**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**Database Management Systems (22CS3PCDBM)**

***Submitted by***

**DHIKSHA RATHIS (1BM21CS055)**

***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**

**October-2022 to Feb-2023**

**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 (22CS3PCDBM)” carried out by **DHIKSHA RATHIS(1BM21CS055),** 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 (22CS3PCDBM) work prescribed for the said degree.

Name of the Lab-Incharge               **Dr. Jyothi S Nayak**

Designation Professor and Head

Department of CSE Department of CSE

BMSCE, Bengaluru BMSCE, Bengaluru

`

**Index**

| **Sl. No.** | **Date** | **Experiment Title** | **Page No.** |
| --- | --- | --- | --- |
| 1 | 08-11-2022 | Insurance Database | 4-11 |
| 2 | 15-11-2022 | More Queries on Insurance Database | 12-20 |
| 3 | 22-11-2022 | Bank Database | 21-29 |
| 4 | 29-11-2022 | More Queries on Bank Database | 30-41 |
| 5 | 06-12-2022 | Employee Database | 42-49 |
| 6 | 13-12-2022 | More Queries on Employee Database | 50-59 |
| 7 | 20-12-2022 | Supplier Database | 60-66 |
| 8 | 27-12-2022 | Flight Database | 67-75 |
| 9 | 17-01-2023 | NoSQL Lab 1 | 76-80 |
| 10 | 24-01-2023 | NoSQL Lab 2 | 81-87 |

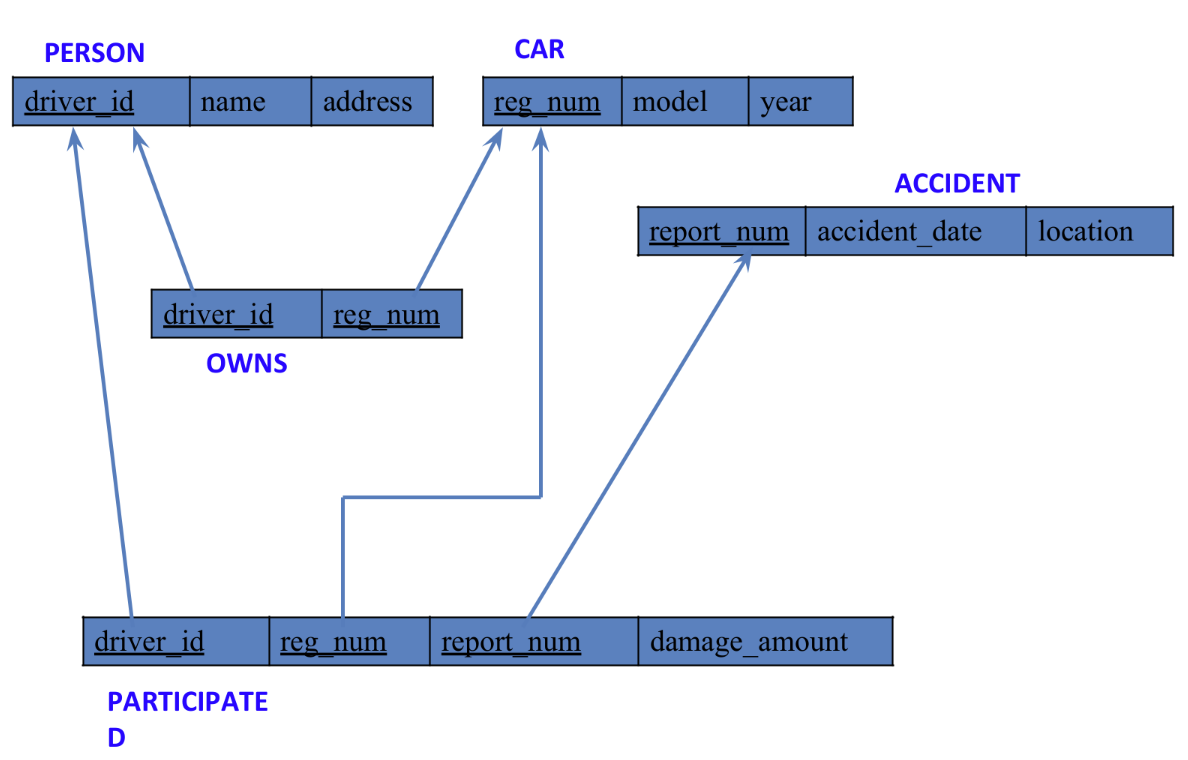
Insurance Database

**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\_dhiksha;

use insurance\_dhiksha;

**Create table**

create table insurance\_dhiksha.person(

driver\_id varchar(20),

name varchar(30),

address varchar(50),

PRIMARY KEY(driver\_id)

);

create table insurance\_dhiksha.car(

reg\_num varchar(15),

model varchar(10),

year int,

PRIMARY KEY(reg\_num)

);

create table insurance\_dhiksha.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\_dhiksha.accident(

report\_num int,

accident\_date date,

location varchar(50),

PRIMARY KEY(report\_num)

);

create table insurance\_dhiksha.participated(

driver\_id varchar(20),

reg\_num varchar(10),

report\_num int,

damage\_amount int,

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

FOREIGN KEY(driver\_id) REFERENCES person(driver\_id),

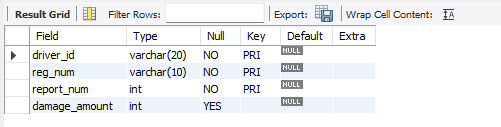
FOREIGN KEY(reg\_num) REFERENCES car(reg\_num),

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

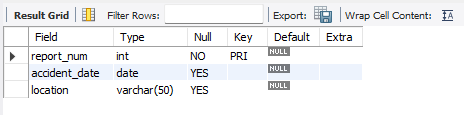
);

**Structure of the table**

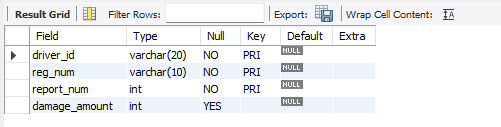
desc person;



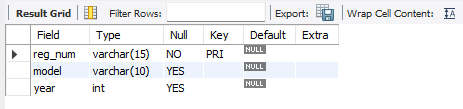
desc accident;



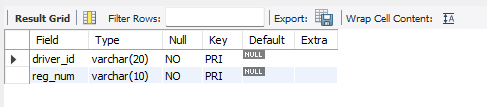
desc participated;



desc car;



desc owns;



**Inserting Values to the table**

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

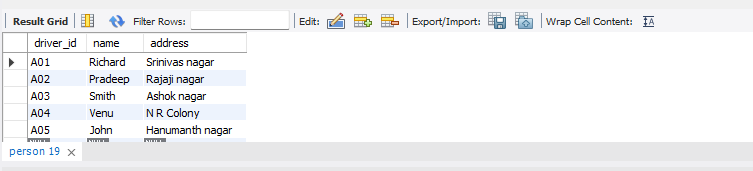
insert into person values("A02","Pradeep", "Rajaji nagar");

insert into person values("A03","Smith", "Ashok nagar");

insert into person values("A04","Venu", "N R Colony");

insert into person values("A05","John", "Hanumanth nagar");

select \* from person;



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

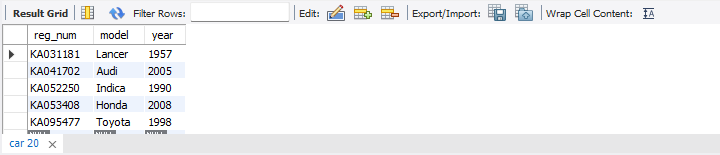
insert into car values("KA031181","Lancer", "1957");

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

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

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

select \* from car;



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

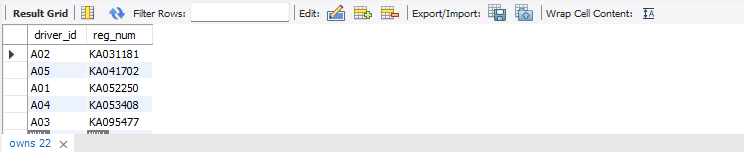
insert into owns values("A02","KA031181");

insert into owns values("A03","KA095477");

insert into owns values("A04","KA053408");

insert into owns values("A05","KA041702");

select \* from owns;



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

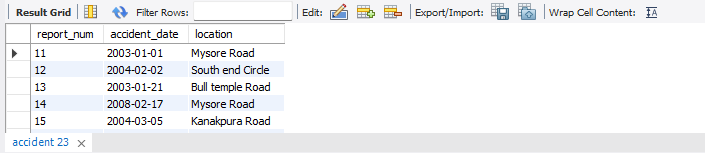
insert into accident values(12,'2004-02-02',"South end Circle");

insert into accident values(13,'2003-01-21',"Bull temple Road");

insert into accident values(14,'2008-02-17',"Mysore Road");

insert into accident values(15,'2004-03-05',"Kanakpura Road");

select \* from accident;



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

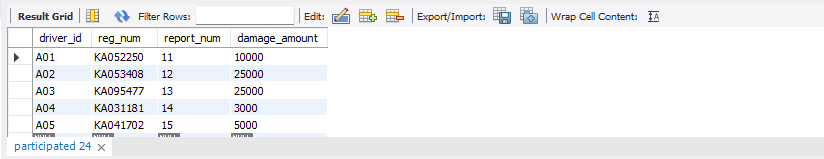
insert into participated values("A02","KA053408",12,50000);

insert into participated values("A03","KA095477",13,25000);

insert into participated values("A04","KA031181",14,3000);

insert into participated values("A05","KA041702",15,5000);

select \* from participated;



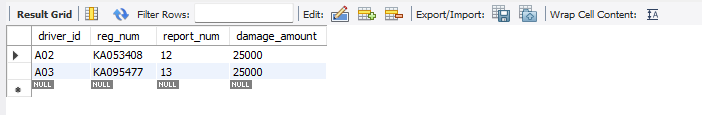
**Queries**

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

update participated

set damage\_amount=25000

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

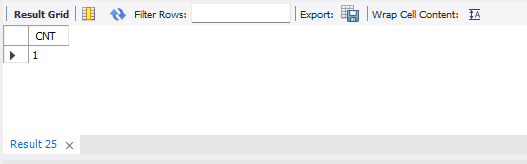


* **Find the total number of people who owned cars that were involved in accidents in 2008.**

select count(distinct driver\_id) CNT

from participated a, accident b

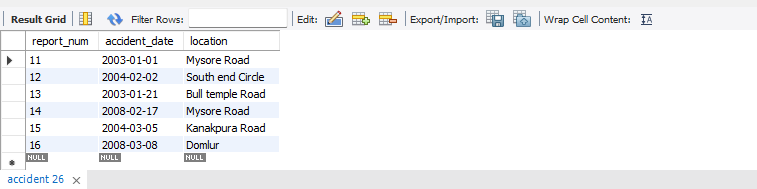
where a.report\_num=b.report\_num and b.accident\_date like '2008%';



