**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

### LAB REPORT on

Database Management Systems (23CS3PCDBM)

***Submitted by***

**DARSHAN YG (1BM23CS087)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

### COMPUTER SCIENCE AND ENGINEERING

****

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

**(Autonomous Institution under VTU)**

**B. M. S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**

****

**CERTIFICATE**

This is to certify that the Lab work entitled “Database Management Systems (23CS3PCDBM)” carried out by **DARSHAN YG(1BM23CS087),** who is Bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

Dr. Kavitha Sooda Professor & HOD

Department of CSE, BMSCE

Dr Kayarvizhy N Assistant Professor

Department of CSE, BMSCE

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| **1** | **4-10-2024** | **Insurance Database** | **4** |
| **2** | **11-10-2024** | **More Queries on Insurance Database** | **13** |
| **3** | **18-10-2024** | **Bank Database** | **16** |
| **4** | **25-10-2024** | **More Queries on Bank Database** | **22** |
| **5** | **8-11-2024** | **Employee Database** | **25** |
| **6** | **15-11-2024** | **More Queries on Employee Database** | **30** |
| **7** | **22-11-2024** | **Supplier Database** | **31** |
| **8** | **20-12-2024** | **NO SQL - Student Database** | **35** |
| **9** | **20-12-2024** | **NO SQL - Customer Database** | **38** |
| **10** | **27-12-2024** | **NO SQL – Restaurant Database** | **40** |

# Insurance Database

### Question (Week 1)

* PERSON (driver\_id: String, name: String, address: String)
* CAR (reg\_num: String, model: String, year: int)
* ACCIDENT (report\_num: int, accident\_date: date, location: String)
* OWNS (driver\_id: String, reg\_num: String)
* PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)
* Create the above tables by properly specifying the primary keys and the foreign keys. **-**

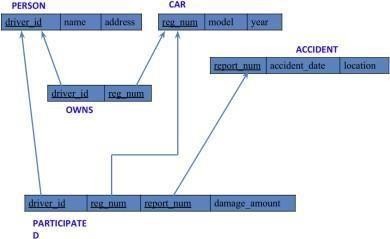
Enter at least five tuples for each relation

* Display Accident date and location
* Update the damage amount to 25000 for the car with a specific reg\_num (example 'K A053408'

) for which the accident report number was 12.

* Add a new accident to the database.
* To Do
* Find maximum damage amount
* Display driver id who did accident with damage amount greater than or average amount

### Schema Diagram



**Create database**

create database insure; use insure;

### Create table

create table insurance\_074( driver\_id varchar(20), name varchar(30), address varchar(50),

PRIMARY KEY(driver\_id)

);

create table insurance\_074.car( reg\_num varchar(15),

model varchar(10), year int,

PRIMARY KEY(reg\_num)

);

create table insurance\_074.owns(

driver\_Id varchar(20), reg\_num varchar(10),

PRIMARY KEY(driver\_id, reg\_num),

FOREIGN KEY(driver\_id) REFERENCES person(driver\_id), FOREIGN KEY(reg\_num) REFERENCES car(reg\_num)

);

create table insurance\_074.accident( report\_num int,

accident\_date date, location varchar(50),

PRIMARY KEY(report\_num)

);

Create table insurance\_074.participated( driver\_id varchar(20),

reg\_num varchar(10), report\_num int, damage\_amount int,

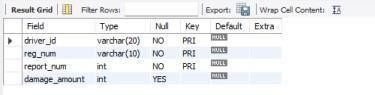
PRIMARY KEY(driver\_id,reg\_num,report\_num),

FOREIGN KEY(driver\_id) REFERENCES person(driver\_id), FOREIGN KEY(reg\_num) REFERENCES car(reg\_num), FOREIGN KEY(report\_num) REFERENCES accident(report\_num)

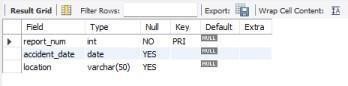
);

