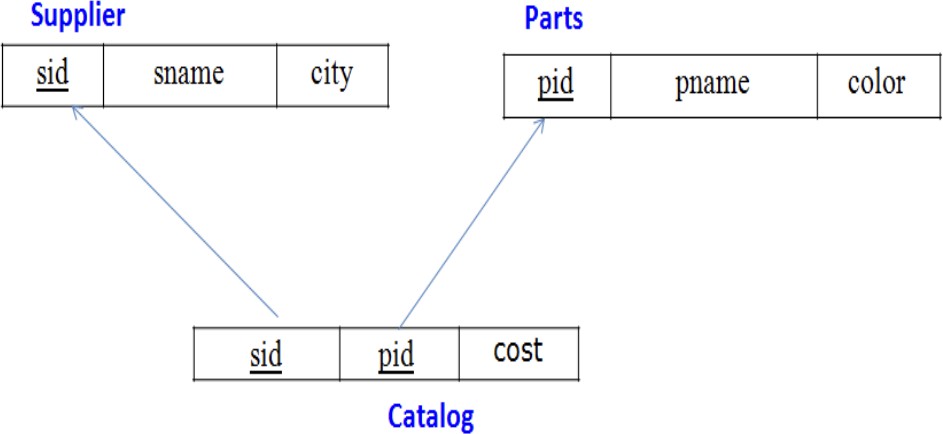


**Supplier Database**

**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 database

create database supply\_;

use supply\_;

Create table

create table supplier\_(

sid int primary key, sname varchar(20),

city varchar(30)

);

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)

);

Inserting Values to the table

insert into supplier\_ values

(10001, "acne", "Bangalore"),

(10002, "johns", "Kolkata"),

(10003, "vimal", "Mumbai"),

(10004, "reliance", "Delhi"); select \* from supplier\_;

insert into parts\_ values

(20001, "Book", "Red"),

(20002, "Pen", "Red"),

(20003, "Pencil", "Green"),

(20004, "Mobile", "Green"),

(20005, "Charger", "Black"); select \* from parts\_;

insert into catalog\_ values

(10001, 20001, 10),

(10001, 20002, 10),

(10001, 20003, 30),

(10001, 20004, 10),

(10001, 20005, 10),

(10002, 20001, 10),

(10002, 20002, 20),

(10003, 20003, 30),

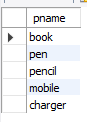
(10004, 20003, 40);

select \* from catalog\_;

**Queries**

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

select pname from parts\_ where pid in (select pid from catalog\_);

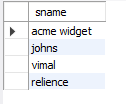


**2) Find the snames of suppliers who supply every part.**

select sname from supplier\_ where sid in

(select sid from catalog\_ group by sid having count(distinct pid) = (select count(distinct

pid) from parts\_));

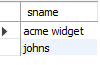


**3) Find the snames of suppliers who supply every red part.**

select distinct sname from supplier\_, parts\_, catalog\_

where supplier\_.sid = catalog\_.sid and parts\_.pid = catalog\_.pid and

parts\_.color="Red";



**4) Find the pnames of parts supplied by Acme Widget Suppliers and by no one else. select pname from parts\_ where pid not in**

(select pid from catalog\_ where sid in (select sid from supplier\_ where sname !=

"acne"));



**5) 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 sid from catalog\_ a where a.cost > (select avg(b.cost) from catalog\_ b where a.pid

= b.pid group by b.pid);

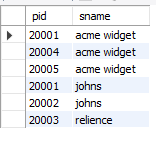


**6) For each part, find the sname of the supplier who charges the most for that part.**

select pid, sname from catalog\_ a, supplier\_

where a.cost = (select max(b.cost) from catalog\_ b where a.pid = b.pid group by b.pid) and

supplier\_.sid = a.sid;



**NO SQL 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**

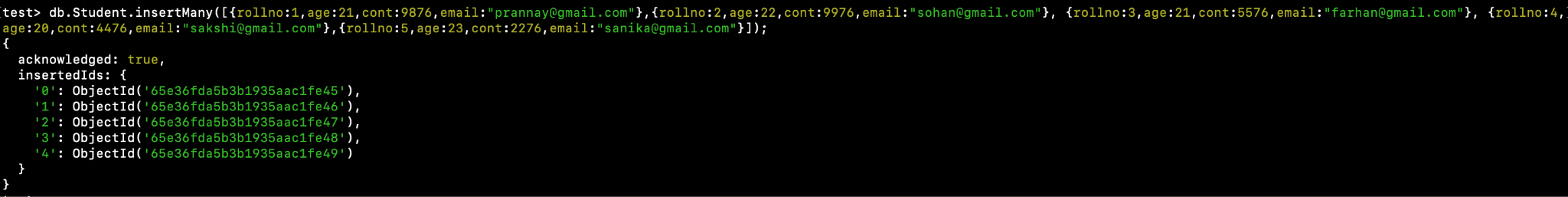
db.Student.insertMany([{**rollno:**1,**age:**21,**cont:**9876,[**email:"prannay@gmail.com"**](mailto:prannay@gmail.com)},

