# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

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



### LAB REPORT

**on**

Database Management Systems (23CS3PCDBM)

***Submitted by***

**Jayashree Tarai (1BM24CS407)**

***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 **Jayashree Tarai (1BM24CS407),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

| Basavaraj Jakkali Assistant Professor  Department of CSE, BMSCE | Dr. Kavitha Sooda Professor & HOD  Department of CSE, BMSCE |
| --- | --- |

# Index

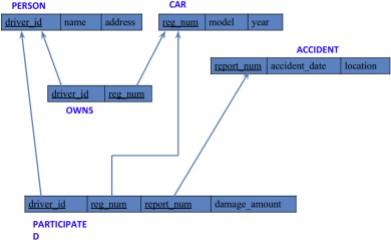
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| --- | --- | --- | --- |
| 1 | 4-10-2024 | Insurance Database | 4-12 |
| 2 | 9-10-2024 | More Queries on Insurance Database | 13-16 |
| 3 | 16-10-2024 | Bank Database | 17-26 |
| 4 | 23-10-2024 | More Queries on Bank Database | 27-29 |
| 5 | 30-10-2024 | Employee Database | 30-36 |
| 6 | 13-11-2024 | More Queries on Employee Database | 37-39 |
| 7 | 20-11-2024 | Supplier Database | 40-46 |
| 8 | 27-11-2024 | NO SQL - Student Database | 47-50 |
| 9 | 4-12-2024 | NO SQL - Customer Database | 51-53 |
| 10 | 4-12-2024 | NO SQL – Restaurant Database | 54-58 |

## Insurance Database

### Question (Week 1)

* PERSON (driver\_id: String, name: String, address: String)
* CAR (reg\_num: String, model: String, year: int)
* ACCIDENT (report\_num: int, accident\_date: date, location: String)
* OWNS (driver\_id: String, reg\_num: String)
* PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)
* Create the above tables by properly specifying the primary keys and the foreign keys. **-** Enter at least five tuples for each relation
* Display Accident date and location
* Update the damage amount to 25000 for the car with a specific reg\_num (example 'K A031181' ) for which the accident report number was 12.
* Add a new accident to the database.
* To Do
* Display Accident date and location
* Display driver\_id who did accident with damage amount greater than or equal to Rs.25000

### Schema Diagram



**Create database**

**create database** insurances\_410;

**use** insurances\_410;

### Create table

**create table** person(

driver\_id varchar(3) **primary key,** name varchar(20) **not null**, address varchar(100)

);

**create table** car(

reg\_no char(8) **primary key,**

model varchar(20), year int(4) **not null**

);

**create table** accident(

report\_no int(4) **primary key,**

accident\_date date, location varchar(100)

);

**create table** owns( driver\_id varchar(3), reg\_no char(8),

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

**foreign key**(reg\_no) **references** car(reg\_no)

);

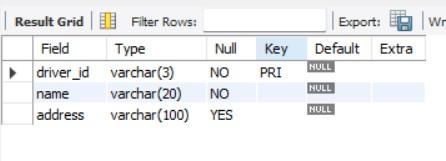
**create table** participated( driver\_id varchar(3), reg\_no char(8),

report\_no int(4), damage\_amt int,

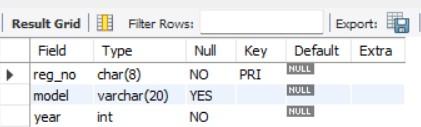
foreign **key**(driver\_id) **references** person(driver\_id), foreign **key**(reg\_no) **references** car(reg\_no),

foreign **key** (report\_no) **references** accident(report\_no)

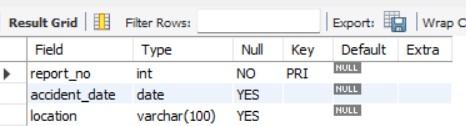
);

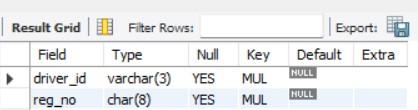
**Person table :**

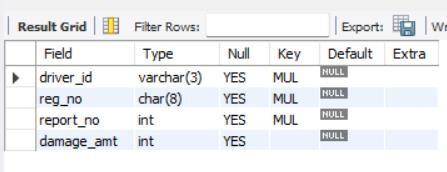
**Car table :**



**Accident table :**



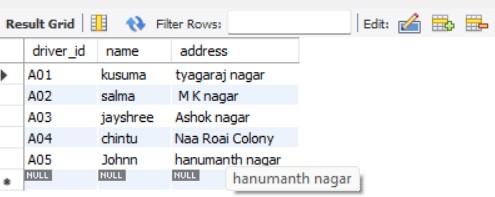
**Owns table :**

**Participated table :**

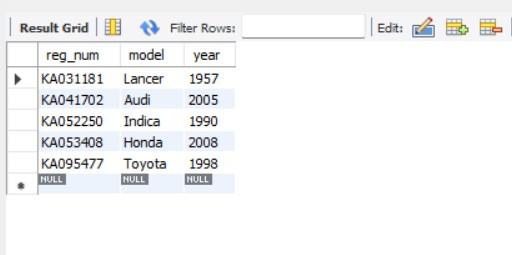
### Inserting Values into the table

insert into person values("A01","kusuma", "tyagaraj nagar"); insert into person values("A02","salma", " M K nagar"); insert into person values("A03","jayshree", "Ashok nagar");

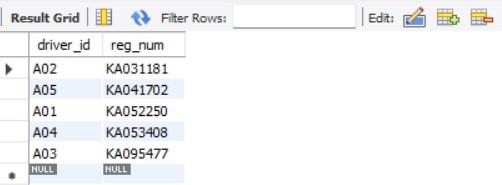
insert into person values("A04","chintu", "Naa Roai Colony"); insert into person values("A05","Johnn", "hanumanth nagar"); **select \*** from person;