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

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

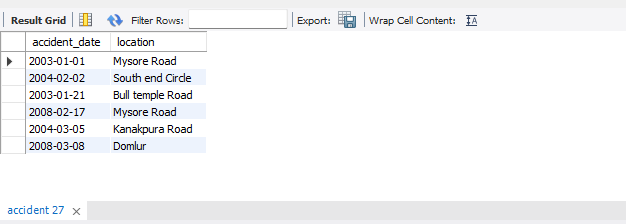
select \* from accident;



**TO DO:**

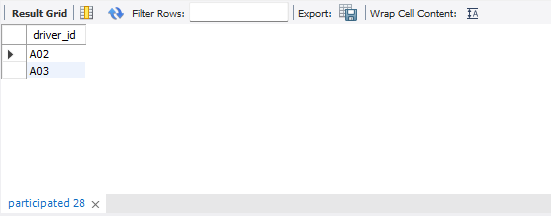
* **DISPLAY ACCIDENT DATE AND LOCATION**

select accident\_date, location from accident;



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

select driver\_id from participated where damage\_amount>=25000;



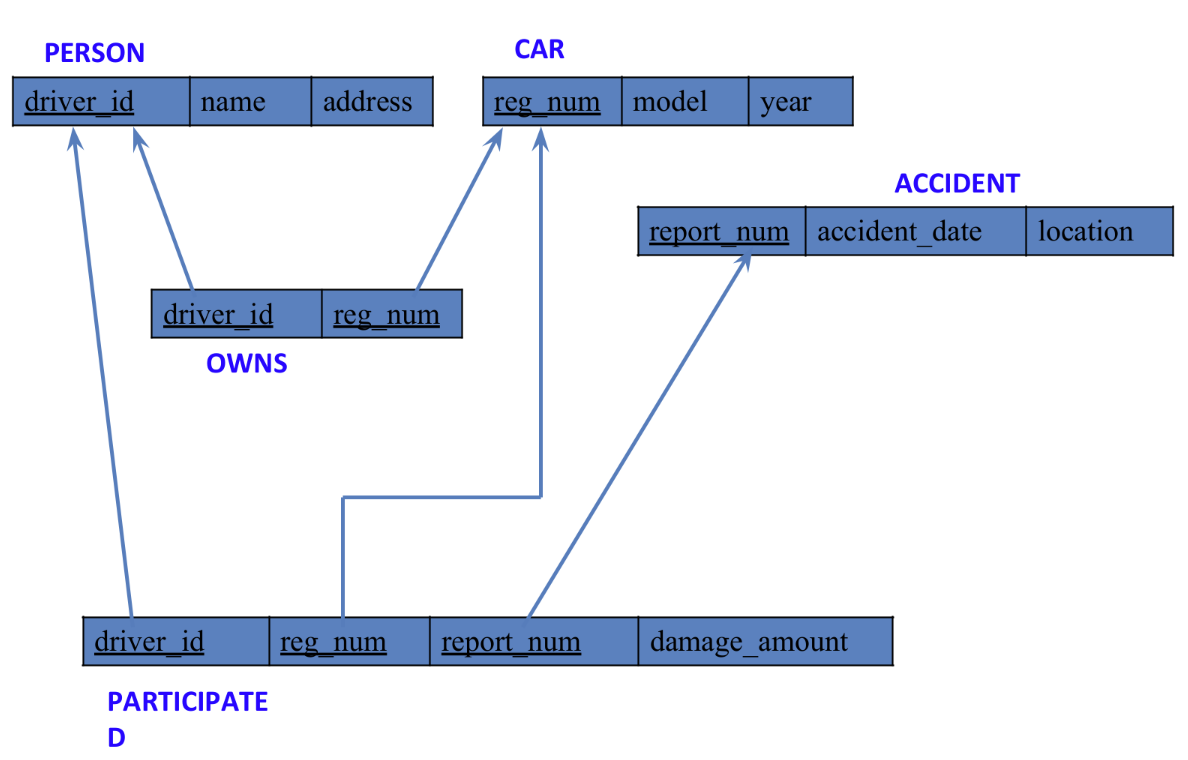
More Queries on Insurance Database

**Question**

**(Week 2)**

* PERSON (driver\_id: String, name: String, address: String)
* CAR (reg\_num: String, model: String, year: int)
* ACCIDENT (report\_num: int, accident\_date: date, location: String)
* OWNS (driver\_id: String, reg\_num: String)
* PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)
* Display the entire CAR relation in the ascending order of manufacturing year.
* Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
* Find the total number of people who owned cars that were involved in accidents in 2008.

**Schema Diagram**



**Create database**

create database insurance\_dhiksha;

use insurance\_dhiksha;

**Create table**

create table insurance\_dhiksha.person(

driver\_id varchar(20),

name varchar(30),

address varchar(50),

PRIMARY KEY(driver\_id)

);

create table insurance\_dhiksha.car(

reg\_num varchar(15),

model varchar(10),

year int,

PRIMARY KEY(reg\_num)

);

create table insurance\_dhiksha.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\_dhiksha.accident(

report\_num int,

accident\_date date,

location varchar(50),

PRIMARY KEY(report\_num)

);

create table insurance\_dhiksha.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),

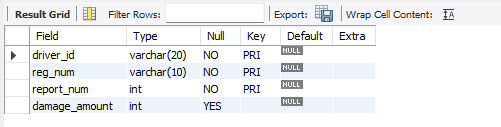
FOREIGN KEY(reg\_num) REFERENCES car(reg\_num),

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

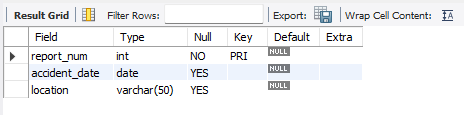
);

**Structure of the table**

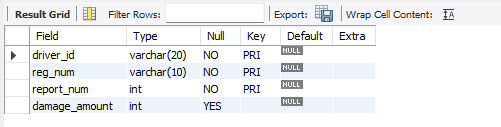
desc person;



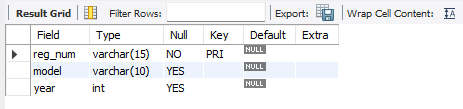
desc accident;



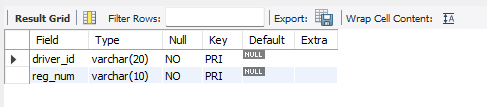
desc participated;



desc car;



desc owns;



**Inserting Values to the table**

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

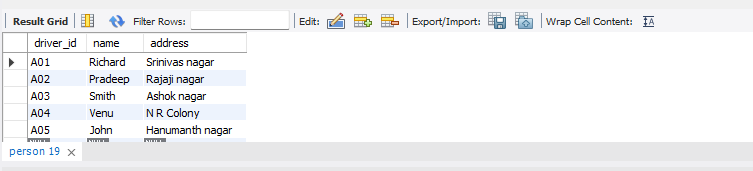
insert into person values("A02","Pradeep", "Rajaji nagar");

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");

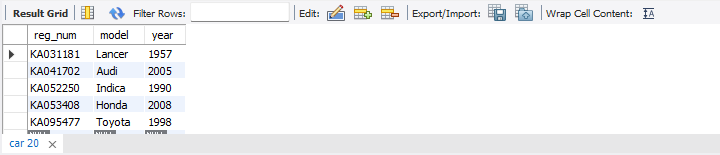
insert into car values("KA031181","Lancer", "1957");

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");

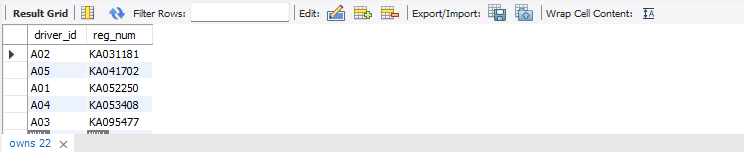
insert into owns values("A02","KA031181");

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");

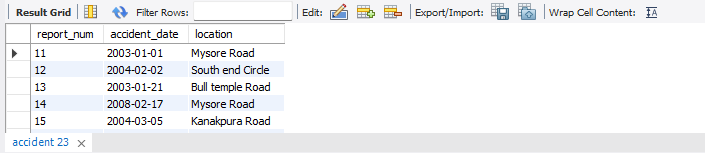
insert into accident values(12,'2004-02-02',"South end Circle");

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);

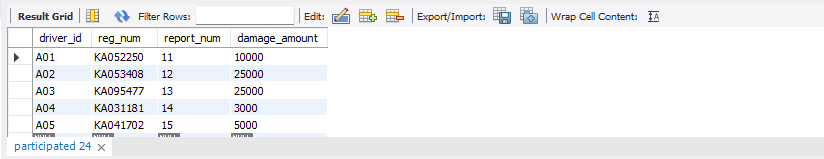
insert into participated values("A02","KA053408",12,50000);

insert into participated values("A03","KA095477",13,25000);

insert into participated values("A04","KA031181",14,3000);

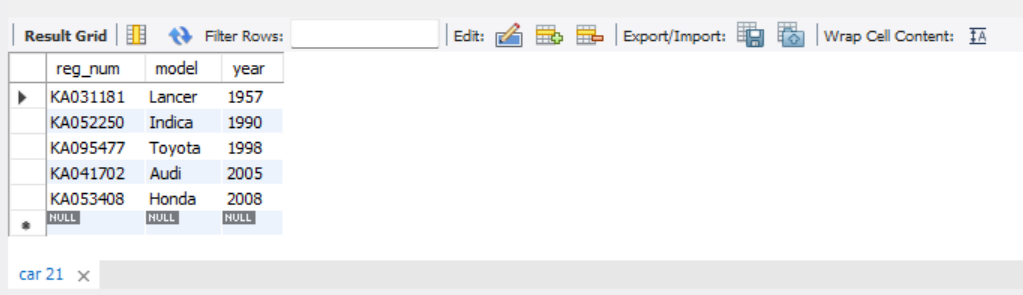
insert into participated values("A05","KA041702",15,5000);

select \* from participated;



**Queries**

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

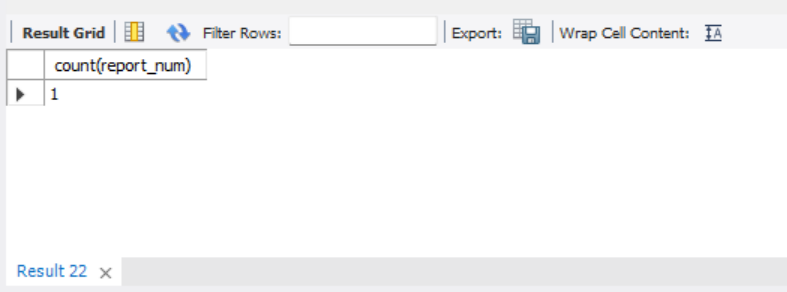
select \* from car order by year asc; 

