**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**Database Management Systems (23CS3PCDBM)**

***Submitted by***

**Mahamad Aziz Ansari (1BM23CS174)**

***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 **Mahamad Aziz Ansari (1BM23CS174),** who is a 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. 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.

|  |  |
| --- | --- |
| Rajeshwari Madli  Assistant Professor  Department of CSE, BMSCE | Dr. Joythi S Nayak  Professor & HOD  Department of CSE, BMSCE |

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 9-10-2024 | Insurance Database | 4 |
| 2 | 9-10-2024 | More Queries on Insurance Database | 11 |
| 3 | 16-10-2024 | Bank Database | 14 |
| 4 | 23-10-2024 | More Queries on Bank Database | 21 |
| 5 | 30-10-2024 | Employee Database | 24 |
| 6 | 13-11-2024 | More Queries on Employee Database | 32 |
| 7 | 20-11-2024 | Supplier Database | 36 |
| 8 | 27-11-2024 | NO SQL - Student Database | 42 |
| 9 | 4-12-2024 | NO SQL - Customer Database | 47 |
| 10 | 4-12-2024 | NO SQL – Restaurant Database | 50 |

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.

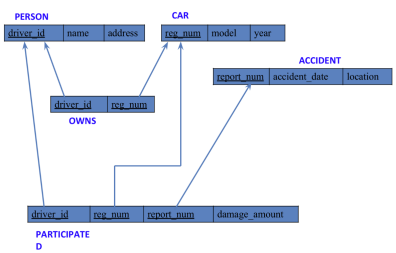
**-** Add a new accident to the database.

**-** 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\_kumud160;

use insurance\_kumud160;

**Create table**

driver\_id varchar(20),

name varchar(30),

address varchar(50),

PRIMARY KEY(driver\_id)

);

create table insurance\_kumud.car(

reg\_num varchar(15),

model varchar(10),

year int,

PRIMARY KEY(reg\_num)

);

create table insurance\_kumud.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\_kumud.accident(

report\_num int,

accident\_date date,

location varchar(50),

PRIMARY KEY(report\_num)

);

create table insurance\_kumud.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),

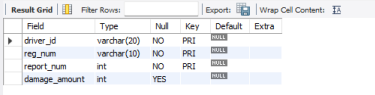
FOREIGN KEY(report\_num) REFERENCES accident(report\_num)

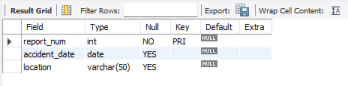
);

);

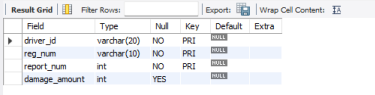
**Structure of the table**

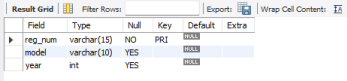
desc person;

desc accident;

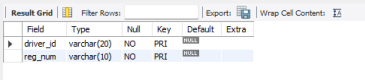


desc participated;

desc car;



desc owns;



**Inserting Values to the table**

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

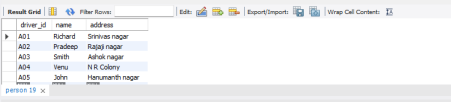
insert into person values("A02","Pradeep", "Rajaji nagar");

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");

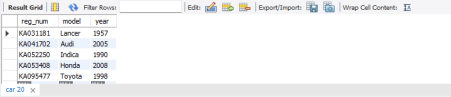
insert into car values("KA031181","Lancer", "1957");

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");

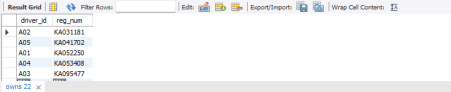
insert into owns values("A02","KA031181");

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");

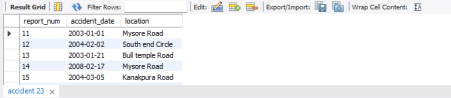
insert into accident values(12,'2004-02-02',"South end Circle");

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);

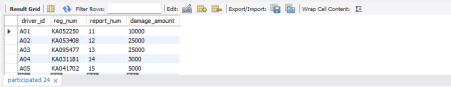
insert into participated values("A02","KA053408",12,50000);

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;



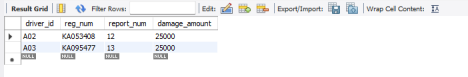
**Queries**

**- 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;

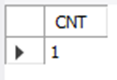


**● Find the total number of people who owned cars that were involved in accidents in 2008.**

select count(distinct driver\_id) CNT

from participated a, accident b

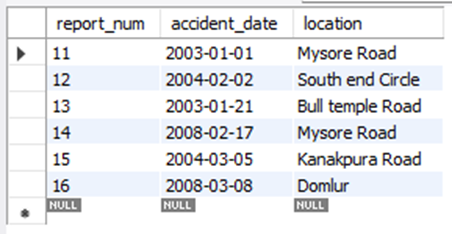
where a.report\_num=b.report\_num and b.accident\_date like '2008%';



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

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

select \* from accident;



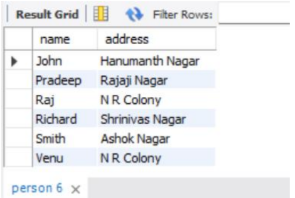
More Queries on Insurance Database

**Questions:**

**(Week 2)**

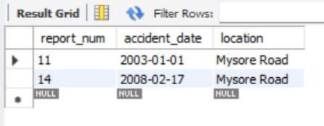
**1. Find drivers who live in a specific city and sort them alphabetically by name.**

select name, address from person order by name;

****

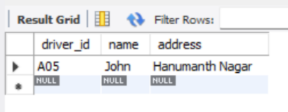
**2. Find all accidents that occurred on 'Mysore Road'**

select \* from accident where location="Mysore Road";

****

**3. Find drivers whose name starts with 'J' and live in an area containing 'nagar'.**

select \* from person where name LIKE "J%" and address LIKE "%Nagar";

****

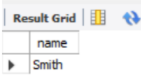
**4. Find drivers who own more than one car.**

select a.name from person a join owns b

on a.driver\_id=b.driver\_id

group by b.driver\_id,a.name

having count(b.driver\_id)>1;