### Structure of the table

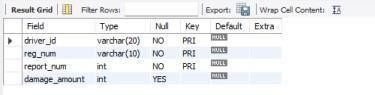
desc person;



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;



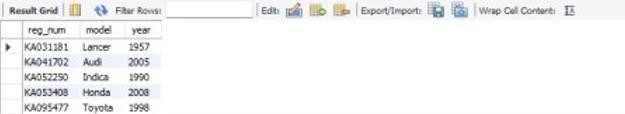
insert into car values("KA052250","Indica", "1990");

insert into carvalues("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;

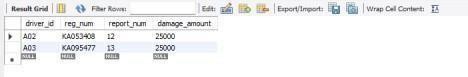


### Queries

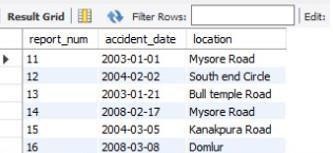
##### Update the damage amount to 25000 for the car with a specific reg-nu m (example'KA053408'

**) for which the accident report number was 12.** update participated set damage\_amount=25000

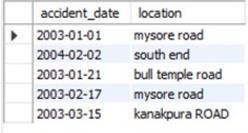
where reg\_num='KA053408' and report\_num=12;



**Add a new accident to the database.** insert into accident values(16,'2008-03-08',"Domlur"); select \* from accident;



##### Display Accident date and location

select accident\_date,location from accident;

**To-Do**

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

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



#### FIND MAXIMUM DAMAGE AMOUNT.

SELECT MAX(DAMAGE\_AMOUNT) FROM PARTICIPATED;

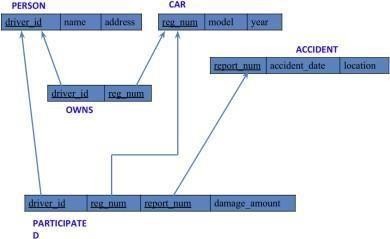


# More Queries on Insurance Database

### (Week 2)

* PERSON (driver\_id: String, name: String, address: String)
* CAR (reg\_num: String, model: String, year: int)
* ACCIDENT (report\_num: int, accident\_date: date, location: String)
* OWNS (driver\_id: String, reg\_num: String)
* PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)
* Display the entire CAR relation in the ascending order of manufacturing year.
* Find the number of accidents in which cars belonging to a specific model (example 'Lancer' were involved.
* DISPLAY DRIVER ID WHO DID ACCIDENT WITH DAMAGE AMOUNT GREATER THAN OR EQUAL TO RS.25000

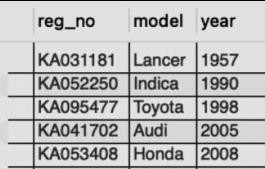
**Schema diagram**

****

**Queries**

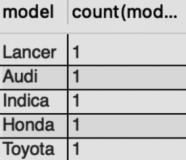
* + **Display the entire CAR relation in the ascending order of manufacturing year.**

select \* from car\_074 order by year asc;



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

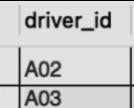
select model, count(model) from participated\_074 car\_074 where participated\_074.reg\_no = car\_074.reg\_no group by model;



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

select participated\_18.driver\_id as driver\_id from accident\_074, participated\_074 where

accident\_074.report\_no = participated\_074.report\_no and participated\_074.damage\_amt >= 25000;