{**rollno:**2,**a ge:**22,**cont:**9976,[**email:"sohan@gmail.com"**](mailto:sohan@gmail.com)},

{**rollno:**3,**age:**21,**cont:**5576,[**email:"farhan@gmail.com"**](mailto:farhan@gmail.com)},

{**rollno:**4,**age:**20,**cont:**4476,[**email:"sakshi@gmail.com"**](mailto:sakshi@gmail.com)},

{**rollno:**5,**age:**23,**cont:**2276,**email:"sa** [**nika@gmail.com"**](mailto:nika@gmail.com)}]);



**Structure of the table**

db.Student.find();



**Queries**

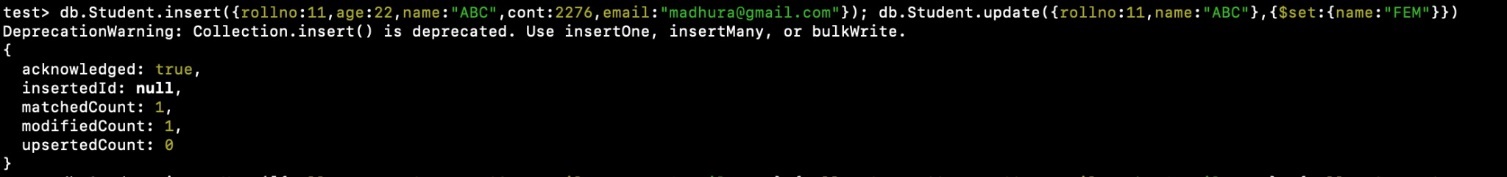
**1) Write a query to update the Email-Id of a student with rollno 5.**

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



**2) 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"**](mailto:madhura@gmail.com)}); **db.Student.update**({**rollno**:11,**name**:**"ABC"**},{$**set**:{**name**:**"FEM"**}})



**3) Export the created table into local file system**

mongoexport mongodb+srv://:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Student -- out C:*\U*sers*\nidhi*y*\D*ocuments*\t*est.Students.json

**4) Drop the table**

db.Student.drop();



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

mongoimport mongodb+srv://:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Student -- type json -file C:*\U*sers*\nidhi\D*ocuments*\t*est.Students.json db.Student.find();

**NoSQL Lab 2**

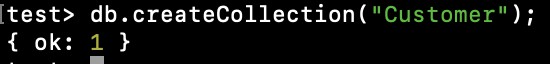
**Question (Week 9)**

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"},**

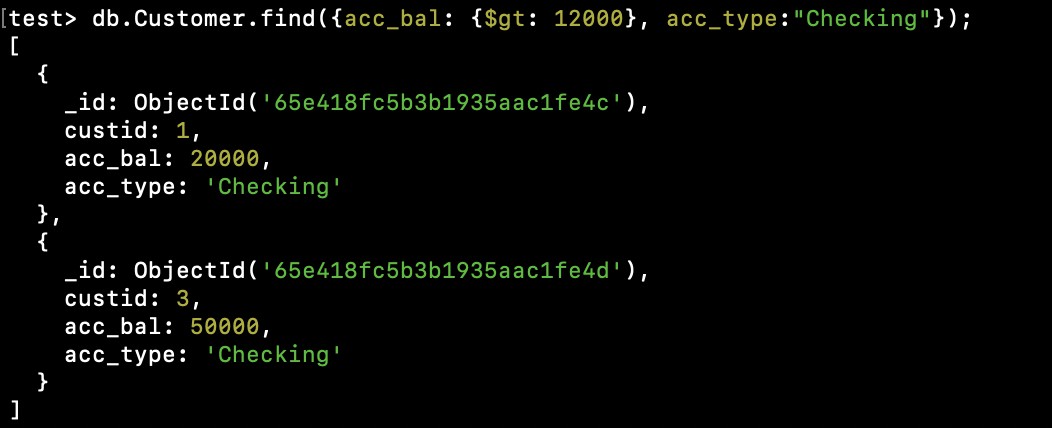
**{custid: 5, acc\_bal:2000, acc\_type: "Checking"}]);**