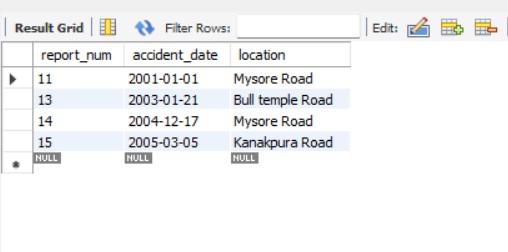
insert into car values("KA052250","Indica", "1990"); insert into car values("KA031181","Lancer", "1957"); insert into car values("KA095477","Toyota", "1998"); insert into car values("KA053408","Honda", "2008"); insert into car values("KA041702","Audi", "2005"); **select \*** from car;



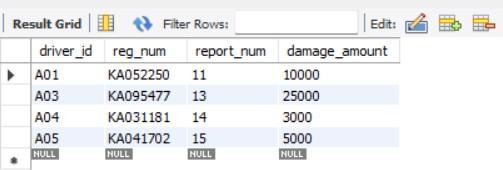
insert into owns values("A01","KA052250"); insert into owns values("A02","KA031181"); insert into owns values("A03","KA095477"); insert into owns values("A04","KA053408"); insert into owns values("A05","KA041702"); **select \*** from owns;



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



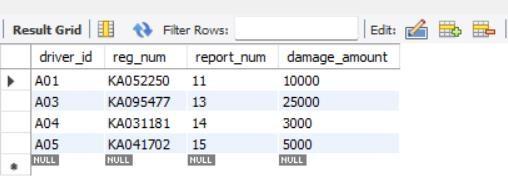
insert into participated values("A01","KA052250",11,10000); insert into participated values("A02","KA053408",12,50000); insert into participated values("A03","KA095477",13,25000); insert into participated values("A04","KA031181",14,3000); insert into participated values("A05","KA041702",15,5000); **select \*** 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; select \* from participated;



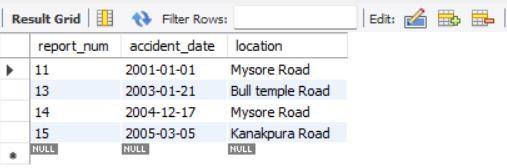
###### Find the total number of people who owned cars that were involved in accidents in 2008.

**select** count(distinct driver\_id) CNT from participated a, accident b **where** a.report\_num=b.report\_num and b.accident\_date like '2001%';



###### Add a new accident to the database.

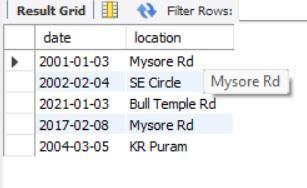
insert into accident values(16,'2003-03-08',"Domlur"); select \* from accident;



**TO DO:**

##### DISPLAY ACCIDENT DATE AND LOCATION

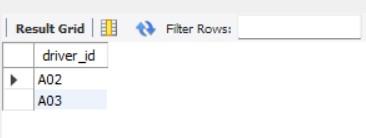
**select** accident\_date **as** date, location **from** accident;



##### DISPLAY DRIVER ID WHO DID ACCIDENT WITH DAMAGE AMOUNT

**GREATER THAN OR EQUAL TO RS.25000**

**Select** participated.driver\_id as driver\_id from accident,participated **where** accident.report\_no = participated.report\_no and participated.damage\_amt >= 25000;

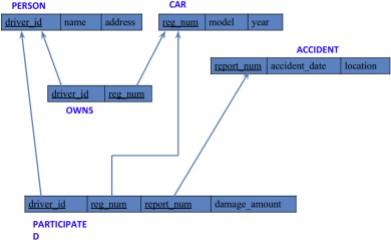


## More Queries on Insurance Database

### 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)
* Create the above tables by properly specifying the primary keys and the foreign keys. **-** Enter at least five tuples for each relation
* Display Accident date and location
* Update the damage amount to 25000 for the car with a specific reg\_num (example 'K A031181' ) for which the accident report number was 12.
* Add a new accident to the database.
* To Do
* Display Accident date and location
* Display driver\_id who did accident with damage amount greater than or equal to Rs.25000

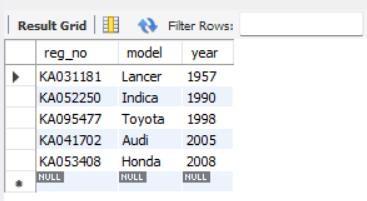
### Schema Diagram



**Queries**

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

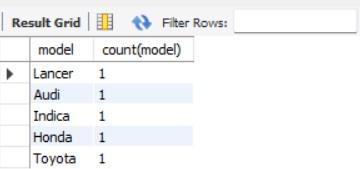
select \* from car **order by** year asc;



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

select model, **count**(**model**) **from** participated, car **where** participated.reg\_no = car.reg\_no

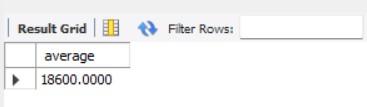
**group by** model;



**TO DO:**

##### FIND THE AVERAGE DAMAGE AMOUNT

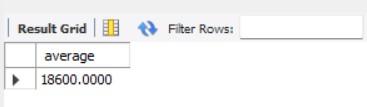
**select avg**(damage\_amout) **as** average **from** participated;



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

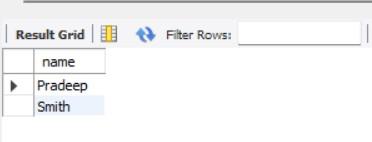
**delete from** participated **where** damage\_amt < (select \* from (select

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



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

**select** name from person, participated **where** person.driver\_id = participated.driver\_id and participated.damage\_amount > (select **avg**(damage\_amount) from participated);