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

select count(report\_num)

from car c, participated p

where c.reg\_num=p.reg\_num and c.model='Lancer';

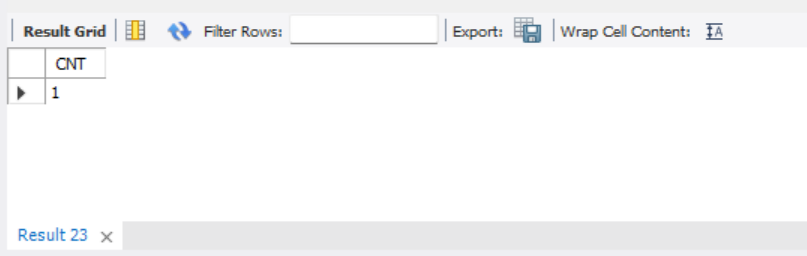


* **Find the total number of people who owned cars that were involved in accidents in 2008.**

select count(distinct driver\_id) CNT

from participated a, accident b

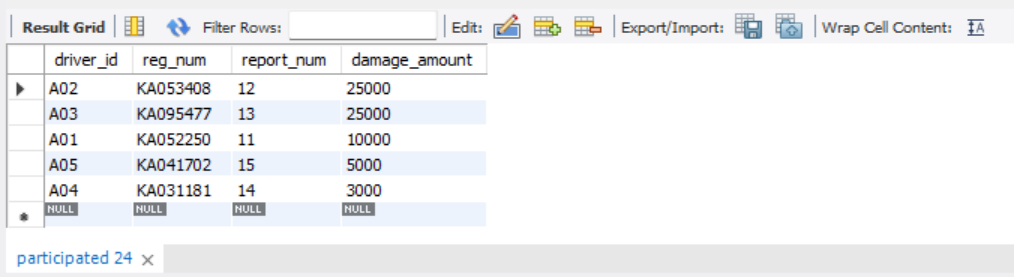
where a.report\_num=b.report\_num and b.accident\_date like '\_\_08%';



**TO DO:**

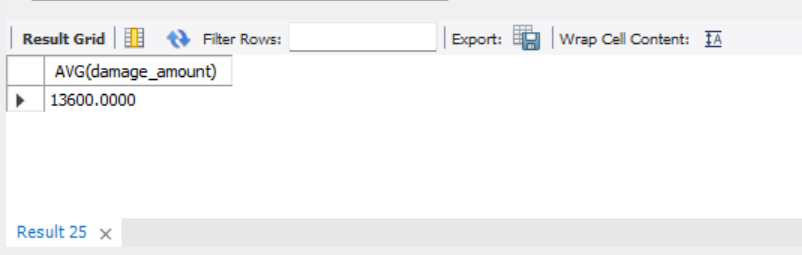
* **LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER OF DAMAGE AMOUNT.**

select \* from participated order by damage\_amount desc;



* **FIND THE AVERAGE DAMAGE AMOUNT**

SELECT AVG(damage\_amount) from participated;



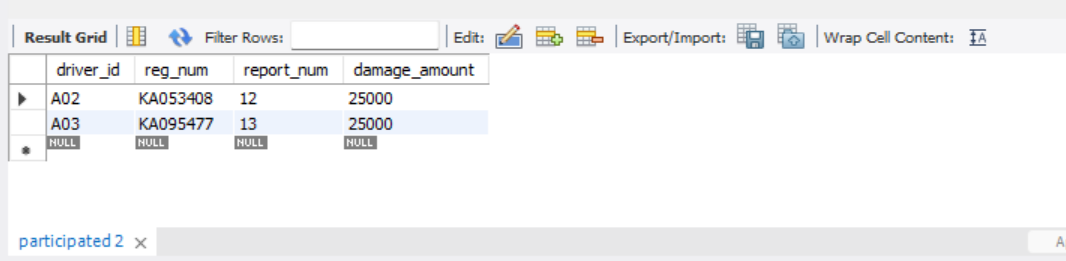
* **DELETE THE TUPLE WHOSE DAMAGE AMOUNT IS BELOW THE AVERAGE DAMAGE AMOUNT**

delete from participated

where damage\_amount < (select p.damage\_amount from(select AVG(damage\_amount) as damage\_amount FROM participated )p);

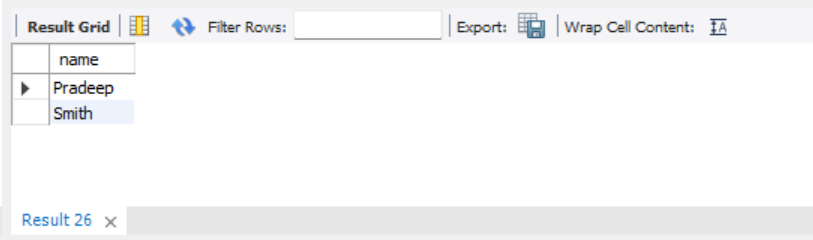


select \* from participated;



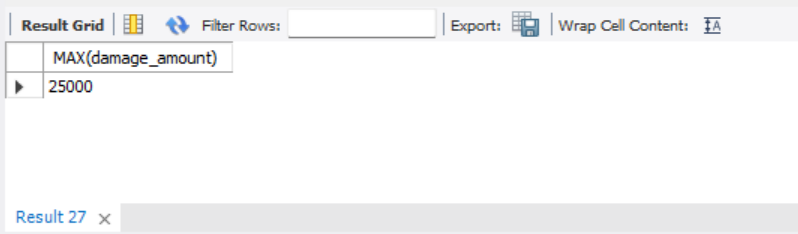
* **LIST THE NAME OF DRIVERS WHOSE DAMAGE IS GREATER THAN THE AVERAGE DAMAGE AMOUNT.**

select name from person p, participated part where p.driver\_id=part.driver\_id and damage\_amount>(select AVG(damage\_amount) FROM participated);

****

* **FIND MAXIMUM DAMAGE AMOUNT.**

select MAX(damage\_amount) from participated;

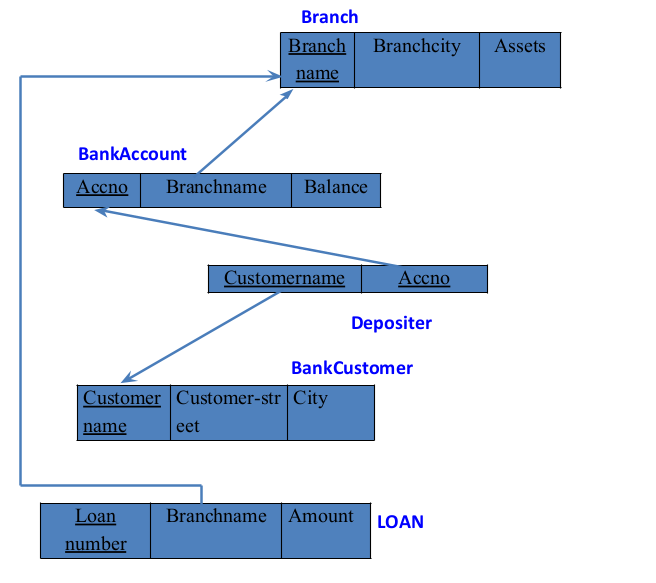


Bank Database

**Question**

**(Week 3)**

* Branch (branch-name: String, branch-city: String, assets: real)
* BankAccount(accno: int, branch-name: String, balance: real)
* BankCustomer (customer-name: String, customer-street: String, customer-city: String)
* Depositer(customer-name: String, accno: int)
* LOAN (loan-number: int, branch-name: String, amount: real)
* Create the above tables by properly specifying the primary keys and the foreign keys.
* Enter at least five tuples for each relation.
* Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
* Find all the customers who have at least two accounts at the same branch (ex. SBI\_ResidencyRoad).
* Create a view which gives each branch the sum of the amount of all the loans at the branch.

**Schema Diagram** 

**Create database**

create database dhiksha\_bank;

use dhiksha\_bank;

**Create table**

create table dhiksha\_bank.branch(

Branch\_name varchar(30),

Branch\_city varchar(25),

assets int,

PRIMARY KEY (Branch\_name)

);

create table dhiksha\_bank.BankAccount(

Accno int,

Branch\_name varchar(30),

Balance int,

PRIMARY KEY(Accno),

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

);

create table dhiksha\_bank.BankCustomer(

Customername varchar(20),

Customer\_street varchar(30),

CustomerCity varchar (35),

PRIMARY KEY(Customername)

);

create table dhiksha\_bank.Depositer(

Customername varchar(20),

Accno int,

PRIMARY KEY(Customername,Accno),

foreign key (Accno) references BankAccount(Accno),

foreign key (Customername) references BankCustomer(Customername)

);

create table dhiksha\_bank.Loan(

Loan\_number int,

Branch\_name varchar(30),

Amount int,

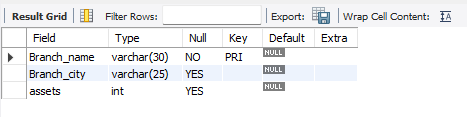
PRIMARY KEY(Loan\_number),

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

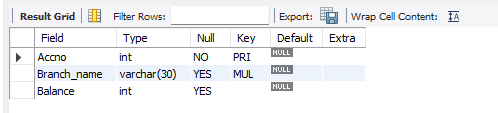
);

**Structure of the table**

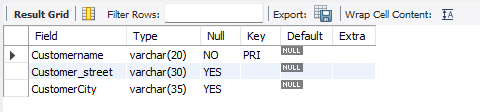
desc branch;



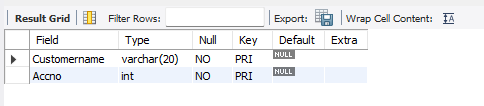
desc BankAccount;



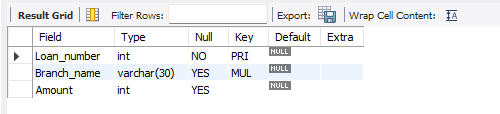
desc BankCustomer;



desc Depositer;



desc Loan;



**Inserting Values to the table**

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

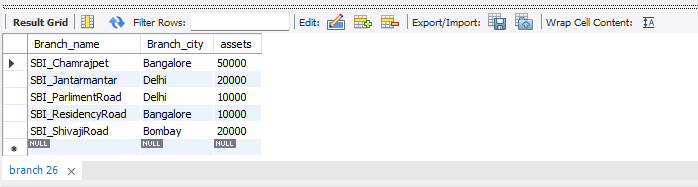
insert into branch values("SBI\_ResidencyRoad","Bangalore",10000);

insert into branch values("SBI\_ShivajiRoad","Bombay",20000);

insert into branch values("SBI\_ParlimentRoad","Delhi",10000);

insert into branch values("SBI\_Jantarmantar","Delhi",20000);

select \* from branch;