****

**5. Count the number of drivers living in each unique address and list those with more than one**

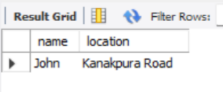
**resident.**

select a.name,c.location

from person a join participated b on a.driver\_id=b.driver\_id

join accident c on b.report\_num=c.report\_num

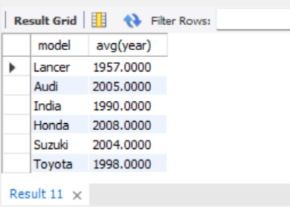
where a.name like 'J%' and c.location like '%Road';

****

**6. List the average year of manufacture for each car model, for models that have more than one**

**car.**

select model, avg(year) from car group by model having avg(year)>1;

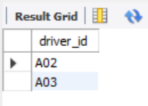
****

**7. Calculate the average damage amount for each driver and list those with an average above**

**20,000.**

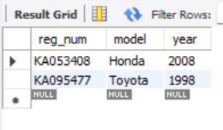
select p1.driver\_id from participated p1 where (select avg(damage\_amount)

from participated p2 where p1.driver\_id=p2.driver\_id)>20000;

****

**8. Retrieve all cars of the model 'Toyota' or 'Honda' manufactured after 1995**

select \* from car where year>1995 and (model="Honda" or model="Toyota");

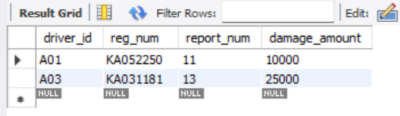
****

**9. Retrieve all records where the damage amount was between 5000 and 30000, and the report**

**number is less than 15.**

select \* from participated where damage\_amount BETWEEN 5000 and 30000

and report\_num<15;

****

Bank Database

**Question**

**(Week 3)**

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

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

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

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

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

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

-Enter at least five tuples for each relation.

- Display the branch name and assets from all branches in lakhs of rupees and rename the

assets column to 'assets in lakhs'.

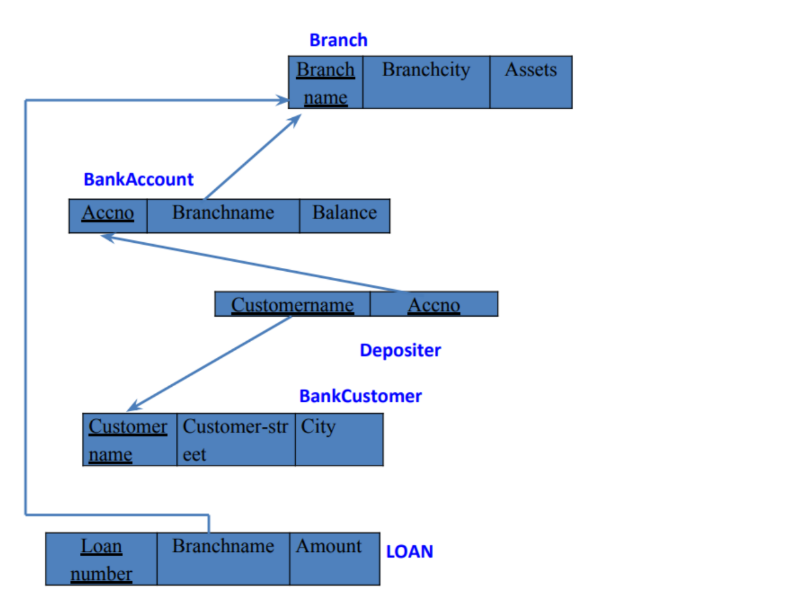
- Find all the customers who have at least two accounts at the same branch (ex.

SBI\_ResidencyRoad).

- Create a view which gives each branch the sum of the amount of all the loans at the

branch.

**Schema Diagram**

****

**Create Database**

create database Bank\_Database\_160;

use Bank\_Database\_160;

**Create table**

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 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),

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

("SBI\_ShivajiRoad", "Bombay", 200000),

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

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

insert into BankAccount

values

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

(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),

(10, "SBI\_ResidencyRoad", 8000),

(11, "SBI\_Jantarmantar", 8000);

insert into Customer

values

("Avinash", "Bull temple road","Bangalore" ),

("Dinesh", "Bannerghatta Road","Bangalore" ),

("Mohan", "NationalCollegeRoad","Bangalore" ),

("Nikhil", "Akbar Road","Delhi" ),

("Ravi", "Prithviraj Road","Delhi" );

insert into Depositer

values

("Avinash", 1),

("Dinesh", 2),

("Mohan", 3),

("Nikhil", 4),

("Ravi", 5),

("Avinash", 8),

("Nikhil", 9),

("Dinesh", 10),

("Nikhil", 11);

insert into Loan

values

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

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

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

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

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

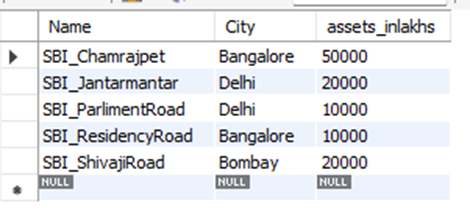
**Queries**

**Q1. 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 assets\_inlakhs real;

select \* from Branch;



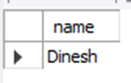
**Q2. Find all the customers who have at least two accounts at the same branch (ex.**

**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;



**Q3. Create a view which gives each branch the sum of the amount of all the loans at the branch.**

create view br

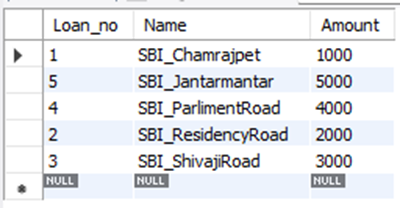
as

select name, sum(amount)

from Loan

group by name;

select \* from br;



**More queries on Bank Database**

**(Week 4)**

**Q1. Find all the customers who have an account at all the branches located in a specific city**

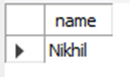
**(Ex. Delhi).**

select distinct d.name

from Depositer d, BankAccount ba, Branch b

where d.accno = ba.accno and ba.Name = b.Name and b.City = "Delhi"

group by d.name having count(b.Name)>1;