##### FIND MAXIMUM DAMAGE AMOUNT.

**select max**(damage\_amount) **from** participated;

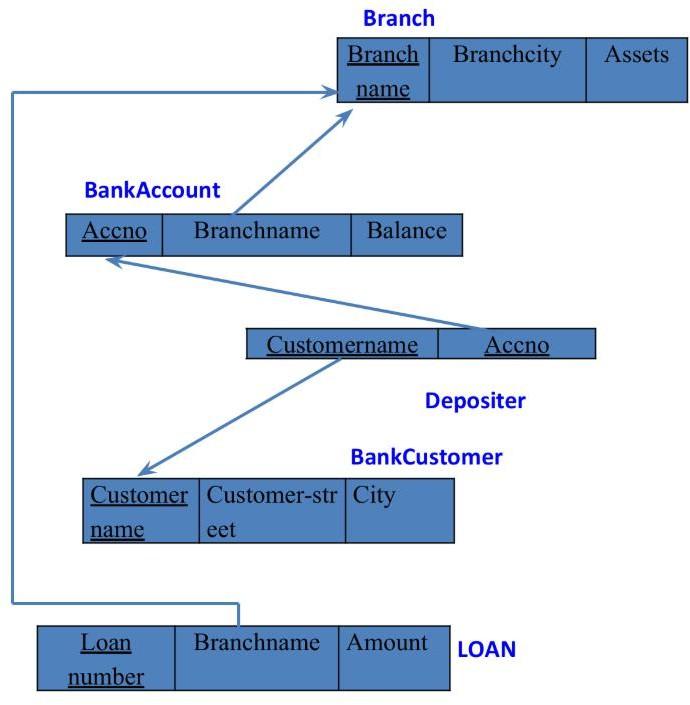


## Bank Database

### 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 database**

**create database** bank\_410;

**use** bank\_410;

### Create table

**create table** Branch( branchname varchar(20), city varchar(20),

assets varchar(20),

**primary key**(branchname)); desc Branch;

**create table** Bankaccount( accno int,

branchname varchar(20), balance varchar(20), **primary key** (accno),

**foreign key**(branchname) **references** Branch(branchname)); desc Bankaccount;

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

desc bankcustomer;

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

**primary key**(customername, accno),

**foreign key**(customername)**references** bankcustomer(customername),

**foreign key**(accno)**references** Bankaccount(accno)); desc depositer;

**create table** loan( loannumber int, branchname varchar(20), amount int,

**primary key**(loannumber),

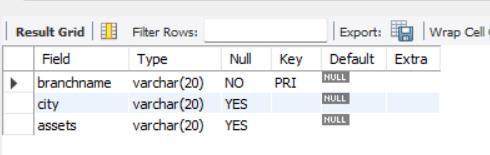
**foreign key**(branchname)**references** Branch(branchname)); desc loan;

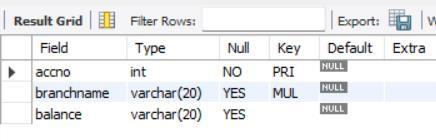
**create table** Borrower( customername varchar(20), loannumber int,

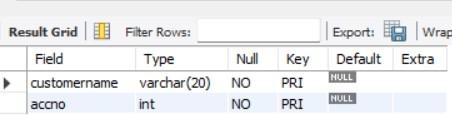
**Primary key**(customername,loannumber),

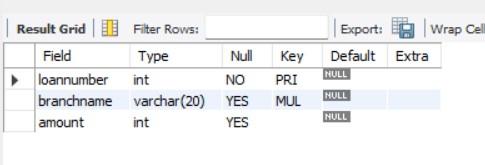
**foreign key**(loannumber)**references** loan\_402(loannumber),

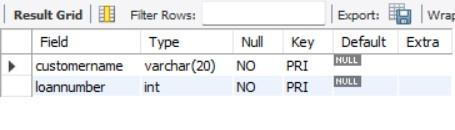
**foreign key**(customername) **references** bankcustomer\_402(customername)); Desc Borrower;







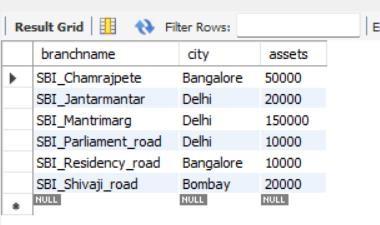




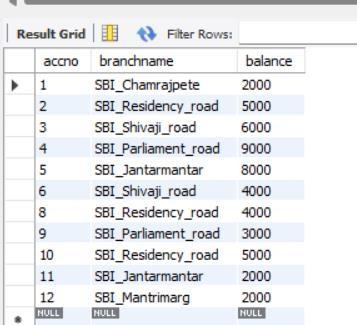
### Inserting the values

insert into Branch values('SBI\_Chamrajpete', 'Bangalore', 50000); insert into Branch values('SBI\_Residency\_road', 'Bangalore',10000); insert into Branch values('SBI\_Shivaji\_road', 'Bombay', 20000); insert into Branch values('SBI\_Parliament\_road','Delhi', 10000); insert into Branch values('SBI\_Jantarmantar', 'Delhi',20000);

insert into Branch values('SBI\_Mantrimarg','Delhi',150000); select \* from Branch;

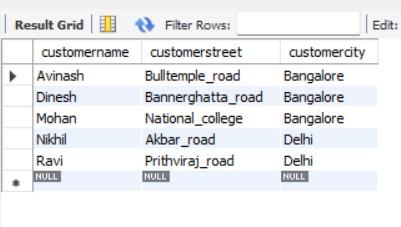


insert into Bankaccount values(1, 'SBI\_Chamrajpete',2000); insert into Bankaccount values(2,'SBI\_Residency\_road', 5000); insert into Bankaccount values(3,'SBI\_Shivaji\_road', 6000); insert into Bankaccount values(4, 'SBI\_Parliament\_road', 9000); insert into Bankaccount values(5, 'SBI\_Jantarmantar', 8000); insert into Bankaccount values(6, 'SBI\_Shivaji\_road', 4000); insert into Bankaccount values(8, 'SBI\_Residency\_road', 4000); insert into Bankaccount values(9, 'SBI\_Parliament\_road', 3000); insert into Bankaccount values(10, 'SBI\_Residency\_road', 5000); insert into Bankaccount values(11, 'SBI\_Jantarmantar', 2000); insert into Bankaccount values(12, 'SBI\_Mantrimarg',2000); select \* from Bankaccount;

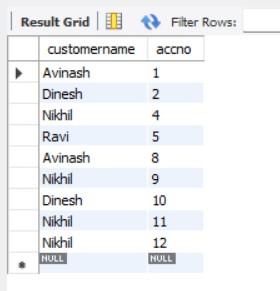


insert into bankcustomer values('Avinash','Bulltemple\_road','Bangalore'); insert into bankcustomer values('Dinesh', 'Bannerghatta\_road','Bangalore'); insert into bankcustomer values('Mohan', 'National\_college','Bangalore'); insert into bankcustomer values('Nikhil', 'Akbar\_road', 'Delhi');

insert into bankcustomer values('Ravi', 'Prithviraj\_road', 'Delhi'); select \* from bankcustomer;