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",4000);

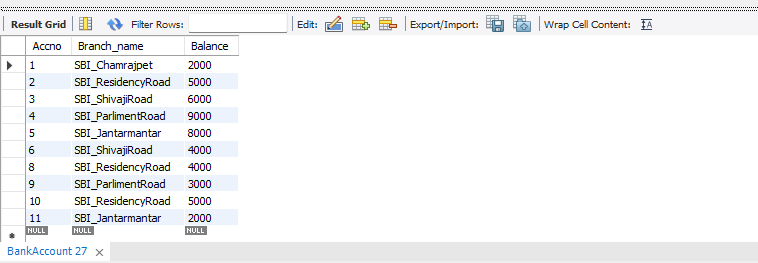
insert into BankAccount values(8,"SBI\_ResidencyRoad",4000);

insert into BankAccount values(9,"SBI\_ParlimentRoad",3000);

insert into BankAccount values(10,"SBI\_ResidencyRoad",5000);

insert into BankAccount values(11,"SBI\_Jantarmantar",2000);

select \* from BankAccount;



insert into BankCustomer values("Avinash","Bull\_Temple\_Road","Bangalore");

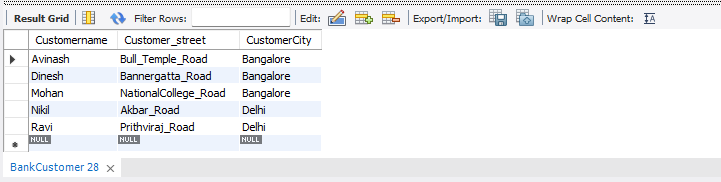
insert into BankCustomer values("Dinesh","Bannergatta\_Road","Bangalore");

insert into BankCustomer values("Mohan","NationalCollege\_Road","Bangalore");

insert into BankCustomer values("Nikil","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("Nikil",4);

insert into Depositer values("Ravi",5);

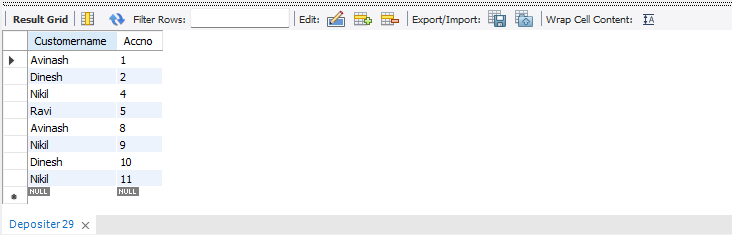
insert into Depositer values("Avinash",8);

insert into Depositer values("Nikil",9);

insert into Depositer values("Dinesh",10);

insert into Depositer values("Nikil",11);

select \* from Depositer;



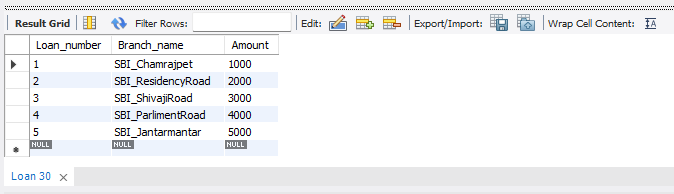
insert into Loan values(1,"SBI\_Chamrajpet",1000);

insert into Loan values(2,"SBI\_ResidencyRoad",2000);

insert into Loan values(3,"SBI\_ShivajiRoad",3000);

insert into Loan values(4,"SBI\_ParlimentRoad",4000);

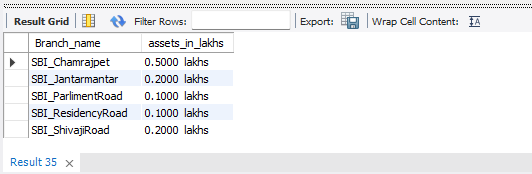
insert into Loan values(5,"SBI\_Jantarmantar",5000);

select \* from Loan;

**Queries**

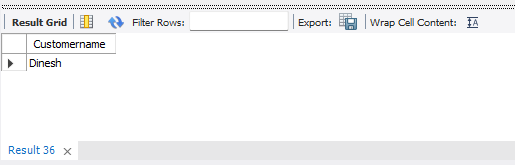
* **Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.**

select Branch\_name, CONCAT(assets/100000,' lakhs')assets\_in\_lakhs 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.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.**

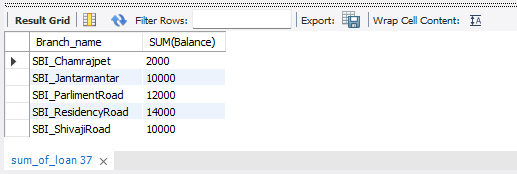
create view sum\_of\_loan

as select Branch\_name, SUM(Balance)

from BankAccount

group by Branch\_name;

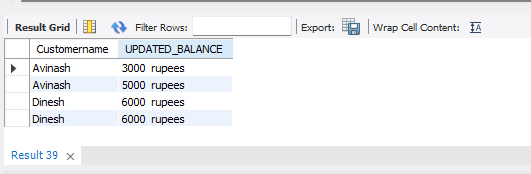
select \* from sum\_of\_loan;



**SPOT QUERY:**

**UPDATE OR ADD RUPEES 1000 TO ACCOUNT BALANCE FOR THE CUSTOMERS WHO ARE RESIDING IN BANGALORE.**

select bc.Customername, CONCAT(Balance+1000,' rupees') UPDATED\_BALANCE from BankAccount b, BankCustomer bc, Depositer d where bc.Customername=d.Customername and b.Accno=d.Accno and bc.Customercity='Bangalore';

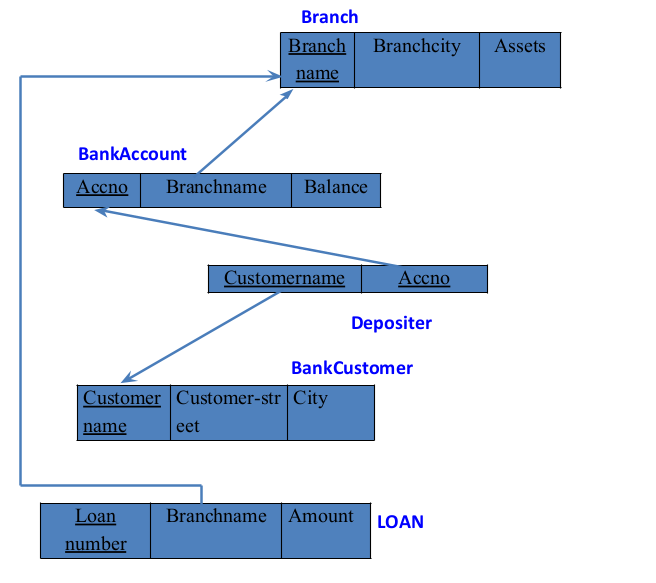


More Queries on Bank Database

**Question**

**(Week 4)**

* Branch (branch-name: String, branch-city: String, assets: real)
* BankAccount(accno: int, branch-name: String, balance: real)
* BankCustomer (customer-name: String, customer-street: String, customer-city: String)
* Depositer(customer-name: String, accno: int)
* LOAN (loan-number: int, branch-name: String, amount: real)
* Find all the customers who have an account at all the branches
* located in a specific city (Ex. Delhi).
* Find all customers who have a loan at the bank but do not have an account.
* Find all customers who have both an account and a loan at the Bangalore branch
* Find the names of all branches that have greater assets than all branches located in Bangalore.
* Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).
* Update the Balance of all accounts by 5%

**Schema Diagram** 

**Create database**

create database dhiksha\_bank;

use dhiksha\_bank;

**Create table**

create table dhiksha\_bank.branch(

Branch\_name varchar(30),

Branch\_city varchar(25),

assets int,

PRIMARY KEY (Branch\_name)

);

create table dhiksha\_bank.BankAccount(

Accno int,

Branch\_name varchar(30),

Balance int,

PRIMARY KEY(Accno),

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

);

create table dhiksha\_bank.BankCustomer(

Customername varchar(20),

Customer\_street varchar(30),

CustomerCity varchar (35),

PRIMARY KEY(Customername)

);

create table dhiksha\_bank.Depositer(

Customername varchar(20),

Accno int,

PRIMARY KEY(Customername,Accno),

foreign key (Accno) references BankAccount(Accno),

foreign key (Customername) references BankCustomer(Customername)

);

create table dhiksha\_bank.Loan(

Loan\_number int,

Branch\_name varchar(30),

Amount int,

PRIMARY KEY(Loan\_number),

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

);

create table Borrower(

Customername varchar(20),

Loan\_number int,

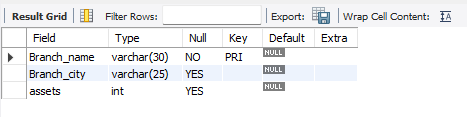
foreign key(Customername) references BankCustomer(Customername),

foreign key(Loan\_number) references Loan(Loan\_number)

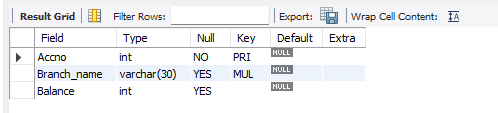
);

**Structure of the table**

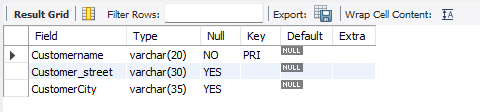
desc branch;



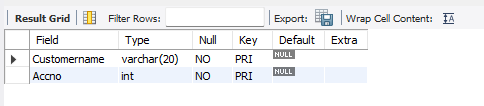
desc BankAccount;



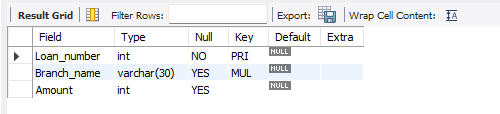
desc BankCustomer;



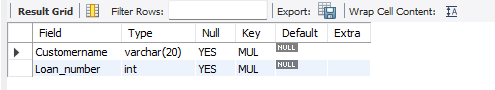
desc Depositer;



desc Loan;



desc Borrower;