****

**Queries:**

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

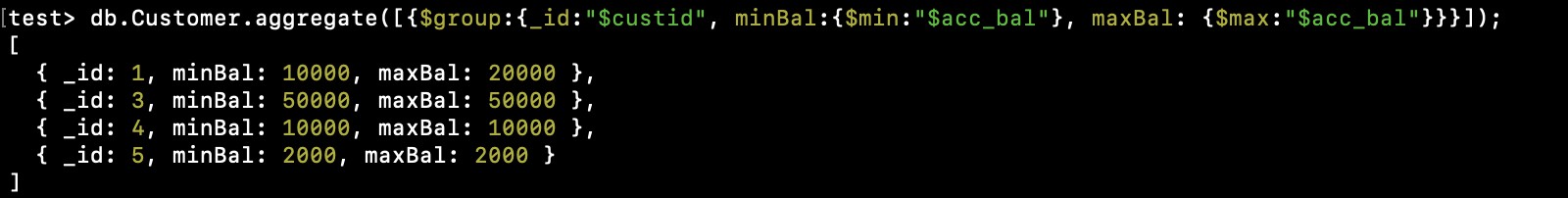
db.Customer.find({acc\_bal: {$gt: 12000}, acc\_type:"Checking"});

****

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

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

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

****

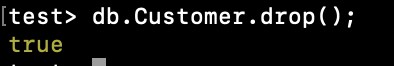
**3) Exporting the collection to a json file**

mongoexport mongodb+srv://:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Customer -- out C:*\U*sers*\nidhi\D*ocuments*\t*est.Customer.json

**4) Dropping collection “Customer”**

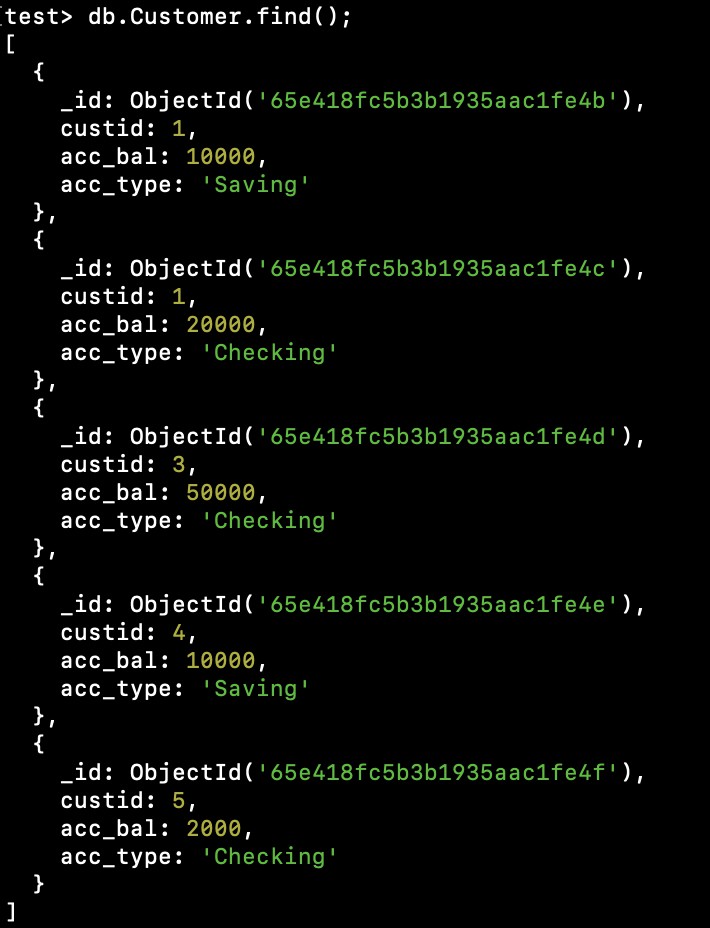
db.Customer.drop();

****

**5) Exporting from a json file to the collection**

mongoimport mongodb+srv://:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Customer -- type json -file C:*\U*sers*\nidhi\D*ocuments*\t*est.Customer.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");