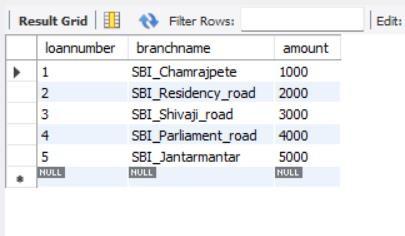


insert into depositer values('Avinash' , 1); insert into depositer values('Dinesh',2); insert into depositer values('Nikhil',4); insert into depositer values('Ravi', 5); insert into depositer values('Avinash',8); insert into depositer values('Nikhil', 9); insert into depositer values('Dinesh',10); insert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

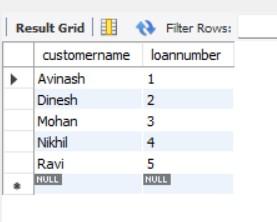


insert into loan values(1, 'SBI\_Chamrajpete',1000); insert into loan values(2, 'SBI\_Residency\_road', 2000); insert into loan values(3, 'SBI\_Shivaji\_road', 3000);

insert into loan values(4, 'SBI\_Parliament\_road', 4000); insert into loan values(5, 'SBI\_Jantarmantar', 5000); select \* from loan;



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



### Queries

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

**alter table** Branch **rename column** assets **to** assets\_in\_lks;

**select** branchname, assets\_in\_lks **from** Branch;



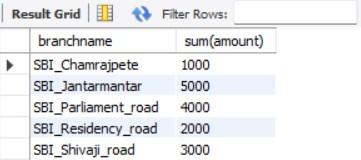
###### Find all the customers who have at least two accounts at the same branch (ex.SBI\_ResidencyRoad).

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

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

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

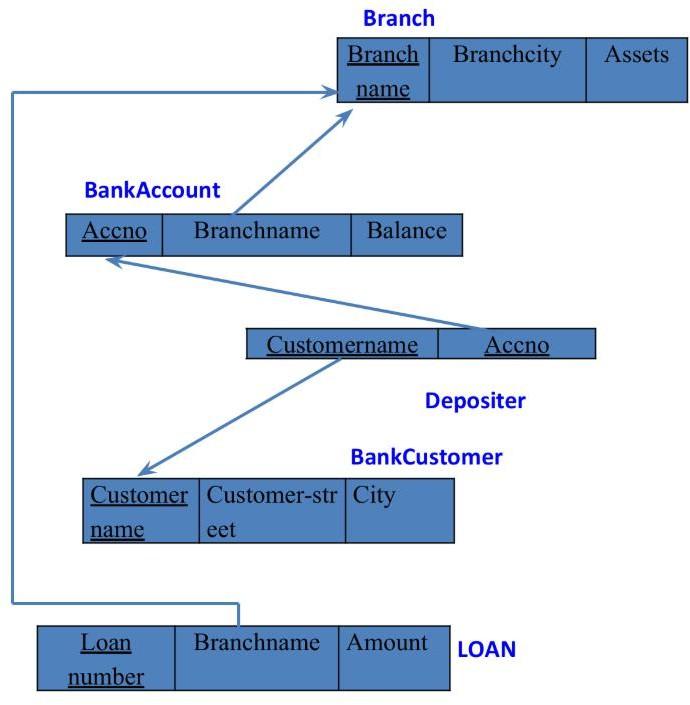
select \* from br;



## 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%



### Queries

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

**select** d.customername **from** depositer d, Branch b, Bankaccount a

**where** b.branchname = a.branchname and a.accno=d.accno and city = 'Delhi'

**group by** d.customername having **count(**distinct b.branchname)=(select count(branchname) from Branch

**where** city = 'Delhi');



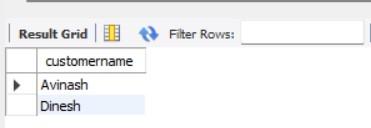
###### Find all customers who have a loan at the bank but do not have an account.

**select distinct** d.customername from depositer d, Bankaccount ba, Branch b where d.accno = ba.accno And ba.branchname=b.branchname and b.city='Delhi' group by d.customername **having count**(distinct b.branchname)>1;



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

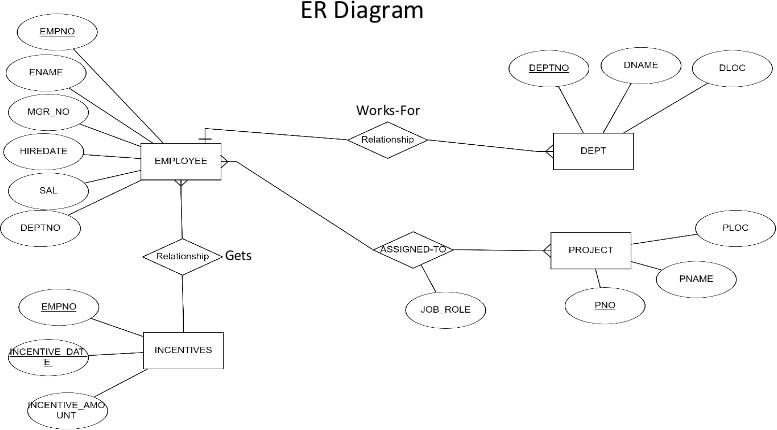
**select** b.customername from Borrower b **where** b.loannumber in(select d.accno **from** depositer d, Bankaccount ba, Branch b **where** b.loannumber = d.accno and d.accno = ba.accno and ba.branchname=b.branchname and b.city='Bangalore');



## Employee Database

### Question (Week 5)

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2. Enter greater than five tuples for each table.
3. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru
4. Get Employee ID’s of those employees who didn’t receive incentives
5. Write a SQL query to find the employees name, number, dept, job\_role, department location and project location who are working for a project location same as his/her department location.