**Q2. Find all customers who have a loan at the bank but do not have an account.**

select l.Name

from Loan l

where l.Loan\_no not in(select d.accno from Depositer d

where l.Loan\_no = d.accno);



**Q3. Find all customers who have both an account and a loan at the Bangalore branch**

select b.name

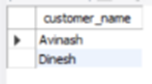
from Loan b

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.Name=b.Name

and b.City="Bangalore");



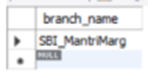
**Q4. Find the names of all branches that have greater assets than all branches located in Bangalore.**

select Name

from Branch

where assets\_inlakhs > all (select assets\_inlakhs from Branch

where City="Bangalore");



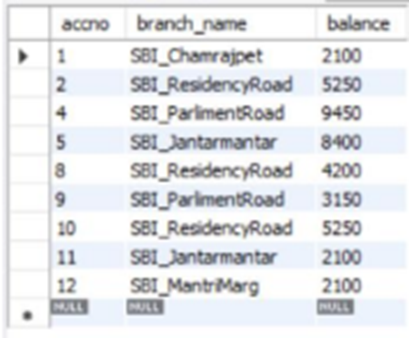
**Q5. Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).**

delete from branchaccount ba

where ba.branch\_name=(select b.branch\_name from branch b where

branch\_city="Bombay");

select \* from branchaccount;

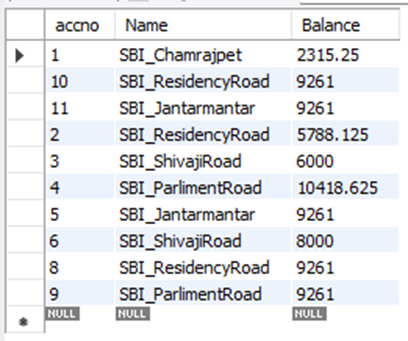


**Q6. Update the Balance of all accounts by 5%**

update BankAccount

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

select \* from BankAccount;



**Employee Database**

**(Week 5)**

**Questions:**

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

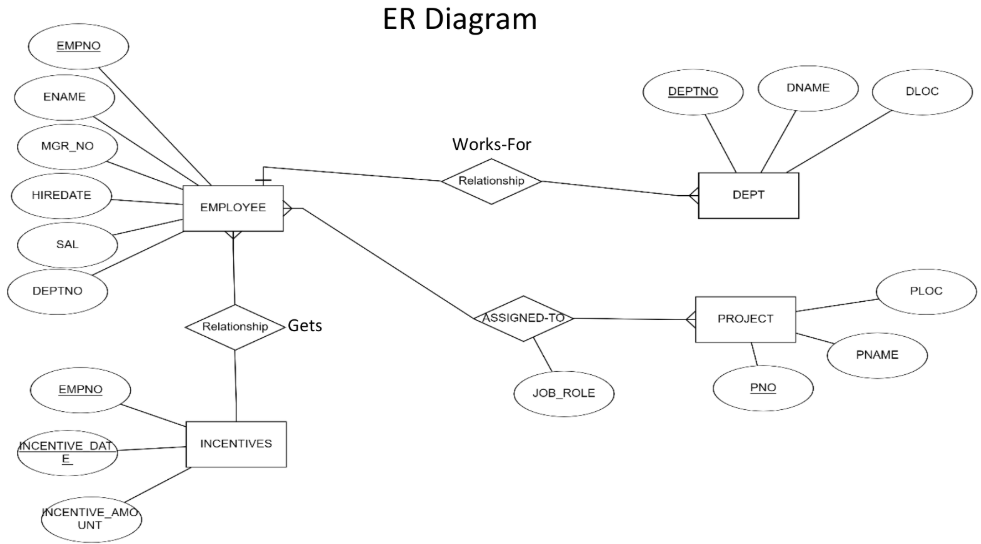
project located in Bengaluru, Hyderabad, or Mysuru

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

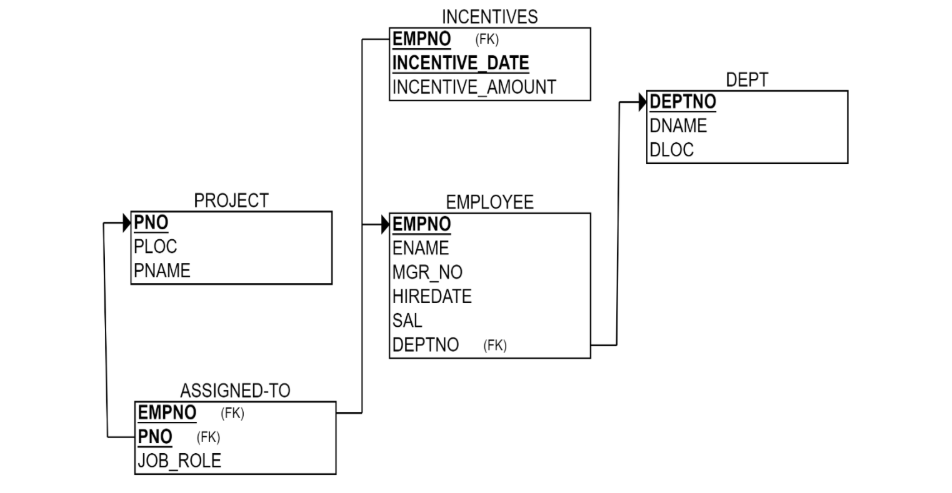
5. Write a SQL query to find the employees name, number, dept,

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\_160;

use employee\_160;

**Create table**

create table Dept(

depno int primary key,

dname varchar(50),

dloc varchar(50)

);

create table Employee(

empno int primary key,

ename varchar (50),

mgrno int,

hiredate date,

salary int,

deptno int,

foreign key (deptno) references Dept (depno)

);

create table Project(

pno int primary key,

ploc varchar(50),

pname varchar(50)