**Inserting Values to the table**

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

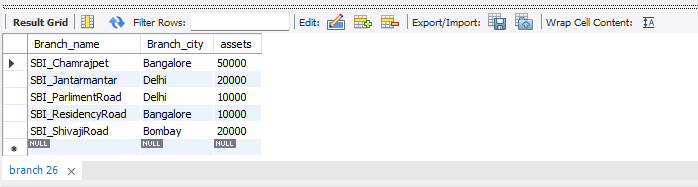
insert into branch values("SBI\_ResidencyRoad","Bangalore",10000);

insert into branch values("SBI\_ShivajiRoad","Bombay",20000);

insert into branch values("SBI\_ParlimentRoad","Delhi",10000);

insert into branch values("SBI\_Jantarmantar","Delhi",20000);

select \* from branch;



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",4000);

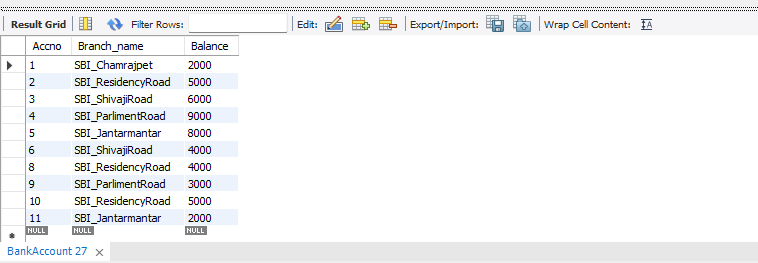
insert into BankAccount values(8,"SBI\_ResidencyRoad",4000);

insert into BankAccount values(9,"SBI\_ParlimentRoad",3000);

insert into BankAccount values(10,"SBI\_ResidencyRoad",5000);

insert into BankAccount values(11,"SBI\_Jantarmantar",2000);

select \* from BankAccount;



insert into BankCustomer values("Avinash","Bull\_Temple\_Road","Bangalore");

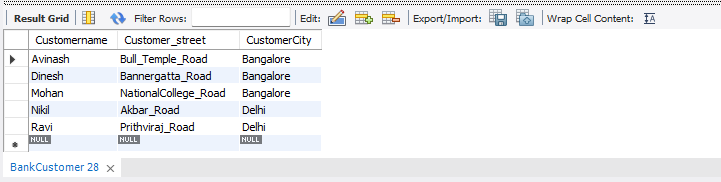
insert into BankCustomer values("Dinesh","Bannergatta\_Road","Bangalore");

insert into BankCustomer values("Mohan","NationalCollege\_Road","Bangalore");

insert into BankCustomer values("Nikil","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("Nikil",4);

insert into Depositer values("Ravi",5);

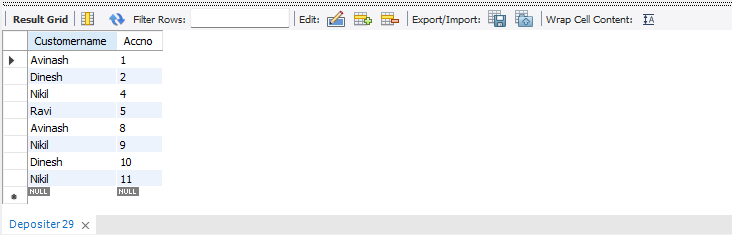
insert into Depositer values("Avinash",8);

insert into Depositer values("Nikil",9);

insert into Depositer values("Dinesh",10);

insert into Depositer values("Nikil",11);

select \* from Depositer;



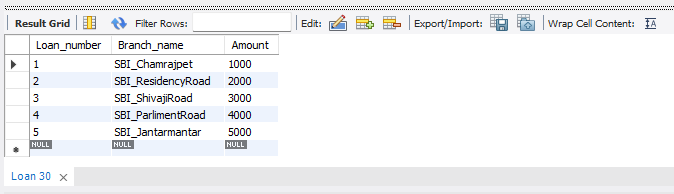
insert into Loan values(1,"SBI\_Chamrajpet",1000);

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 Loan;

insert into Borrower values("Avinash",1);

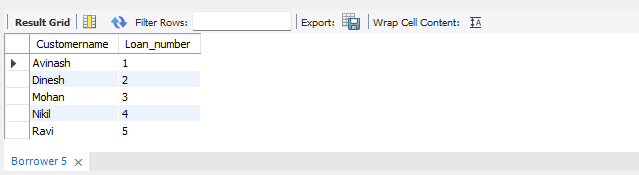
insert into Borrower values("Dinesh",2);

insert into Borrower values("Mohan",3);

insert into Borrower values("Nikil",4);

insert into Borrower values("Ravi",5);

select \* from Borrower ;

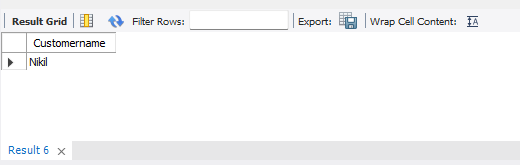


**Queries**

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

select d.Customername from branch b, Depositer d, BankAccount ba where

b.Branch\_city='Delhi' and d.Accno=ba.Accno and b.Branch\_name=ba.Branch\_name

group by d.Customername having count(distinct b.Branch\_name)= (select count(distinct b.Branch\_name) from branch b where b.Branch\_city='Delhi’;

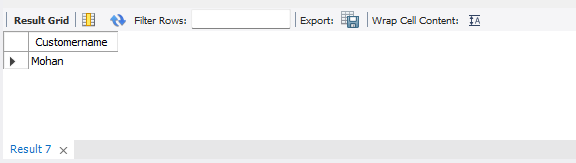
* **Find all customers who have a loan at the bank but do not have an account.**

select distinct b.Customername from Borrower b, Depositer d

where b.Customername NOT IN(

select d.Customername from Loan l,Depositer d, Borrower b

where l.Loan\_number=b.Loan\_number and d.Customername=b.Customername

);

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

select distinct d.Customername from Depositer d

where d.Customername IN(

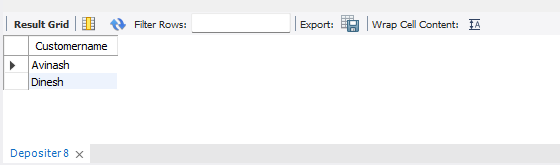
select d.Customername from branch br,Depositer d, BankAccount ba

where br.Branch\_city='Bangalore' and br.Branch\_name=ba.Branch\_name

and ba.accno=d.accno and Customername IN(

select Customername from Borrower)

);



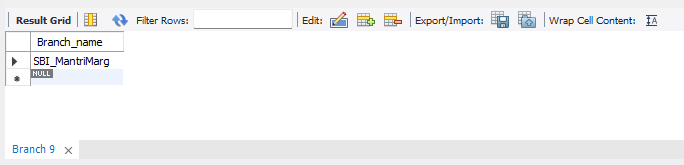
* **Find the names of all branches that have greater assets than all branches located in Bangalore.**

select b.Branch\_name from Branch b

where b.assets> ALL (

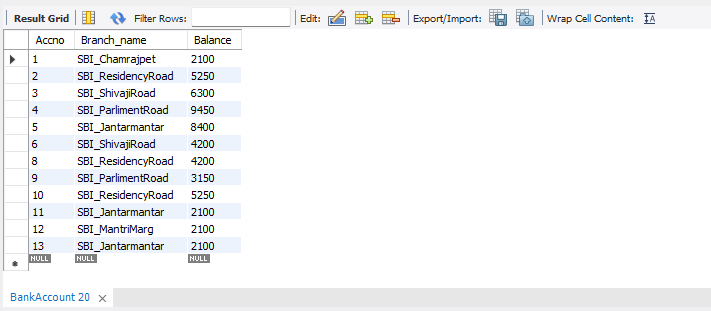
select SUM(b.assets) from Branch b

where b.Branch\_City='Bangalore' );



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

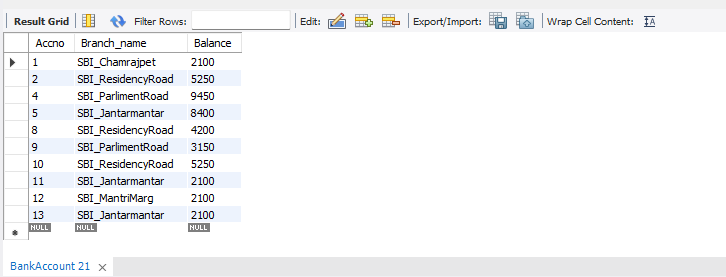
UPDATE BankAccount set Balance=(Balance + (Balance\*0.05));



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

delete ba.\* from BankAccount ba, branch b where branch\_city='Bombay' and ba.Branch\_name=b.Branch\_name;

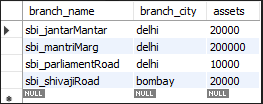
select \* from BankAccount;



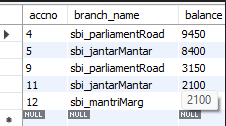
**SPOT QUERY: Demonstrate how to delete all the branches located in Bangalore**

delete b.\* from branch b where Branch\_city=’Bangalore’;

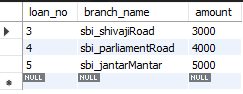
select \* from branch;



select \* from BankAccount;



select \* from Loan;

****

Employee Database

**Question**

**(Week 5)**

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

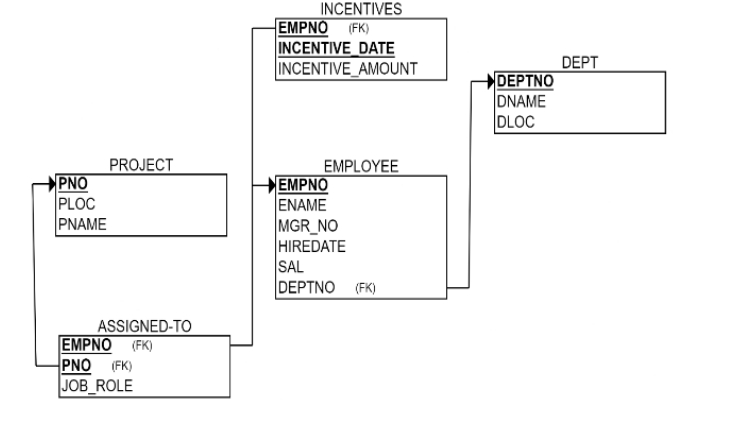
2. Enter greater than five tuples for each table.