### Create database

create database employee\_410; use employee\_410;

### Create tables

**create table** dept ( deptno int **primary key**, dname varchar(50),

dloc varchar(50)

);

**create table** employee ( empno int **primary key,** ename varchar(50), mgr\_no int,

hiredate date, sal int, deptno int,

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

);

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

);

**create table** assigned\_to ( empno int,

pno int,

job\_role varchar(50),

**primary key** (empno, pno),

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

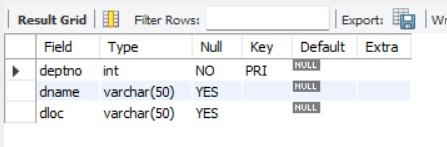
**foreign key** (pno) **references** project(pno)

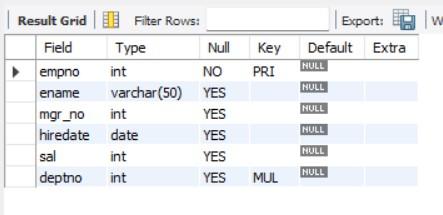
);

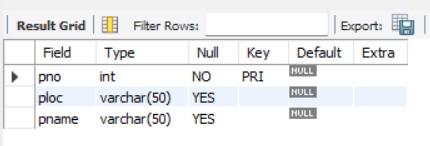
**create table** incentives ( empno int, incentive\_date date, incentive\_amount int,

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

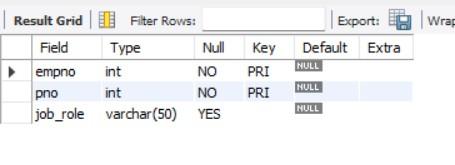
**Department table :**

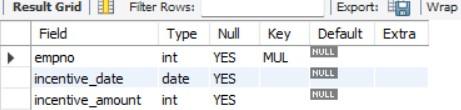


**Employee Table :**

**Project table :**

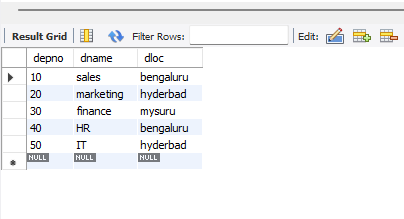
**Assigned\_to table :**



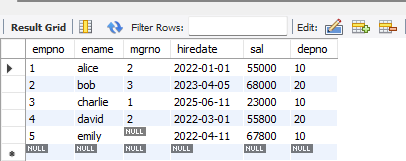
**Incentive table:**

###### Inserting the values to the tables

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

select \* from dept;

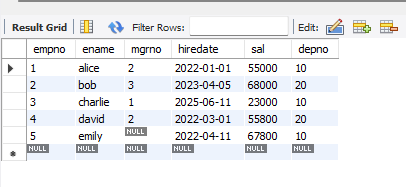
insert into employee values(1,'alice',2,'2022-01-01',55000,10); insert into employee values(2,'bob',3,'2023-04-05',68000,20); insert into employee values(3,'charlie',1,'2025-06-11',23000,10); insert into employee values(4,'david',2,'2022-03-01',55800,20); insert into employee values(5,'emily',null,'2022-04-11',67800,10);

select \* from employee;

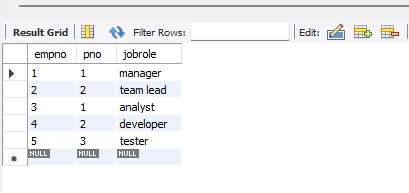
insert into project values(1,'e-learning','bengaluru');

insert into project values(2,'hostel management','hyderbad'); insert into project values(3,'hotel management','bengaluru'); insert into project values(4,'face recognition','chennai');

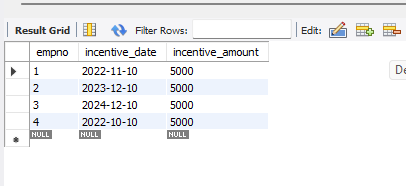
insert into project values(5,'face emotion recognition','mysuru'); select \* from project;



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



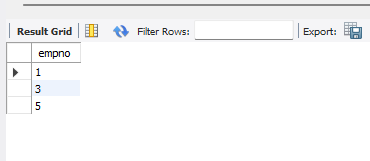
insert into incentives values(1,'2022-11-10',5000); insert into incentives values(2,'2023-12-10',5000); insert into incentives values(3,'2024-12-10',5000); insert into incentives values(4,'2022-10-10',5000); select \* from incentives;



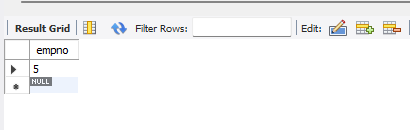
### Queries

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

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

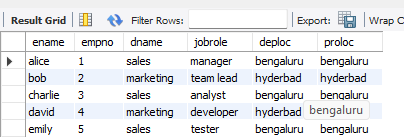


###### Get Employee ID’s of those employees who didn’t receive incentives

**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.

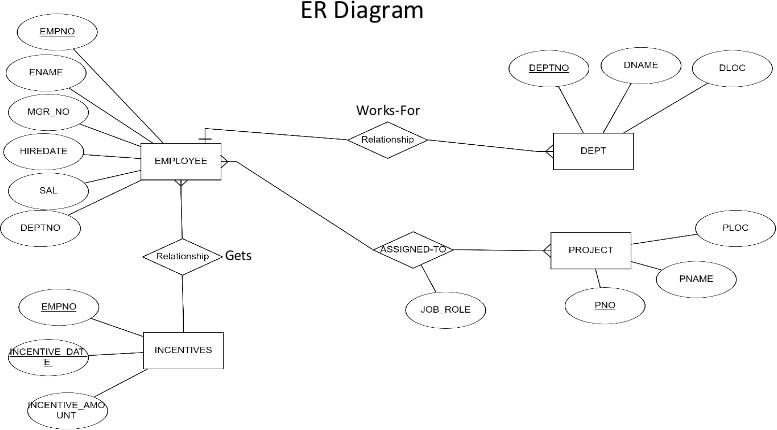
**select** e.ename,e.empno, d.dname, a.jobrole, d.dloc as deploc, p.ploc as proloc **from** employee e **join** dept d on e.depno = d.depno join assignment a **on** e.empno = a.empno **join** project p on a.pno=p.pno **where** d.dloc = p.ploc;