**Inserting Values:**

db.restaurants.insertMany([

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

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

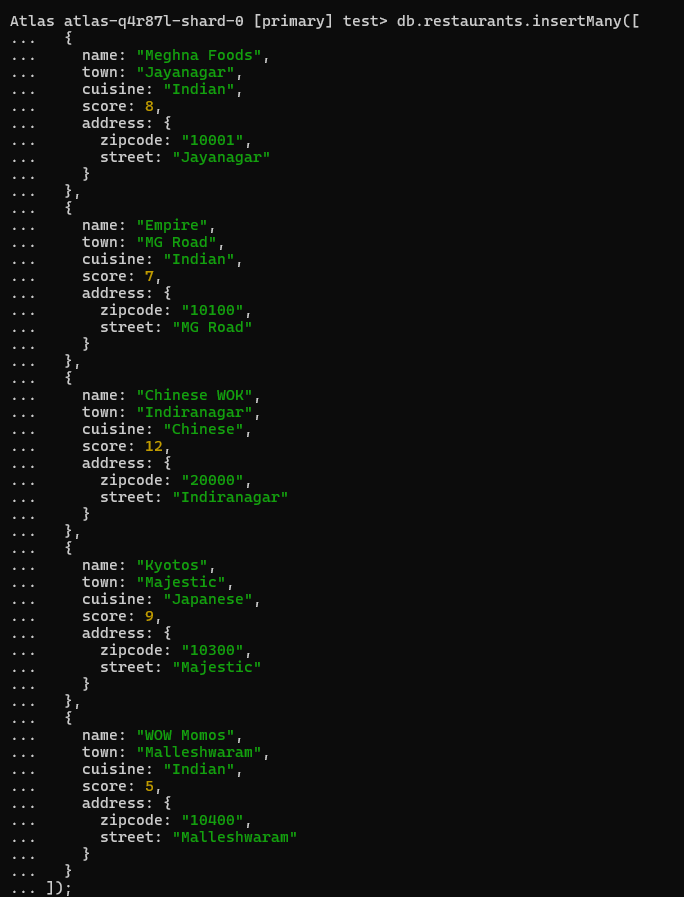
{ name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: { zipcode: "20000", street: "Indiranagar" } },

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

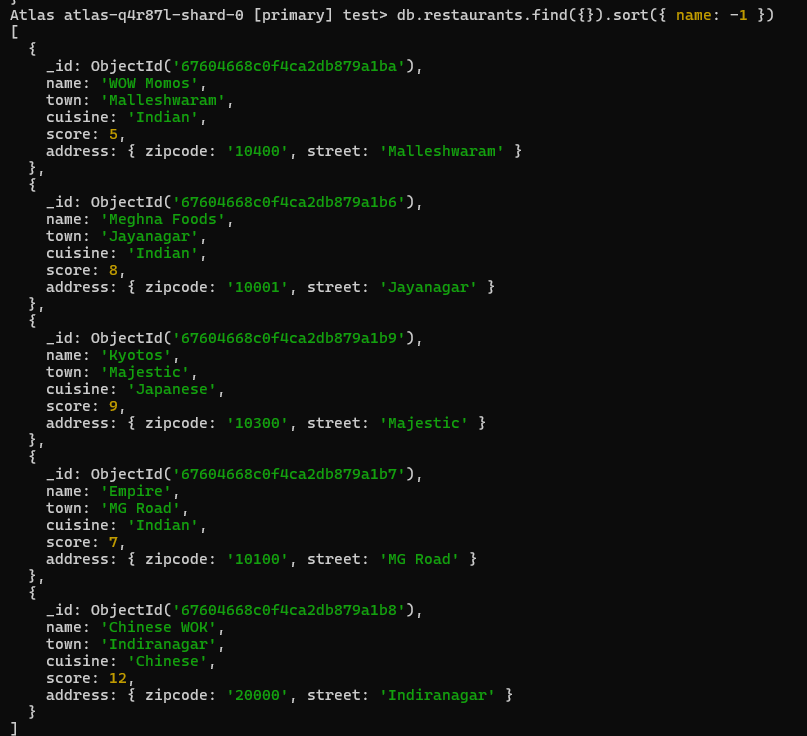
{ name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram" }} ])

**QUERIES**

1) db.Restraunt.find()



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



3) db.Restraunt.find(

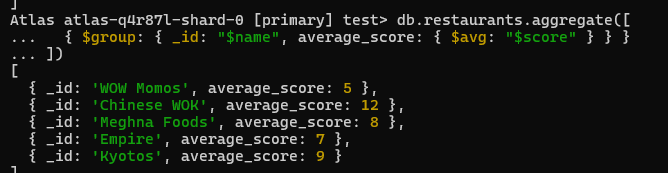
{ "grades.score": { $lte: 10 } },

{ \_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 })