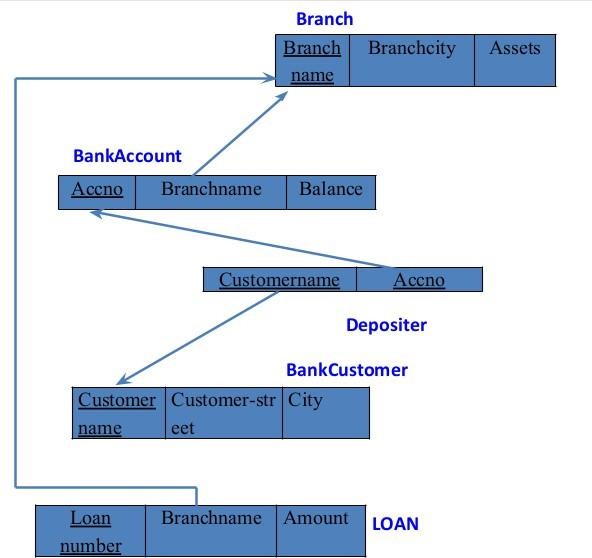
##### Bank Database

Question (Week 3)

* Branch (branch-name: String, branch-city: String, assets: real)
* BankAccount(accno: int, branch-name: String, balance: real)
* BankCustomer (customer-name: String, customer-street: String, customer-city: String) - Depositer(customer-name: String, accno: int)
* LOAN (loan-number: int, branch-name: String, amount: real)

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

* Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation.
* Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
* Find all the customers who have at least two accounts at the same branch (ex. SBI\_ResidencyRoad).
* Create a view which gives each branch the sum of the amount of all the loans at the branch. Schema diagram



DATABASE :

create database Bank\_Database; show databases; use Bank\_Database; create table Branch(Name varchar(20),City varchar(20), Assets varchar(20), primary key(Name));

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

create table Depositer(name varchar(20),accno varchar(20),

primary key(name,accno), foreign key(name) references Customer(name),foreign key(accno) references

BankAccount(accno));

create table Depositer(name varchar(20),accno varchar(20),

primary key(name,accno), foreign key(name) references Customer(name),foreign key(accno) references

BankAccount(accno));

create table Loan(loan\_no varchar(20),Name varchar(20), Amount varchar(20), primary key(Name), foreign key(Name) references Branch(Name));

insert into Branch values("SBI\_Chamrajpet", "Bangalore", 50000); insert into Branch values("SBI\_ResidencyRoad", "Bangalore", 10000);

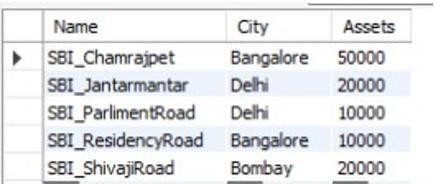
insert into Branch values("SBI\_ShivajiRoad", "Bombay", 20000); insert into Branch values("SBI\_ParlimentRoad", "Delhi", 10000); insert into Branch values("SBI\_Jantarmantar", "Delhi", 20000);

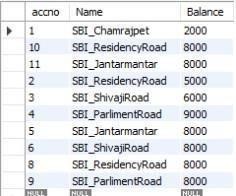
insert into BankAccount values(1, "SBI\_Chamrajpet",2000 ); insert into BankAccount values(2, "SBI\_ResidencyRoad", 5000); insert into BankAccount values(3, "SBI\_ShivajiRoad", 6000); insert into BankAccount values(4, "SBI\_ParlimentRoad", 9000); insert into BankAccount values(5, "SBI\_Jantarmantar", 8000); insert into BankAccount values(6, "SBI\_ShivajiRoad", 8000); insert into BankAccount values(8, "SBI\_ResidencyRoad", 8000); insert into BankAccount values(9, "SBI\_ParlimentRoad", 8000); insert into BankAccount values(10, "SBI\_ResidencyRoad", 8000); insert into BankAccount values(11, "SBI\_Jantarmantar", 8000); insert into Customer values("Avinash", "Bull temple road","Bangalore" ); insert into Customer values("Dinesh", "Bannerghatta Road","Bangalore" ); insert into Customer values("Mohan", "NationalCollegeRoad","Bangalore" ); insert into Customer values("Nikhil", "Akbar Road","Delhi" ); insert into Customer values("Ravi", "Prithviraj Road","Delhi" );