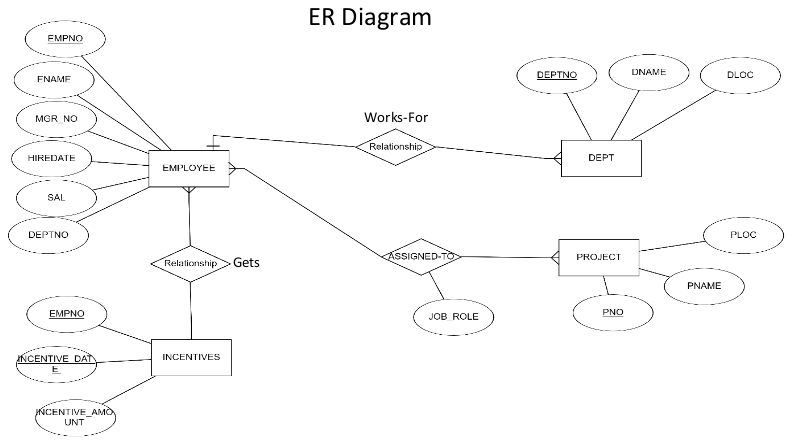
3. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru

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

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

**Schema Diagram**





**Create database**

create database dhiksha\_employee;

use dhiksha\_employee;

**Create table**

create table dhiksha\_employee.project(

pno int,

ploc varchar(40),

pname varchar(40),

PRIMARY KEY(pno)

);

create table dhiksha\_employee.dept(

deptno int,

dname varchar(40),

dloc varchar(40),

PRIMARY KEY(deptno)

);

create table dhiksha\_employee.employee(

empno int,

ename varchar(40),

mgr\_no int,

hiredate date,

sal int,

deptno int,

primary key (empno),

foreign key (deptno) references dept(deptno)

);

create table dhiksha\_employee.incentives(

empno int,

incentive\_date date,

incentive\_amount int,

primary key(incentive\_date),

foreign key (empno) references employee(empno)

);

create table dhiksha\_employee.assigned\_to(

empno int,

pno int,

job\_role varchar(50),

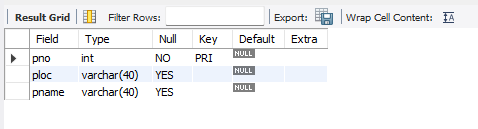
foreign key (pno) references project(pno),

foreign key (empno) references employee(empno)

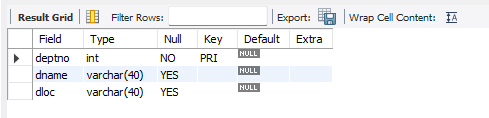
);

**Structure of the table**

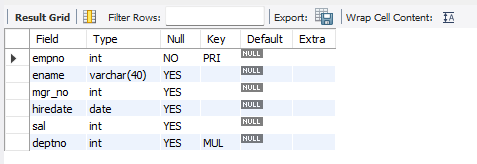
desc project;



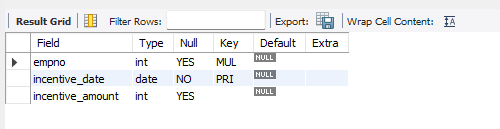
desc dept;



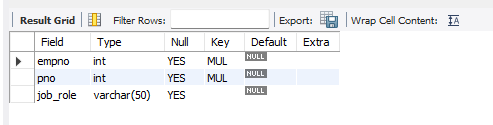
desc employee;



desc incentives;



desc assigned\_to;



**Inserting Values to the table**

insert into project values(1,"Bengaluru","Syntax");

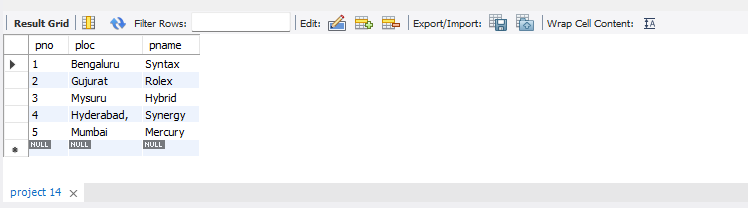
insert into project values(2,"Gujurat","Rolex");

insert into project values(3,"Mysuru","Hybrid");

insert into project values(4,"Hyderabad,","Synergy");

insert into project values(5,"Mumbai","Mercury");

select \* from project;



insert into dept values(10,"Sales","Bengaluru");

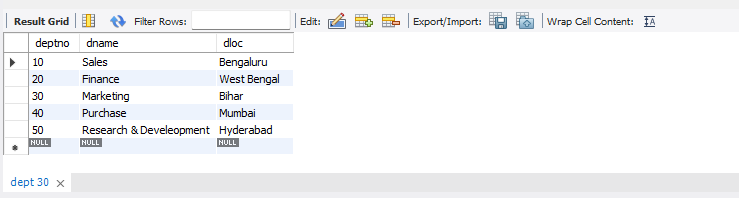
insert into dept values(20,"Finance","West Bengal");

insert into dept values(30,"Marketing","Bihar");

insert into dept values(40,"Purchase","Mumbai");

insert into dept values(50,"Research & Develeopment","Hyderabad");

select \* from dept;



insert into employee values(100,"Prannay",400,'2003-01-01',100000,10);

insert into employee values(200,"Farhaan",500,'2004-02-02',100500,50);

insert into employee values(300,"Sanika",100,'2003-01-21',200500,30);

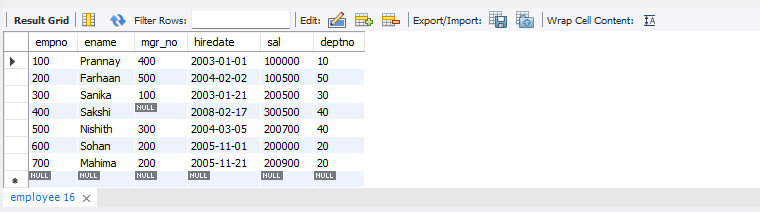
insert into employee values(400,"Sakshi", NULL ,'2008-02-17',300500,40);

insert into employee values(500,"Nishith",300,'2004-03-05',200700,40);

insert into employee values(600,"Sohan",200,'2005-11-01',200000,20);

insert into employee values(700,"Mahima",200,'2005-11-21',200900,20);

select \* from employee;



insert into incentives values(100,'2012-02-17',6000);

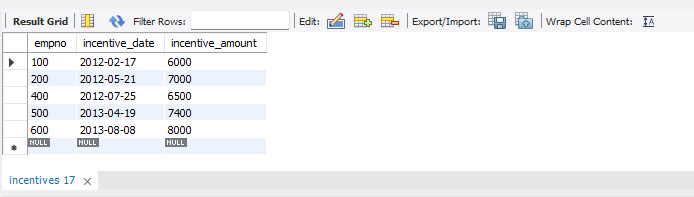
insert into incentives values(200,'2012-05-21',7000);

insert into incentives values(400,'2012-07-25',6500);

insert into incentives values(500,'2013-04-19',7400);

insert into incentives values(600,'2013-08-08',8000);

select \* from incentives;



insert into assigned\_to values(100,1, "Project Manager");

insert into assigned\_to values(200,1, "Resource Manager");

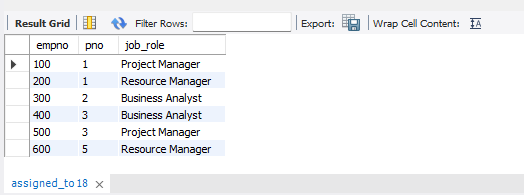
insert into assigned\_to values(300,2, "Business Analyst");

insert into assigned\_to values(400,3, "Business Analyst");

insert into assigned\_to values(500,3, "Project Manager");

insert into assigned\_to values(600,5, "Resource Manager");

select \* from assigned\_to;



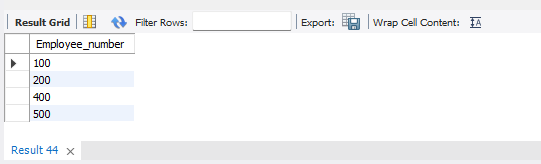
**Queries**

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

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

select a.empno Employee\_number from project p, assigned\_to a

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

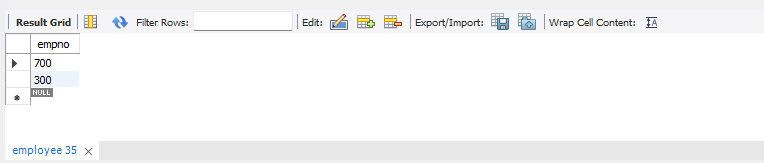


* **Get Employee ID’s of those employees who didn’t receive incentives**

select e.empno from employee e

where e.empno NOT IN

(select i.empno from incentives i);

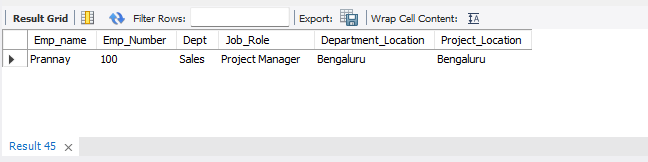


* **Write a SQL query to find the employees name, number, dept, job\_role, department location and project location who are working for a project location same as his/her department location.**

select e.ename Emp\_name, e.empno Emp\_Number, d.dname Dept, a.job\_role Job\_Role, d.dloc Department\_Location, p.ploc Project\_Location

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

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



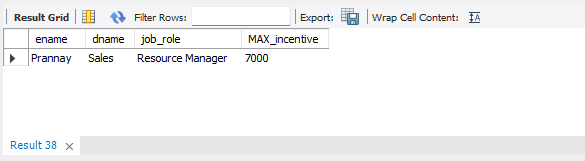
**SPOT QUERY:**

**Find the employee name, dept name and job role of an employee who received maximum incentive in year 2012**

select e.ename, d.dname, a.job\_role, MAX(i.incentive\_amount) MAX\_incentive

from employee e, dept d, incentives i, assigned\_to a

where incentive\_date between '2012-01-01' and '2012-12-31';



More Queries on Employee Database

**Question**

**(Week 6)**

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

and the foreign keys.

2. Enter greater than five tuples for each table.

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

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

employee.

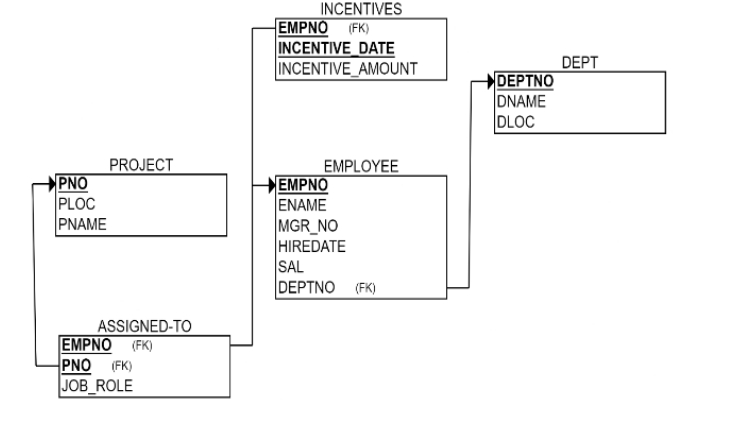
5. Find the name of the second top level managers of each department.