## More Queries on Insurance Database

### Question (Week 6)

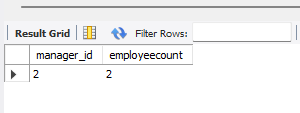
1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2. Enter greater than five tuples for each table.
3. List the name of the managers with the maximum employees
4. Display those managers name whose salary is more than average salary of his employee.
5.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



###### List the name of the managers with the maximum employees

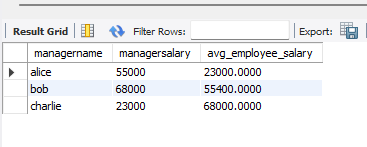
**select** mgrno as manager\_id, **count(**empno) as employeecount **from** employee **group by** mgrno **order by**

employeecount **desc** limit 1;



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

**select** m.ename as managername,m.sal **as** managersalary,emp\_avg.avg\_employee\_salary **from** employee m **join** (select mgrno,avg(sal)as avg\_employee\_salary from employee **group by** mgrno) as emp\_avg **on** m.empno=emp\_avg.mgrno;

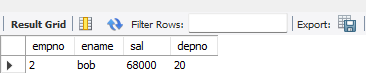


###### Find the name of the second top level managers of each department.

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

###### Find the employee details who got second maximum incentive in January 2019

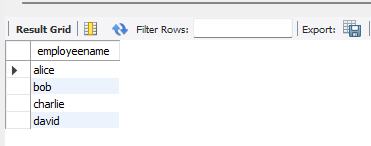
select e.empno,e.ename,e.sal, e.depno from employee e join incentives i on e.empno=i.empno where i.incentive\_date between '2022-11-10' and '2024-12-10' order by i.incentive\_amount desc limit 1 offset 1;



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

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

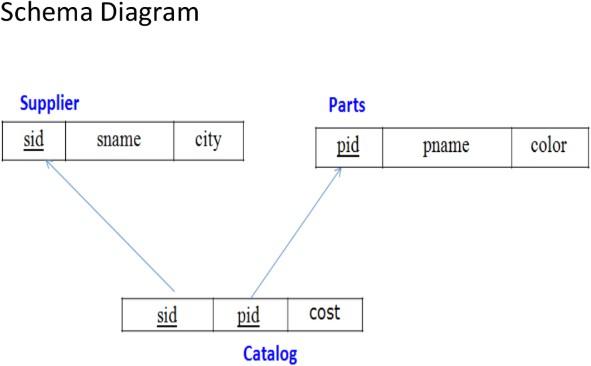
e.depno=m.depno;



## 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.



### Create Database

Create database supplier\_410; Use supplier\_410;

## Create tables

**create table** supplier( sid int **primary key,** sname varchar(20), city varchar(30)

);

desc supplier;

**create table** parts( pid int **primary key,** pname varchar(20), color varchar(20)

);

desc parts;

**create table** catalog(

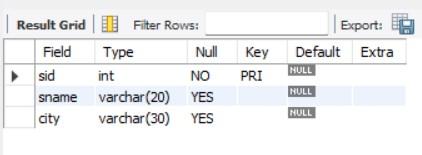
sid int, pid int, cost int,

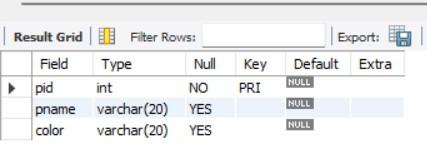
**foreign key**(sid) **references** supplier(sid),

**foreign key**(pid) **references** parts(pid)

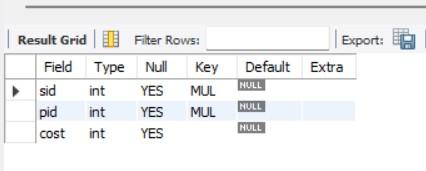
);

desc catalog;

**Supplier Table**

**Parts Table**

**Catalog table**



## Inserting the values

insert into supplier values (10001, "acne", "Bangalore"),

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

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

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

###### Supplier Table

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

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

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

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

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

###### Parts Table

(10001,20001,10),

(10001,20002,10),

(10001,20003,30),

(10001,20004,10),

(10001,20005,10),

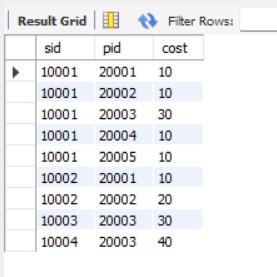
(10002,20001,10),

(10002,20002,20),

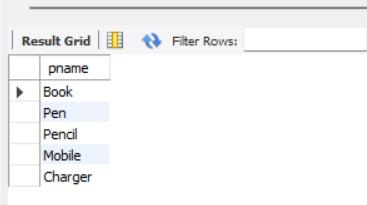
(10003,20003,30),

(10004,20003,40);

select \* from catalog;

**Catalog Table**

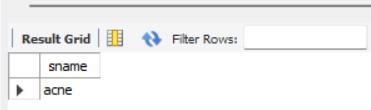
### Queries

* **Find the pnames of parts for which there is some supplier. select** pname **from** parts **where** pid **in** (**select** pid **from** catalog);

###### 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**

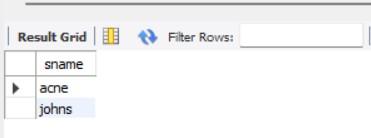
pid) **from** parts));



42

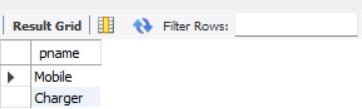
###### Find the snames of suppliers who supply every red part.