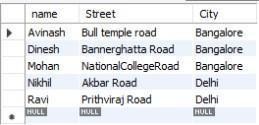
insert into Depositer values("Avinash", 1); insert into Depositer values("Dinesh", 2); insert into Depositer values("Mohan", 3); insert into Depositer values("Nikhil", 4); insert into Depositer values("Ravi", 5); insert into Depositer values("Avinash", 8); insert into Depositer values("Nikhil", 9); insert into Depositer values("Dinesh", 10); insert into Depositer values("Nikhil", 11);

insert into Loan values(1, "SBI\_Chamrajpet", 1000); insert into Loan values(2, "SBI\_ResidencyRoad", 2000); insert into Loan values(3, "SBI\_ShivajiRoad", 3000); insert into Loan values(4, "SBI\_ParlimentRoad", 4000); insert into Loan values(5, "SBI\_Jantarmantar", 5000);

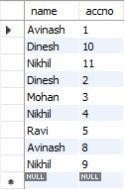
select \* from Branch;

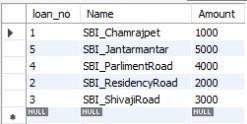


select \* from BankAccount;

select \* from Customer;

select \* from Depositer;



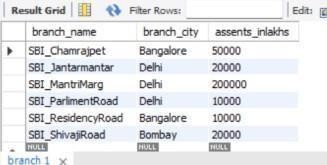
select \* from Loan;

Queries

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

alter table branch

change assets assents\_inlakhs real;



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

select d.Customername from Depositer d, BankAccount b where

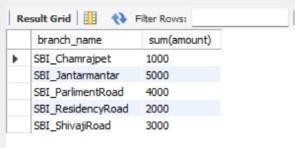
b.Branch\_name='SBI\_ResidencyRoad' and d.Accno=b.Accno group by d.Customername having count(d.Accno)>=2;



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

create view br as

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



More Queries on Bank Database

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

select distinct d.customer\_name

from Depositer d, BranchAccount ba, Branch b where d.accno=ba.accno and ba.branch\_name=b.branch\_name and b.branch\_city="Delhi"

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



1. Find all customers who have a loan at the bank but do not have an account. select b.customer\_name from borrower b

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



1. Find all customers who have both an account and a loan at the Bangalore branch select b.customer\_name from borrower b

where b.loan\_number in(select d.accno from depositer d,branchaccount ba, branch b where b.loan\_number=d.accno and d.accno=ba.accno and ba.branch\_name=b.branch\_name and

b.branch\_city="Bangalore");



1. Find the names of all branches that have greater assets than all branches located in Bangalore. select branch\_name

from branch

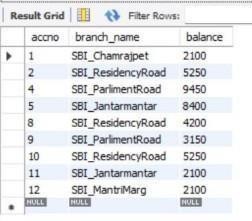
where assents\_inlakhs > all (select assents\_inlakhs from branch where branch\_city="Bangalore");



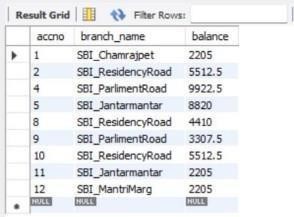
1. Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).

delete from branchaccount ba

where ba.branch\_name=(select b.branch\_name from branch b where branch\_city="Bombay"); select \* from branchaccount;



1. Update the Balance of all accounts by 5% update BranchAccount

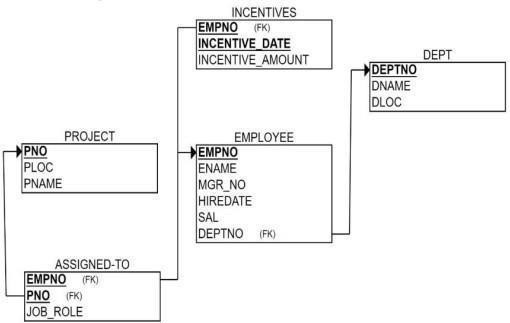
set balance=balance+((5\*balance)/100) where accno in(1,2,4,5,8,9,10,11,12);