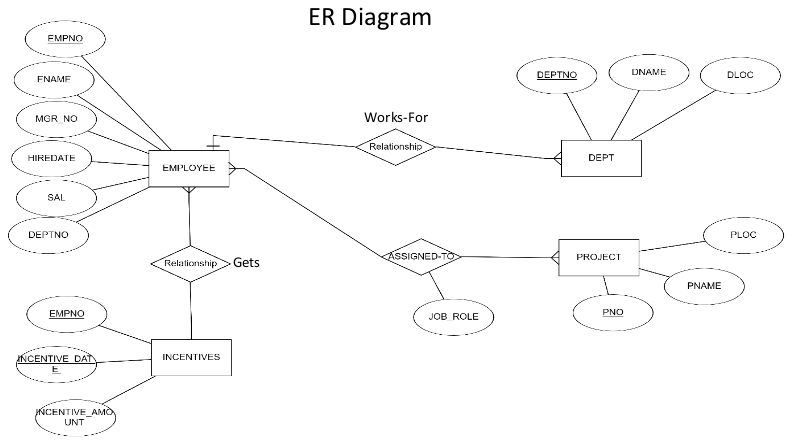
6. Find the employee details who got the second maximum incentive in January 2019.

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

the manager is working.

**Schema Diagram**





**Create database**

create database dhiksha\_employee;

use dhiksha\_employee;

**Create table**

create table dhiksha\_employee.project(

pno int,

ploc varchar(40),

pname varchar(40),

PRIMARY KEY(pno)

);

create table dhiksha\_employee.dept(

deptno int,

dname varchar(40),

dloc varchar(40),

PRIMARY KEY(deptno)

);

create table dhiksha\_employee.employee(

empno int,

ename varchar(40),

mgr\_no int,

hiredate date,

sal int,

deptno int,

primary key (empno),

foreign key (deptno) references dept(deptno)

);

create table dhiksha\_employee.incentives(

empno int,

incentive\_date date,

incentive\_amount int,

primary key(incentive\_date),

foreign key (empno) references employee(empno)

);

create table dhiksha\_employee.assigned\_to(

empno int,

pno int,

job\_role varchar(50),

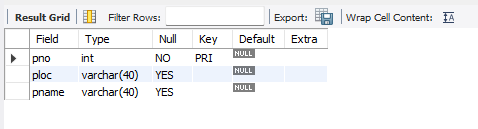
foreign key (pno) references project(pno),

foreign key (empno) references employee(empno)

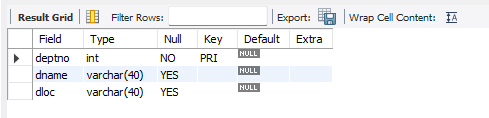
);

**Structure of the table**

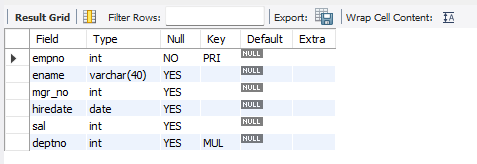
desc project;



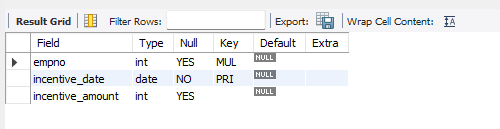
desc dept;



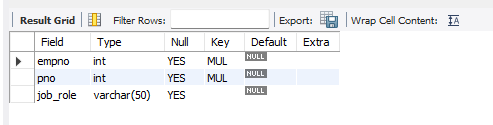
desc employee;



desc incentives;



desc assigned\_to;



**Inserting Values to the table**

insert into project values(1,"Bengaluru","Syntax");

insert into project values(2,"Gujurat","Rolex");

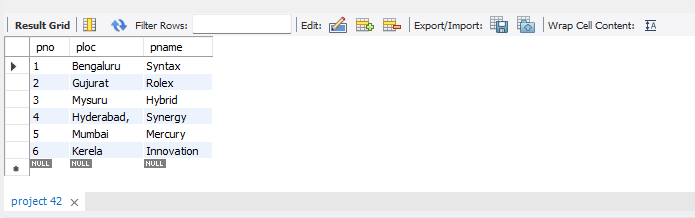
insert into project values(3,"Mysuru","Hybrid");

insert into project values(4,"Hyderabad,","Synergy");

insert into project values(5,"Mumbai","Mercury");

insert into project values(6,"Kerela","Innovation");

select \* from project;



insert into dept values(10,"Sales","Bengaluru");

insert into dept values(20,"Finance","West Bengal");

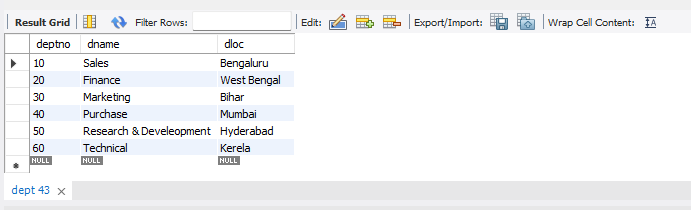
insert into dept values(30,"Marketing","Bihar");

insert into dept values(40,"Purchase","Mumbai");

insert into dept values(50,"Research & Development" ,"Hyderabad");

insert into dept values(60,"Technical","Kerela");

select \* from dept;



insert into employee values(100,"Prannay",700,'2003-01-01',24000,10);

insert into employee values(200,"Farhaan",100,'2004-02-02',17000,50);

insert into employee values(300,"Sanika",100,'2003-01-21',9000,30);

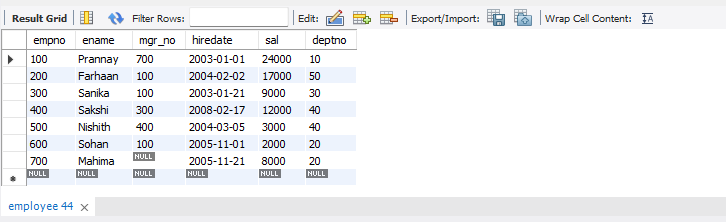
insert into employee values(400,"Sakshi", 300 ,'2008-02-17',12000,40);

insert into employee values(500,"Nishith",400,'2004-03-05',3000,40);

insert into employee values(600,"Sohan",100,'2005-11-01',2000,20);

insert into employee values(700,"Mahima",NULL,'2005-11-21',8000,20);

select \* from employee;



insert into incentives values(100,'2019-02-17',6000);

insert into incentives values(200,'2019-05-21',7000);

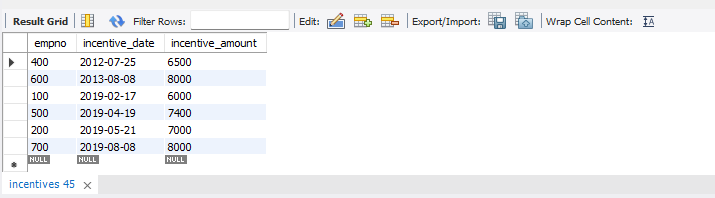
insert into incentives values(400,'2012-07-25',6500);

insert into incentives values(500,'2019-04-19',7400);

insert into incentives values(600,'2013-08-08',8000);

insert into incentives values(700,'2019-08-08',8000);

select \* from incentives;



insert into assigned\_to values(100,1, "Project Manager");

insert into assigned\_to values(200,1, "Resource Manager");

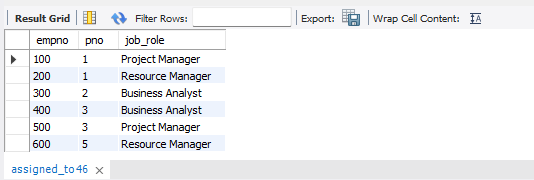
insert into assigned\_to values(300,2, "Business Analyst");

insert into assigned\_to values(400,3, "Business Analyst");

insert into assigned\_to values(500,3, "Project Manager");

insert into assigned\_to values(600,5, "Resource Manager");

select \* from assigned\_to;



**Queries**

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

select e1.ename

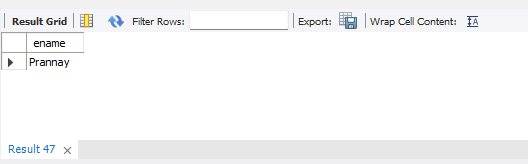
from employee e1, employee e2

where e1.empno=e2.mgr\_no group by e1.ename

having count(e1.mgr\_no)=(select count(e1.ename)

from employee e1, employee e2 where e1.empno=e2.mgr\_no

group by e1.ename order by count(e1.ename) desc limit 1);



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

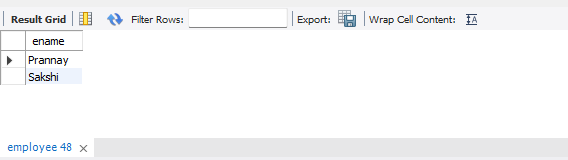
select m.ename from employee m

where m.empno in

(select mgr\_no from employee)

and m.sal>(select avg(n.sal) from employee n

where n.mgr\_no=m.empno);



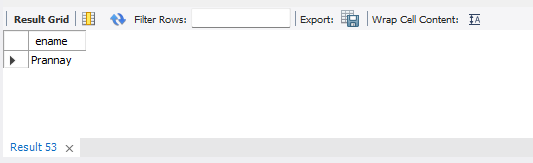
* **Find the name of the second top level managers of each department.**

select ename from employee where empno in(select distinct mgr\_no

from employee where empno in

(select distinct mgr\_no from employee where empno in

(select distinct mgr\_no from employee)));



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

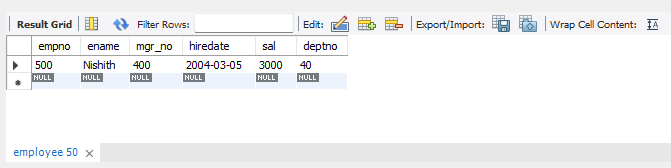
select \* from employee where empno=

(select i.empno from incentives i

where i.incentive\_amount= (select max(n.incentive\_amount) from incentives n

where n.incentive\_amount<(select max(inc.incentive\_amount) from incentives inc

where inc.incentive\_date between '2019-01-01' and '2019-12-31') and incentive\_date between '2019-01-01' and '2019-12-31'));

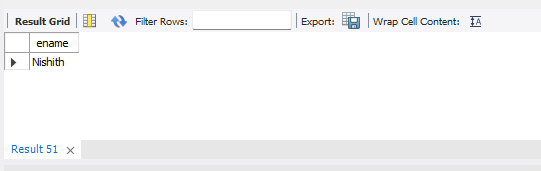


* **Display those employees who are working in the same department where his manager is working.**

select e2.ename

from employee e1, employee e2

where e1.empno=e2.mgr\_no and e1.deptno=e2.deptno;



Supplier Database

**Question**

**(Week 7)**

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

keys and the foreign keys.