);

create table AssignedTo (

emp\_no int,

pno int,

job\_role varchar(50),

primary key (emp\_no,pno),

foreign key (emp\_no) references Employee(empno),

foreign key (pno) references Project(pno)

);

create table Incentives(

empno int,

incentive\_date date,

incentive\_amount int,

primary key (empno,incentive\_date),

foreign key (empno) references Employee (empno)

);

insert into Dept

values

(1, 'HR', 'New York'),

(2, 'Finance', 'Chicago'),

(3, 'IT', 'Bengaluru'),

(4, 'Sales', 'Hyderabad'),

(5, 'Marketing', 'Mysuru'),

(6, 'Support', 'Los Angeles');

insert into Employee

values

(101, 'Alice', 103, '2021-05-01', 70000, 1),

(102, 'Bob', 101, '2019-07-15', 80000, 2),

(103, 'Charlie', 101, '2020-08-20', 75000, 3),

(104, 'David', 102, '2018-10-12', 90000, 4),

(105, 'Eve', 103, '2023-01-25', 85000, 5),

(106, 'Frank', 104, '2023-03-05', 60000, 6),

(107, 'Grace', 102, '2018-09-25', 78000, 3);

insert into Project

values

(201, 'Delhi', 'A'),

(202, 'Mumbai', 'B'),

(203, 'Bengaluru', 'C'),

(204, 'Hyderabad', 'D'),

(205, 'Mysuru', 'E'),

(206, 'Kathmandu', 'F');

insert into AssignedTo

values

(101, 201, 'Manager'),

(102, 202, 'Analyst'),

(103, 203, 'Developer'),

(104, 204, 'Sales Rep'),

(105, 205, 'Marketer'),

(106, 202, 'Assistant'),

(107, 203, 'Developer');

insert into Incentives

values

(101, '2023-05-15', 5000),

(102, '2023-02-20', 3000),

(104, '2023-05-10', 3500),

(106, '2023-09-23', 6000);

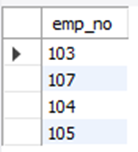
**Queries:**

**Q1. Retrieve the employee numbers of all employees who work on**

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

select a.emp\_no from AssignedTo a, Project p

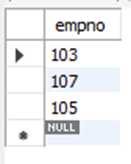
where (a.pno = p.pno) and (p.ploc='Bengaluru' or p.ploc='Hyderabad' or p.ploc='Mysuru');



**Q2. Get Employee ID’s of those employees who didn’t receive incentives**

select empno from Employee

where empno not in (select empno from Incentives);



**Q3. Write a SQL query to find the employees name, number, dept,**

**job\_role, department location and project location who are working for**

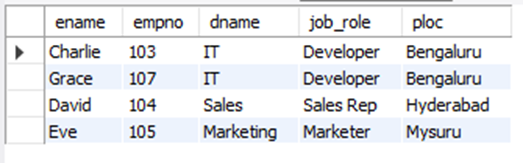
**a project location same as his/her department location.**

select e.ename,e.empno,d.dname,a.job\_role,p.ploc

from Employee e, Dept d, AssignedTo a, Project p

where (p.ploc = d.dloc) and

(e.deptno = d.depno and e.empno = a.emp\_no and a.pno = p.pno);



Additional Queries on Employee Database

**(Week 6)**

**Q1. List the name of the managers with the maximum employees.**

select m.ename manager\_name, m.empno manager\_no, count(e.empno) employee\_count

from Employee e, Employee m

where e.mgrno=m.empno

group by m.empno, m.ename

having count(e.empno)=(

select max(employee\_count)

from(

select count(\*) employee\_count

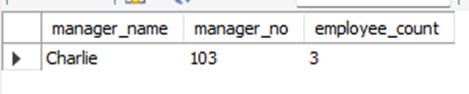
from Employee

where mgrno IS NOT NULL

group by mgrno

)Counts

);



**Q2. Display those managers name whose salary is more than average salary of his**

**employee.**

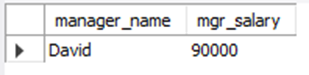
select m.ename manager\_name, m.salary mgr\_salary

from Employee e, Employee m

where e.mgrno = m.empno

group by m.ename, m.salary

having m.salary>avg(e.salary);



**Q3. Find the name of the second top level managers of each department.**

select e.\* from Employee e

join Incentives i on e.empno = i.empno

where i.incentive\_date between '2019-01-01' and '2019-01-31'

and i.incentive\_amount = (

select max(incentive\_amount) from Incentives

where incentive\_date between '2019-01-01' and '2019-01-31'

and incentive\_amount<(

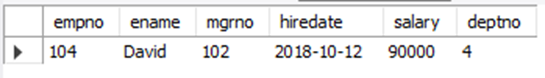
select max(incentive\_amount)

from incentives

where incentive\_date between '2019-01-01' and '2019-01-31'

)

);



**Q4. Find the employee details who got second maximum incentive in January 2019.**

SELECT m.ename

FROM Employee m, Employee e

WHERE e.mgrno = m.empno

GROUP BY m.ename, m.deptno

HAVING COUNT(e.empno) = (

SELECT MAX(counts.emp\_count)

FROM (

SELECT COUNT(e.empno) AS emp\_count

FROM Employee m, Employee e

WHERE e.mgrno = m.empno

GROUP BY m.empno

HAVING COUNT(e.empno) < (

SELECT MAX(emp\_count)

FROM (

SELECT COUNT(e.empno) AS emp\_count

FROM Employee m, Employee e

WHERE e.mgrno = m.empno

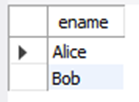
GROUP BY m.empno

) counts

)

) counts

);



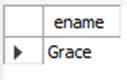
**Q5. Display those employees who are working in the same department where his**

**manager is working.**

select e.ename from Employee e, Employee m

where e.mgrno = m.empno

and e.deptno = m.deptno;



**Supplier Database**

**(Week 7)**

create database supplier\_160;

use supplier\_160;

create table Supplier(

sid int primary key,

sname varchar(50),

city varchar(20)

);

create table Parts(

pid int primary key,

pname varchar(50),

color varchar(20)