# Employee Database

Question (Week 5)

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

Schema diagram



DATABASE :

create database Employee\_Database; use Employee\_Database;

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

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

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

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

create table Assingnedto(empno int, pno int,job\_role text, primary key(empno,pno), foreign key(empno)

references employee(empno), foreign key(pno) references project(pno));

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

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

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

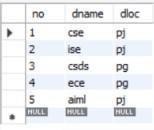
insert into project values(10,"bng","chatbot");

insert into project values(40,"delhi","ml model"); insert into project values(50,"bombay","blockchain"); insert into project values(30,"chennai","stocks"); insert into project values(80,"mysore","android app"); insert into Assingnedto values(101,10,"devops"); insert into Assingnedto values(201,40,"sde");

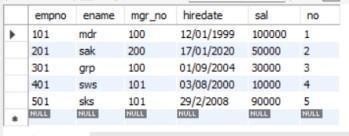
insert into Assingnedto values(301,50,"manager"); insert into Assingnedto values(401,30,"jpa");

insert into Assingnedto values(501,80,"pa");

select \* from dept;

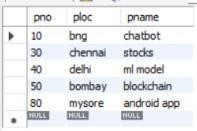


select \* from employee;

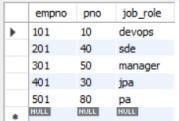


select \* from incentives;



select \* from project;

select \* from Assingnedto;



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

Hyderabad, or Mysuru

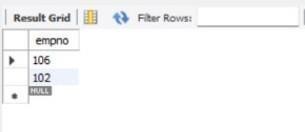
select a.empno

from assignedto a

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



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

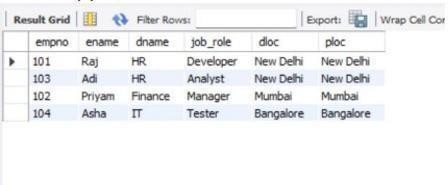
select e.empno from employee e where e.empno != all(select i.empno from incentives i);

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

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

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

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



## More Queries on Employee Database

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

select ename from employee where mgr\_no = (select max(mgr\_no) from employee);



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



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



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

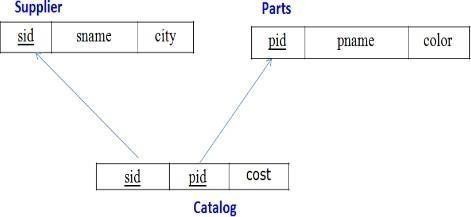


## Supplier Database

Question (Week 7)

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2. Insert appropriate records in each table.
3. Find the pnames of parts for which there is some supplier.
4. Find the snames of suppliers who supply every part.
5. Find the snames of suppliers who supply every red part.
6. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
7. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).
8. For each part, find the sname of the supplier who charges the most for that part.

Schema diagram



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

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

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

create table Catalog( sid int, pid

int, cost int,

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

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

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

insert into Catalog values (10001, 20001 , 10);

insert into Catalog values (10001, 20002 , 10);

insert into Catalog values (10001, 20003 , 30);

insert into Catalog values (10001, 20004 , 10);

insert into Catalog values (10001, 20005 , 10);

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

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

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

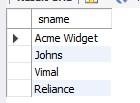
insert into Catalog values (10004, 20003 , 40);

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



1. Find the snames of suppliers who supply every part. select distinct s.sname from Catalog c , Supplier s where c.sid = s.sid and

NOT EXISTS(select p.pid from Parts p where NOT EXISTS(select c1.sid from Catalog c1 where c1.sid=c.sid and c1.pid =c.pid));

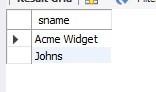


1. Find the snames of suppliers who supply every red part. select distinct s.sname

from Catalog C, Supplier s where C.sid=s.sid and

NOT EXISTS (select P.pid from Parts P

where P.color="Red" and NOT EXISTS (select C1.sid from Catalog C1 where C1.sid = C.sid and C1.pid = P.pid and P.color="Red"));