**select distinct** sname **from** supplier, parts, catalog

**where** supplier.sid = catalog.sid **and** parts.pid = catalog.pid **and** parts.color="Red";

###### 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"));



###### 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).

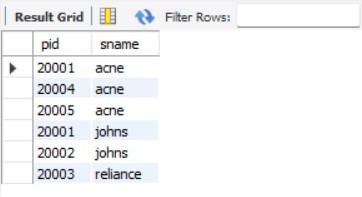
**select sid from** catalog a **where** a.cost > (**select avg**(b.cost) **from** catalog b **where** a.pid = b.pid **group by** b.pid);



###### 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;

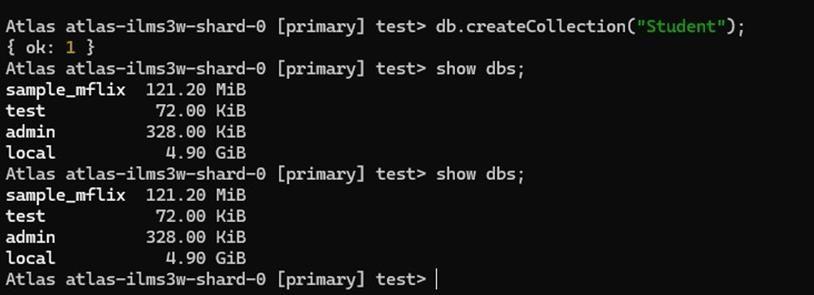


## NoSQL Lab 1

### 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 a query to update the 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**"**)*;* Show dbs;

### Create table & Inserting Values to the table

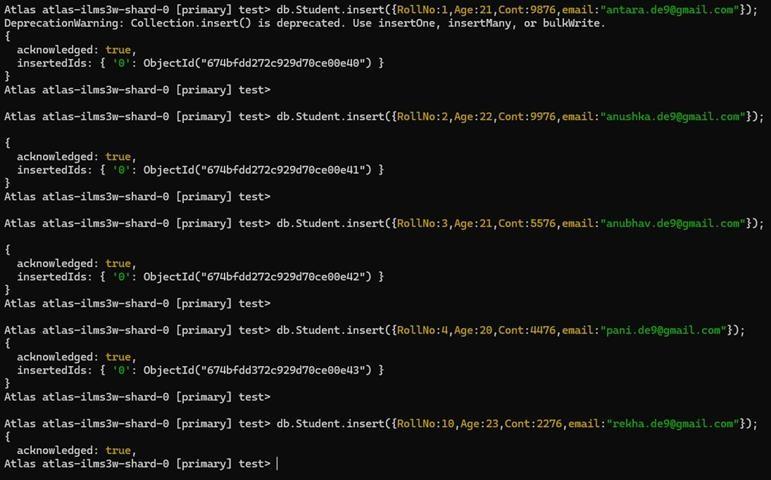
db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"[antara.de9@gmail.com](mailto:antara.de9@gmail.com)"});

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

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

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

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



### Structure of the table

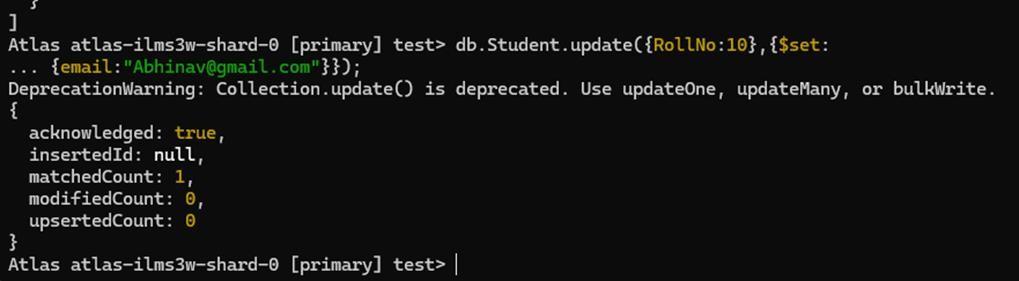
db.Student.**find()**;



### Queries

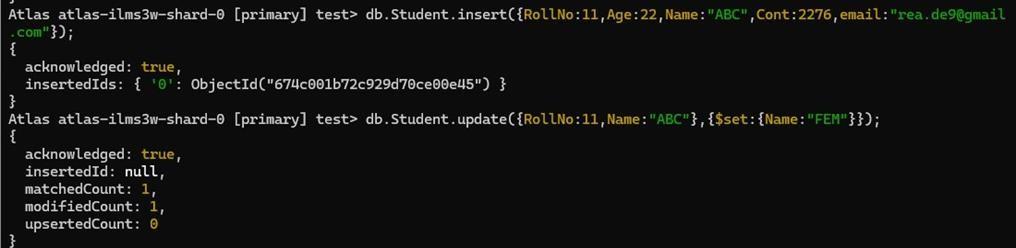
* **Write a query to update the Email-Id of a student with rollno 5.**

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



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

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



# NO SQL Customer Database

### Question (week-09)

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 the local file system into mongodb collection.

### Create Table:

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



# Inserting values into the table.

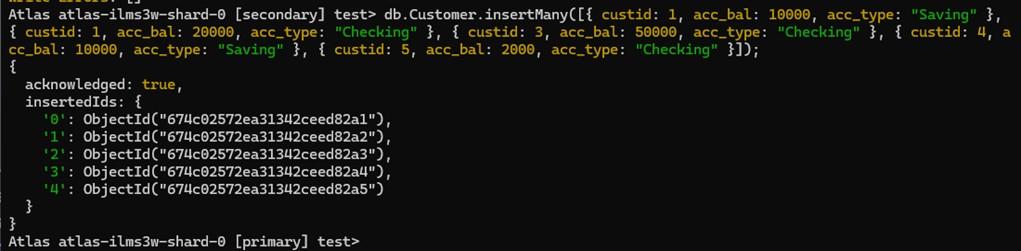
db.Customer.insertMany([{custid: 1, acc\_bal:10000, acc\_type:"Saving"},

{custid: 1, acc\_bal:20000, acc\_type: "Checking"},

{custid: 3,acc\_bal:50000, acc\_type: "Checking"},

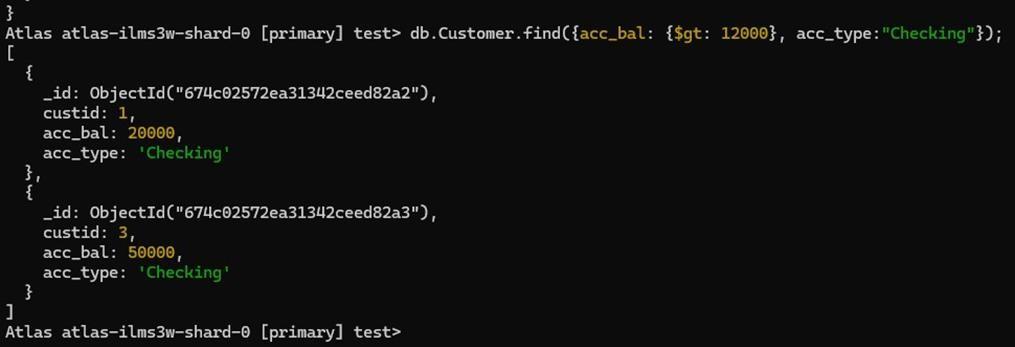
{custid: 4, acc\_bal:10000,acc\_type: "Saving"},