);

create table catalog(

sid int,

pid int,

cost int,

primary key(sid,pid),

foreign key(sid) references Supplier(sid),

foreign key(pid) references Parts(pid)

);

insert into Supplier

values

(10001,'Acme Widget','Bangalore'),

(10002,'Johns','Kolkata'),

(10003,'Vimal','Mumbai'),

(10004,'Reliance','Delhi');

insert into Parts

values

(20001,'Book','Red'),

(20002,'Pen','Red'),

(20003,'Pencil','Green'),

(20004,'Mobile','Green'),

(20005,'Charger','Black');

insert into Catalog

values

(10001,20001,10),

(10001,20002,10),

(10001,20003,30),

(10001,20004,10),

(10001,20005,10),

(10002,20001,10),

(10002,20002,20),

(10003,20003,30),

(10004,20003,40);

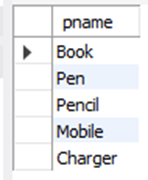
**Queries:**

**q1. Find the pnames of parts for which there is some supplier.**

select distinct p.pname

from Parts p

join Catalog c on p.pid = c.pid;



**q2. Find the snames of suppliers who supply every part.**

select s.sname from Supplier s

where not exists(

select p.pid from Parts p

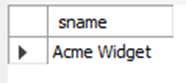
where not exists(

select c.sid from Catalog c

where c.sid = s.sid and c.pid=p.pid

)

);



**q3. 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'

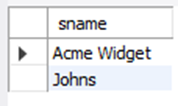
and not exists(

select c.sid from Catalog c

where c.sid=s.sid and c.pid=p.pid

)

);



**q4. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.**

select p.pname from Parts p

join Catalog c on p.pid=c.pid

join Supplier s on c.sid=s.sid

where s.sname = 'Acme Widget'

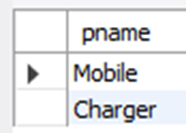
and p.pid not in (

select c1.pid from Catalog c1

join Supplier s1 on c1.sid=s1.sid

where s1.sname!='Acme Widget'

);



**q5. 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

join(

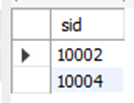
select pid,avg(cost) as avg\_cost

from Catalog

group by pid

)AvgCosts on c.pid=AvgCosts.pid

where c.cost>AvgCosts.avg\_cost;



**q6. For each part, find the sname of the supplier who charges the most for that part.**

select p.pname, s.sname from Parts p

join Catalog c on p.pid=c.pid

join Supplier s on c.sid=s.sid

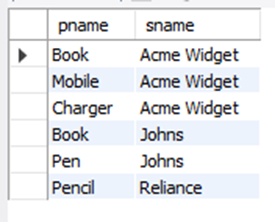
where c.cost=(

select max(c1.cost)

from Catalog c1

where c1.pid=p.pid

);



No SQL Lab 1

(Week 8)

**Questions:**

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.

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](mailto:rekha.de9@gmail.com)"});

db.Student.find()

**Write a query to update the Email-Id of a student with roll no 10.**

db.Student.update({RollNo:10},{$set:

{email:"Abhinav@gmail.com"}})

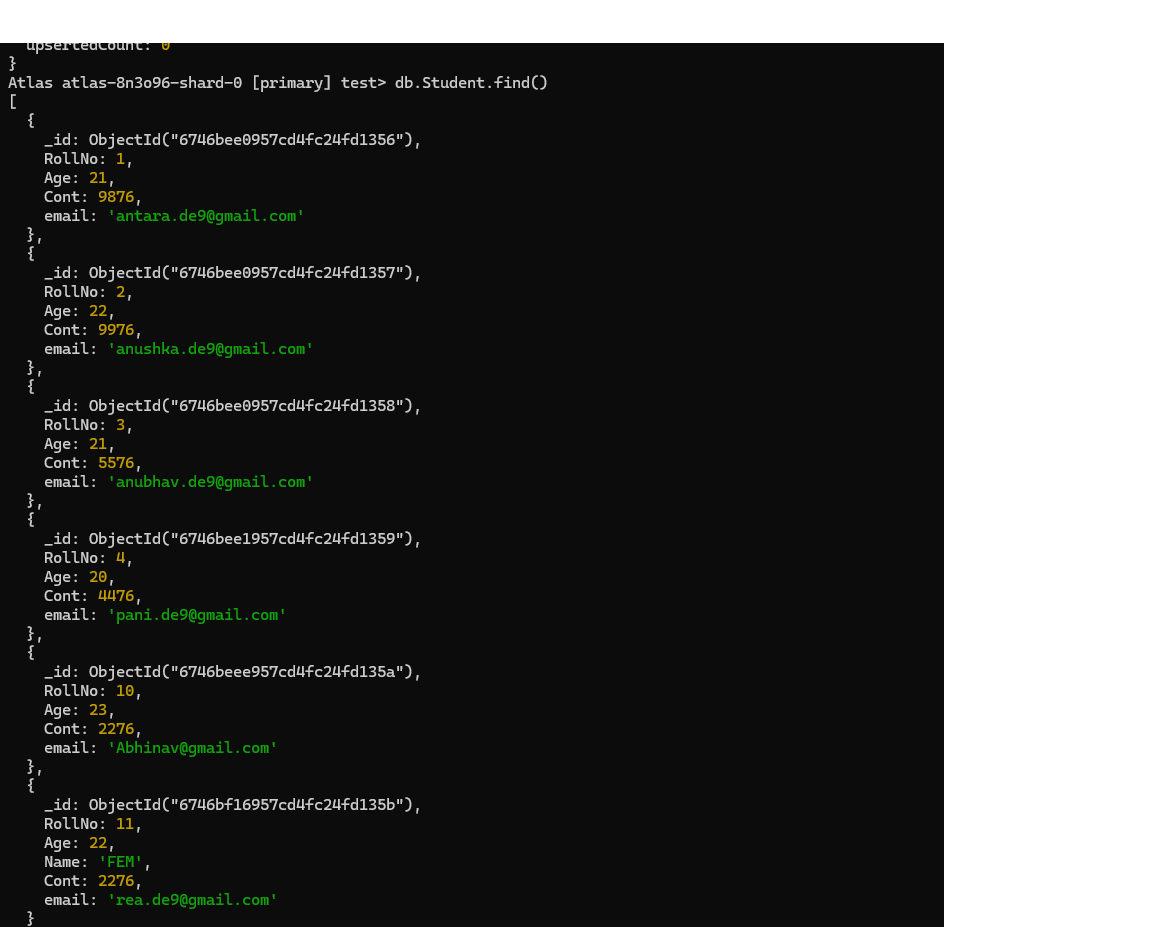
**Replace the student name from “ABC” to “FEM” of roll no 11.**