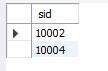


1. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

select p.pname

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



1. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part). select distinct C.sid from Catalog C where C.cost > (select AVG(C1.cost) from Catalog C1 where C1.pid = C.pid);
2. For each part, find the sname of the supplier who charges the most for that part.

select P.pid, S.sname from Parts P, Supplier S, Catalog C where C.pid

= P.pid and C.sid = S.sid and C.cost = (select max(C1.cost) from Catalog C1 where C1.pid = P.pid);



## No SQL – STUDENT DATABASE

Question (Week 8)

Perform the following DB operations using MongoDB.

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

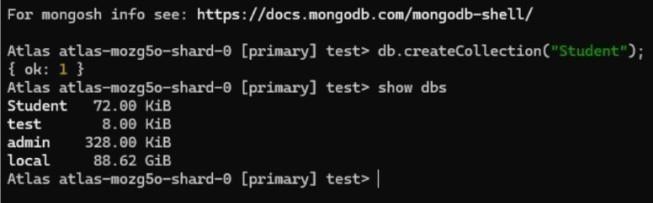
STRUCTURE OF THE COLLECTION

db.Student.find();

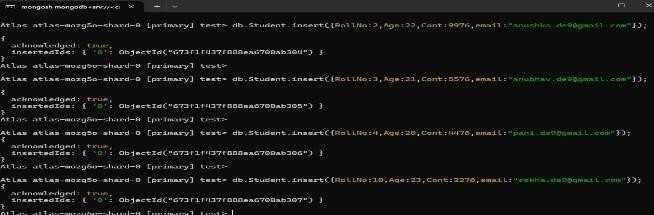
QUERIES

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

db.createCollection("Student"); show dbs

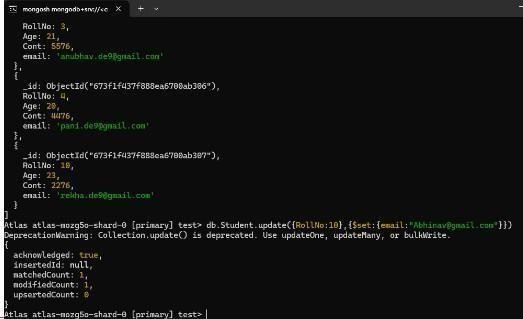


Insert appropriate values db.Student.insert({RollNo:1,Age:21,Cont:9876,email:["antara.de9@gmail.com"](mailto:antara.de9@gmail.com)}); db.Student.insert({RollNo:2,Age:22,Cont:9976,email:["anushka.de9@gmail.com"](mailto:anushka.de9@gmail.com)}); db.Student.insert({RollNo:3,Age:21,Cont:5576,email:["anubhav.de9@gmail.com"}](mailto:anubhav.de9@gmail.com)); 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)});



* + Write a query to update the Email-Id of a student with rollno 5. db.Student.update({RollNo:10},{$set:

{email:["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"},{$se t:{Name:"FEM"}})



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



## NO SQL – CUSTOMER DATABASE

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

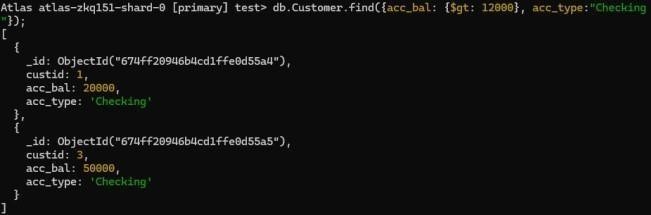
**Create Table:** db.createCollection("Customer"); **Inserting Values:**

db.Customer.insertMany([{custid: 1, acc\_bal:10000, acc\_type: "Saving"}, {custid: 1, acc\_bal:20000, acc\_type: "Checking"}, {custid: 3, acc\_bal:50000, acc\_type: "Checking"}, {custid: 4, acc\_bal:10000, acc\_type: "Saving"}, {custid: 5, acc\_bal:2000, acc\_type: "Checking"}]);