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



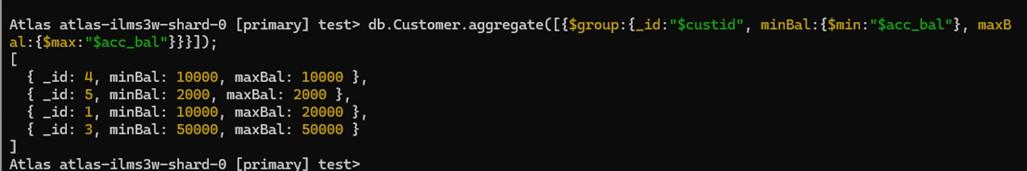
## Queries

Finding all checking accounts with balance greater than 12000 db.Customer.find({acc\_bal: {$gt: 12000}, acc\_type:"Checking"});

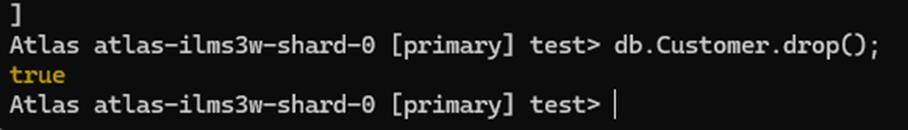


#### Finding the maximum and minimum balance of each customer

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



#### Dropping collection “Customer”

db.Customer.drop();

#### Exporting the collection to a json file

mongodb+srv://architavcs23:<db\_password>@cluster0.memuu.mongodb.net/-- collection=Customer-- out C:\Users\vijay\Documents\test.Customer.json

#### Exporting from a json file to the collection

mongodb+srv://architavcs23:<db\_password>@cluster0.memuu.mongodb.net/-- collection=Customer-- out C:\Users\vijay\Documents\test.Customer.json

db.Customer.find();

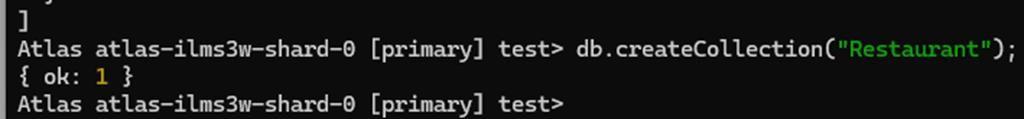
**NO SQL Restaurant Database**

### 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 order 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 zip code that starts with '10'.

#### Create Table

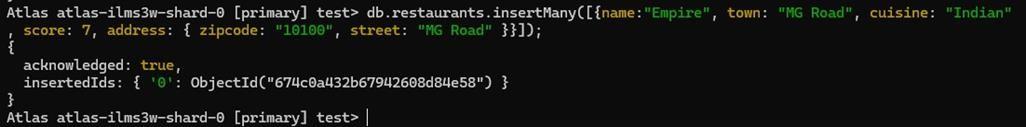
db.**createCollection**("Restaurant");



#### Inserting values into table

db.restaurants.insertMany([{ name: "Meghna Foods", town: "Jayanagar", cuisine: "Indian", score: 8, address: { zipcode: "10001", street: "Jayanagar“ } });

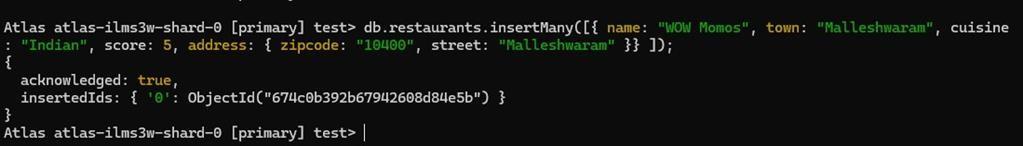
db.restaurants.insertMany([{ name: "Empire", town: "MG Road", cuisine: "Indian", score: 7, address: { zipcode: "10100", street: "MG Road" } }]);



db.restaurants.insertMany([{ name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: { zipcode: "20000", street: "Indiranagar" } }]);

db..restaurants.insertMany([{ name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode: "10300", street: "Majestic" } }]);



db.restaurants.insertMany([{ name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram" }} ]);

#### Queries

db.restaurants.find({});



#### Query to arrange the name of the restaurants in descending along with all the columns.

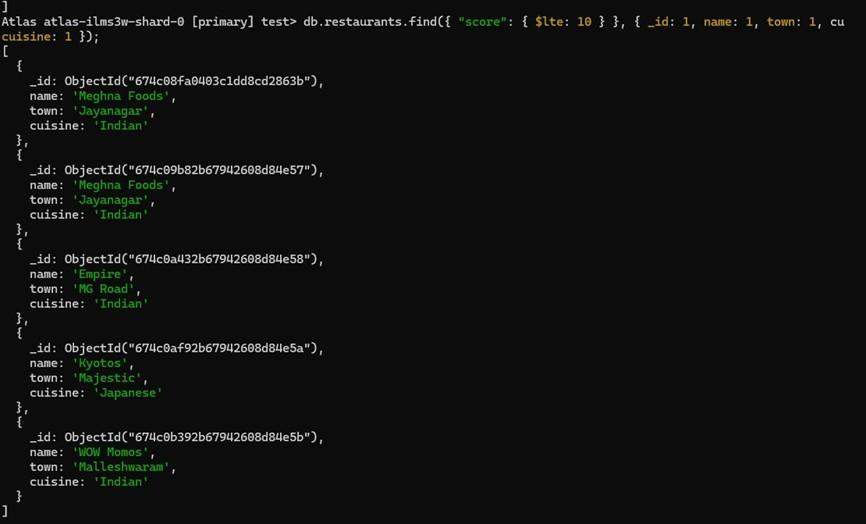
db.restaurants.find({}).sort({ name: -1 });





#### 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" } } } ]) ;