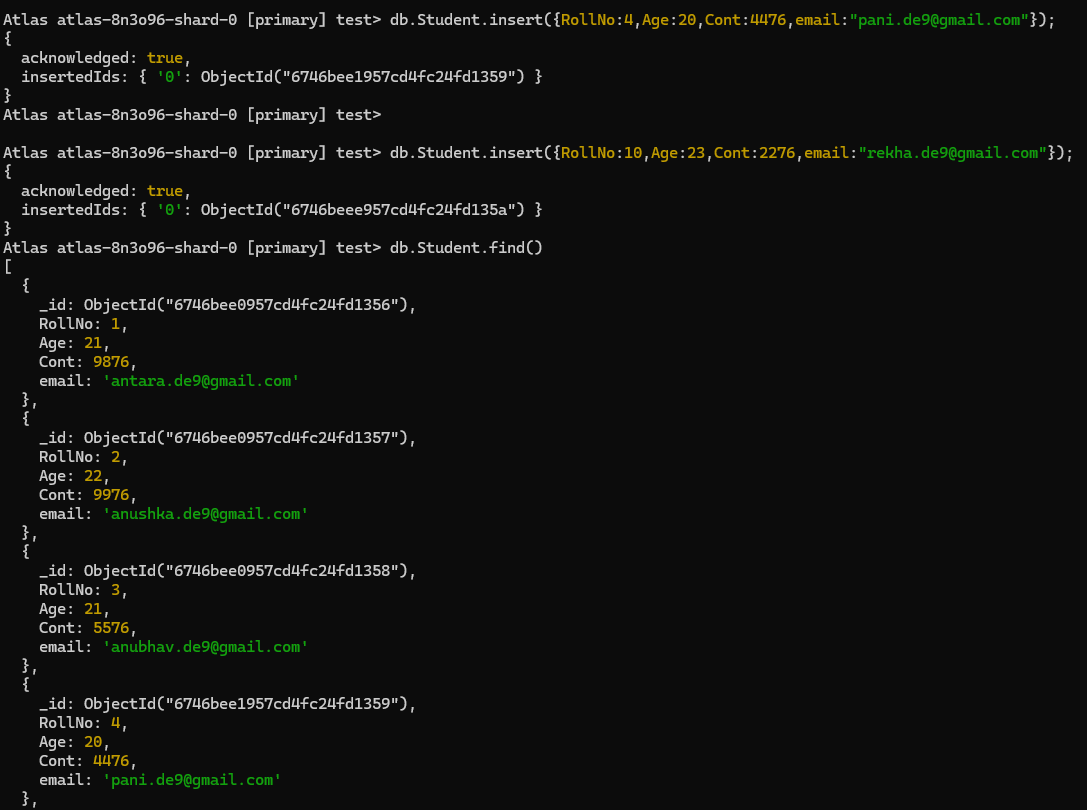
db.Student.insert({RollNo:11,Age:22,Name:

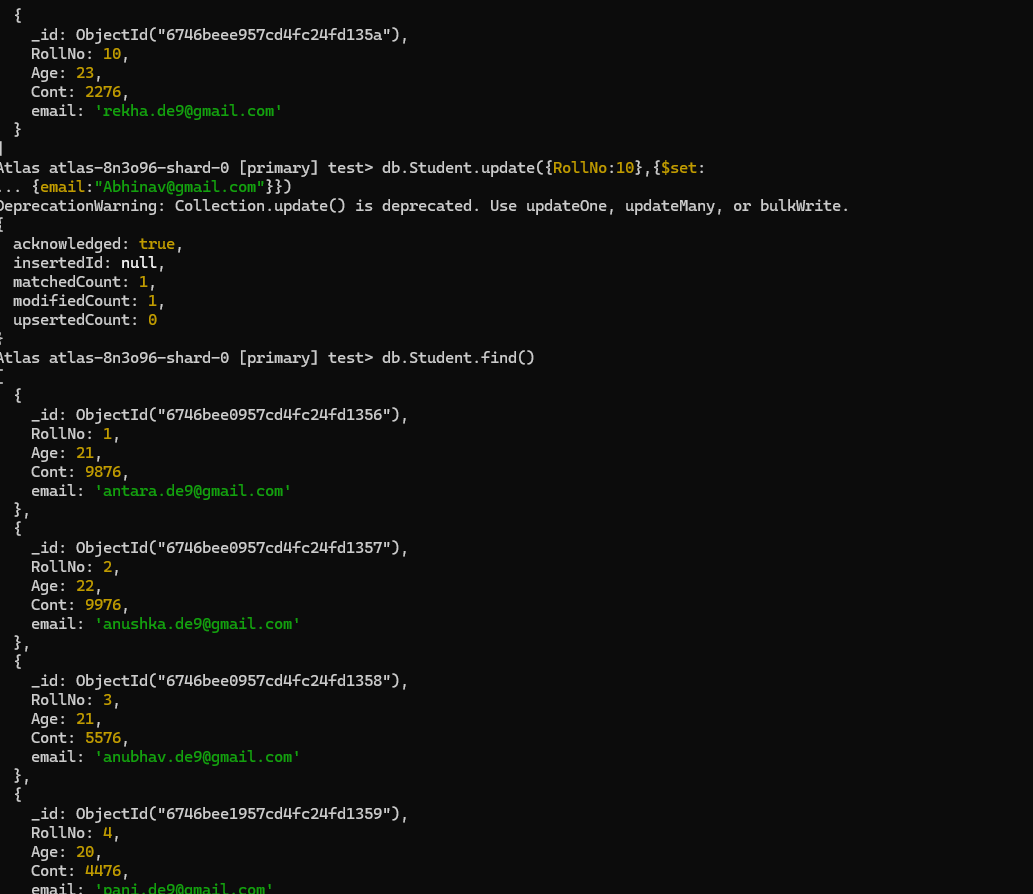
"ABC",Cont:2276,email:"[rea.de9@gmail.com](mailto:rea.de9@gmail.com)"});