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

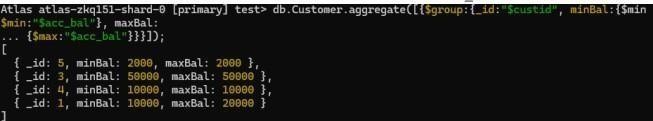
db.Customer.find({acc\_bal: {$gt: 12000}, acc\_type:"Checking"});



##### 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();



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



NO SQL – RESTAURANT DATABASE

1. Write a MongoDB query to display all the documents in thecollection restaurants.
2. Write a MongoDB query to arrange the name of the restaurants indescending along with all the columns.
3. Write a MongoDB query to find the restaurant Id, name, town andcuisine for those restaurants which achieved a score which is not more than 10.
4. Write a MongoDB query to find the average score for eachrestaurant.
5. Write a MongoDB query to find the name and address of therestaurants 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" } },

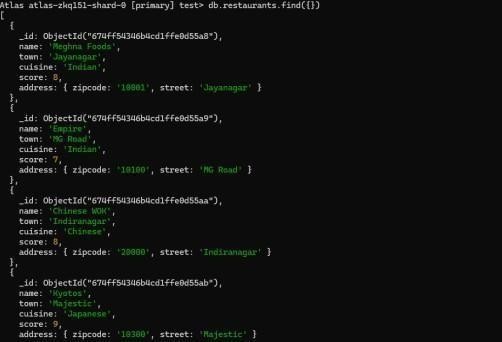
{ name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode: "10300", street: "Majestic" } },

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

} ])

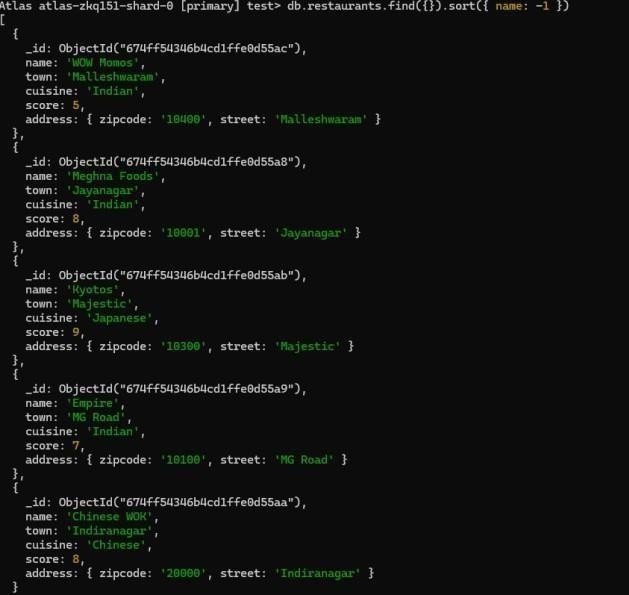


Write a MongoDB query to display all the documents in the collection restaurants. db.restaurants.find({})



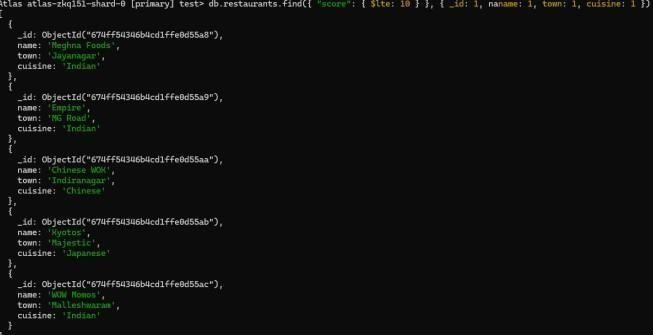
Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

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 })