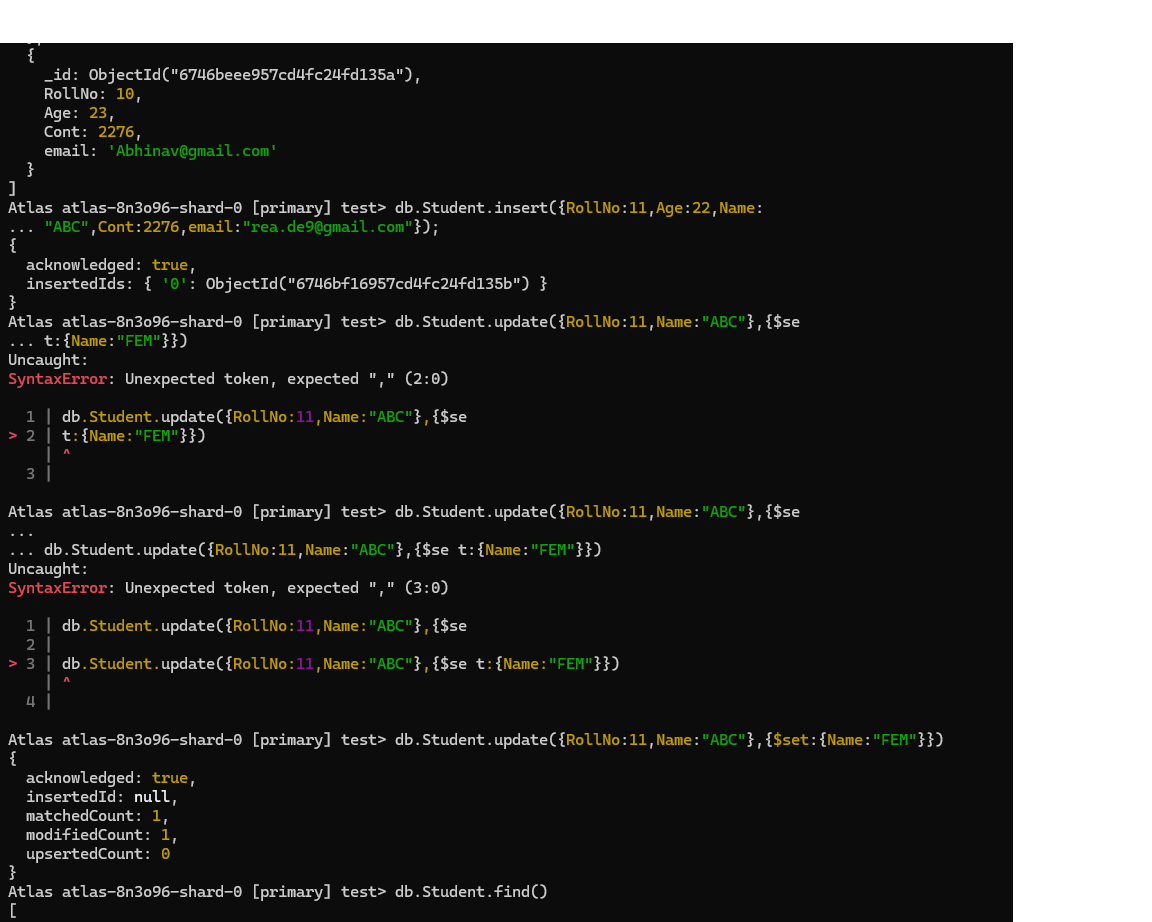
db.Student.update({RollNo:11,Name:"ABC"},{$se

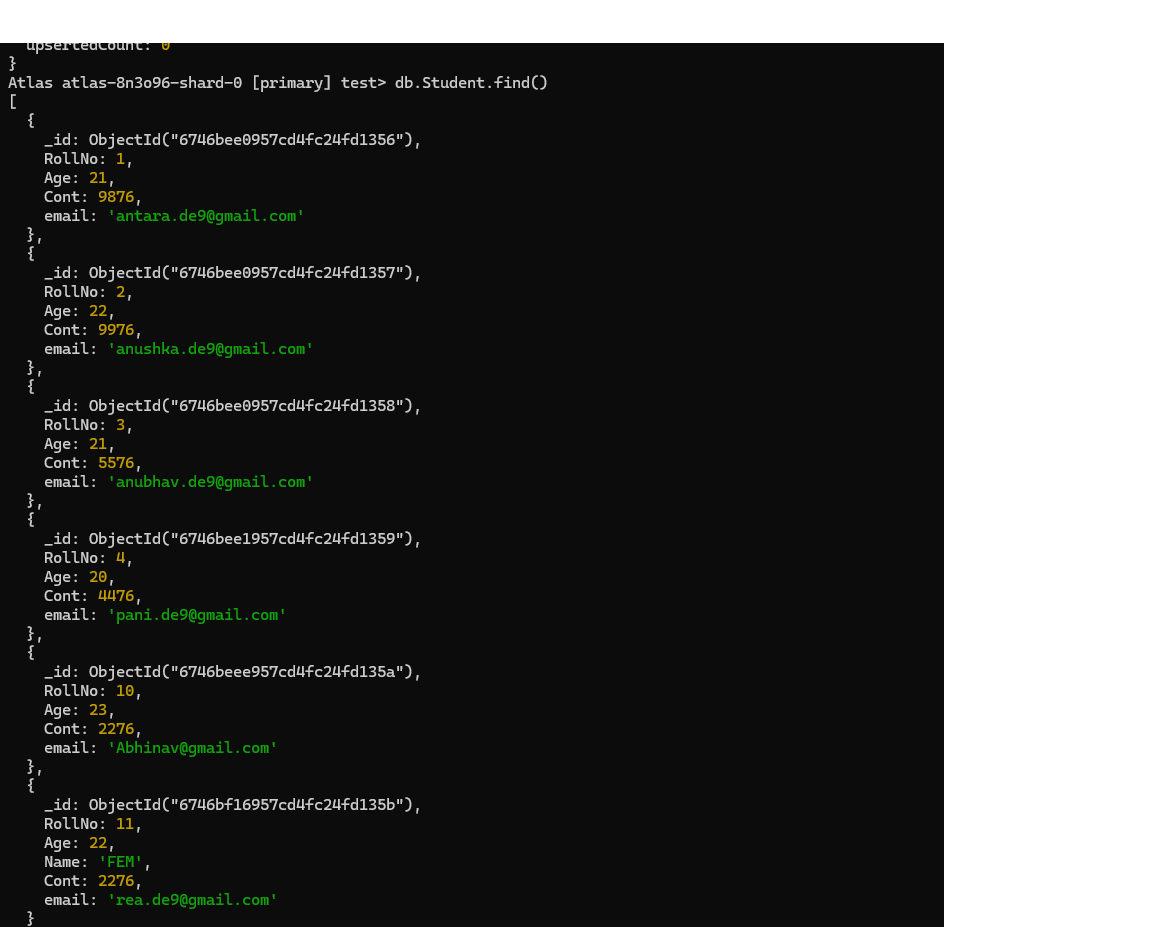
t:{Name:"FEM"}})











No SQL Lab 2

**(Week 9)**

**Questions:**

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"}]);

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"}}}]);

Dropping collection “Customer”

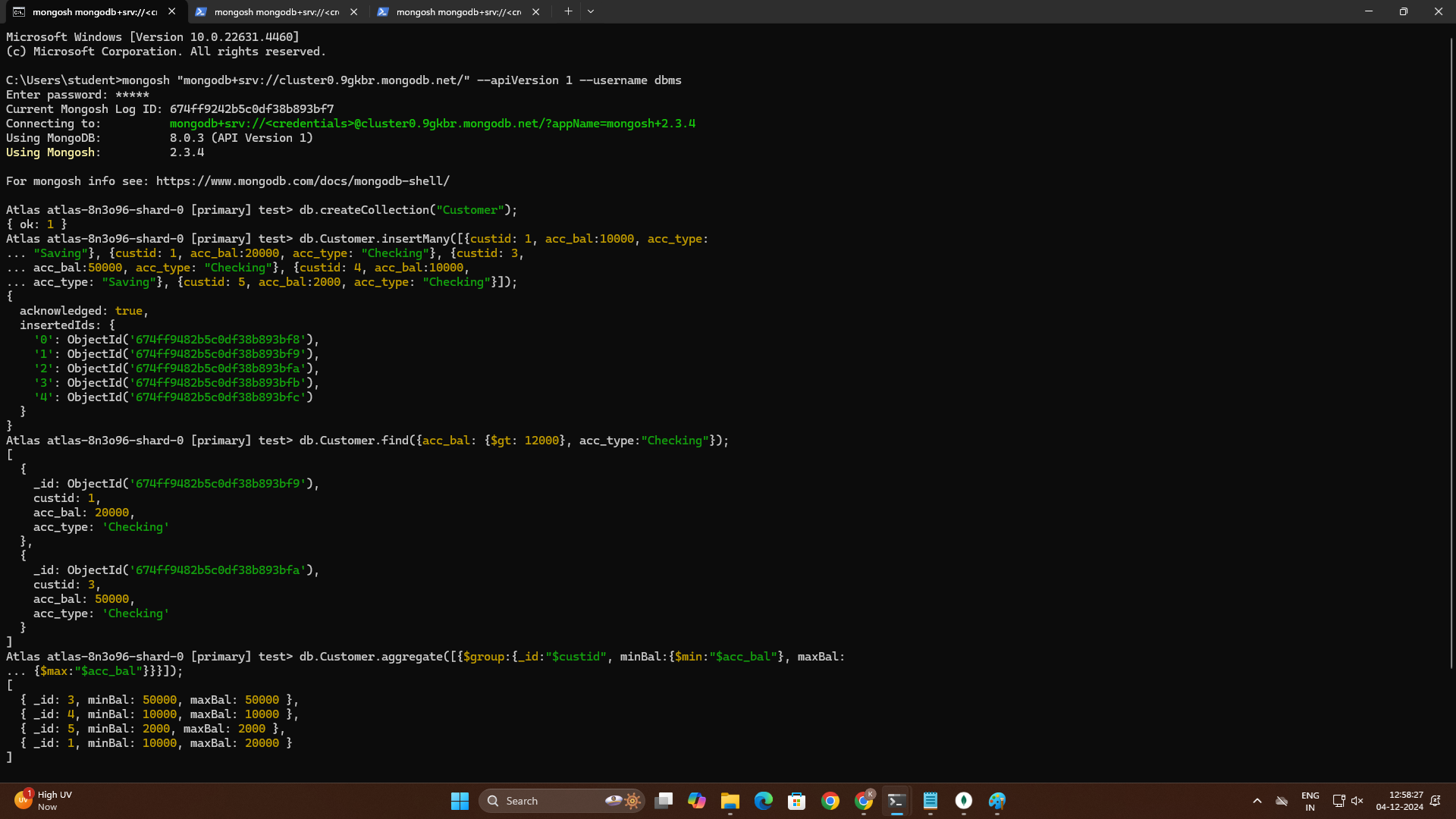
db.Customer.drop();

**Export:**

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

**Import:**

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



NO SQL Lab 3

**(Week 10)**

**Questions:**

Write a MongoDB query to display all the documents in the

collection of restaurants.

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

descending along with all the columns.

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

cuisine for those restaurants which achieved a score which is not

more than 10.

Write a MongoDB query to find the average score for each

restaurant.

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

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

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" }}

])

db.createCollection("restaurants");

db.restaurants.find({})

**Query to arrange the name of the restaurants in descending along with all the**

**columns.**

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

**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 })

**Query to find the average score for each restaurant**

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

**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 })