2. Insert appropriate records in each table.

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

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

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

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

one else.

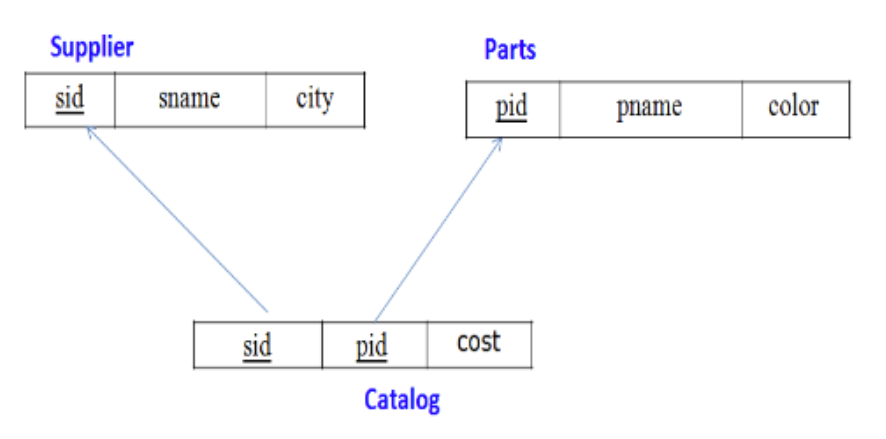
7. Find the sids of suppliers who charge more for some part than the average

cost of that part (averaged over all the suppliers who supply that part).

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

that part.

**Schema Diagram**



**Create database**

create database dhiksha\_supplier;

use dhiksha\_supplier;

**Create table**

create table dhiksha\_supplier.Supplier(

sid int,

sname varchar(15),

city varchar(10),

PRIMARY KEY(sid)

);

create table dhiksha\_supplier.Parts(

pid int,

pname varchar(10),

color varchar(5),

PRIMARY KEY(pid)

);

create table dhiksha\_supplier.Catalog(

sid int,

pid int,

cost int,

PRIMARY KEY(sid, pid),

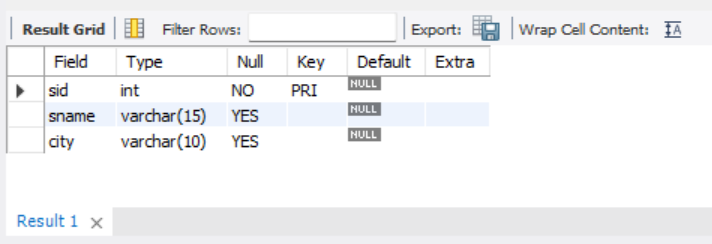
FOREIGN KEY(sid) REFERENCES Supplier(sid),

FOREIGN KEY(pid) REFERENCES Parts(pid)

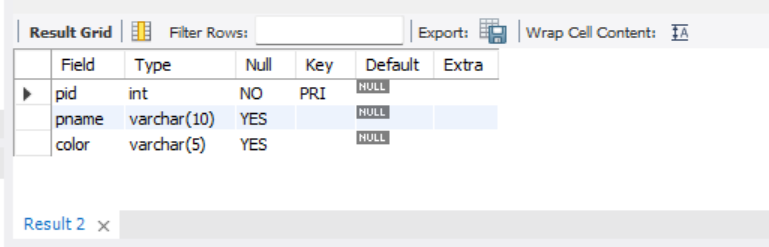
);

**Structure of the table**

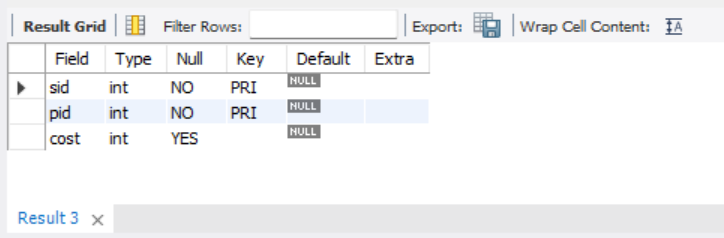
desc Supplier;



desc Parts;



desc Catalog;



**Inserting Values to the table**

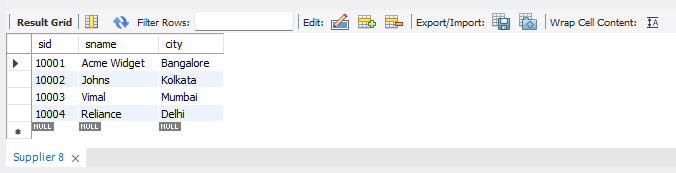
insert into Supplier values(10001,"Acme Widget", "Bangalore");

insert into Supplier values(10002,"Johns", "Kolkata");

insert into Supplier values(10003,"Vimal", "Mumbai");

insert into Supplier values(10004,"Reliance", "Delhi");

select \* from Supplier;



insert into Parts values(20001,"Book", "Red");

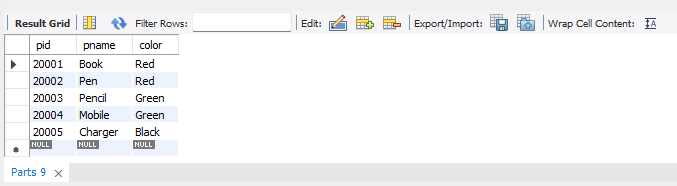
insert into Parts values(20002,"Pen", "Red");

insert into Parts values(20003,"Pencil", "Green");

insert into Parts values(20004,"Mobile", "Green");

insert into Parts values(20005,"Charger", "Black");

select \* from Parts;



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

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

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

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

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

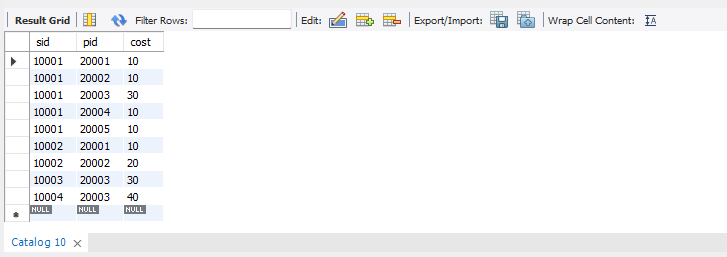
insert into Catalog values(10002,20001, 10);

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

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

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

select \* from Catalog;



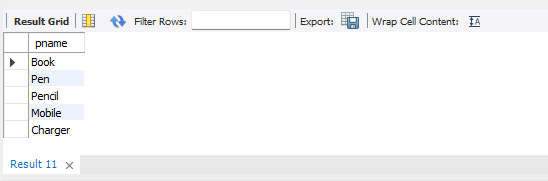
**Queries**

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

select distinct p.pname

from Parts p, Catalog c

where p.pid = c.pid;



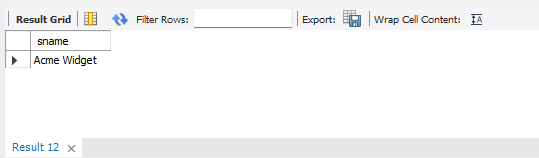
* **Find the snames of suppliers who supply every part.**

select 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 = P.pid));



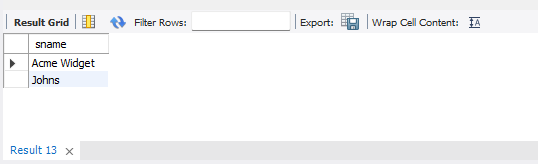
* **Find the snames of suppliers who supply every red part.**

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"));



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

select P.pname

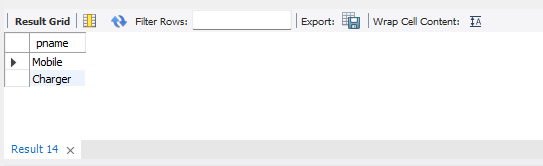
from Parts P, Catalog C, Supplier S

where P.pid = C.pid and C.sid = S.sid and S.sname = "Acme Widget"

and NOT EXISTS (select \* from Catalog C1, Supplier S1

where P.pid = C1.pid and C1.sid = S1.sid and

S1.sname != "Acme Widget");

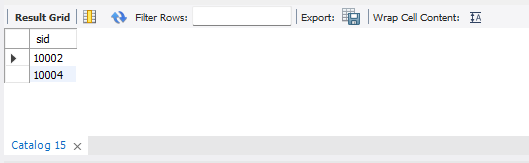


* **Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).**

select distinct C.sid from Catalog C

where C.cost > (select AVG(C1.cost)

from Catalog C1 where C1.pid = C.pid);



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

select P.pid, S.sname

from Parts P, Supplier S, Catalog C

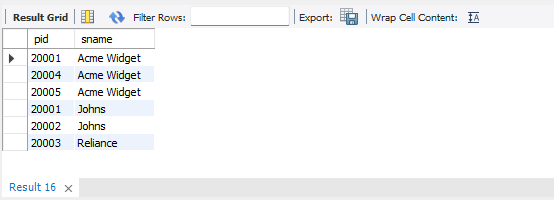
where C.pid = P.pid and

C.sid = S.sid and

C.cost = (select max(C1.cost)

from Catalog C1

where C1.pid = P.pid);



Airline Flight Database

**Question**

**(Week 8)**

FLIGHTS(flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer)

AIRCRAFT(aid: integer, aname: string, cruising\_range: integer)

CERTIFIED(eid: integer, aid: integer)

EMPLOYEES(eid: integer, ename: string, salary: integer)

Note that the Employees relation describes pilots and other kinds of employees as well; Every pilot is certified for some aircraft, and only pilots are certified to fly.

Create database table and insert appropriate data

i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruisingrange of the aircraft for which she or he is certified.

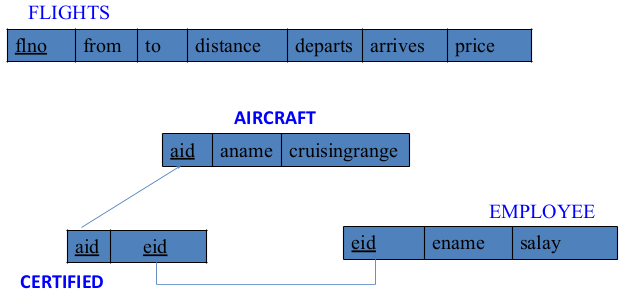
iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the Average salary of all pilots certified for this aircraft.

v. Find the names of pilots certified for some Boeing aircraft.

vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

**Schema Diagram**



**Create database**

create database flight;

use flight;

**Create table**

create table flights(

flno int,

from\_ varchar(20),

to\_ varchar(20),

distance int,

departs time,

arrives time,

price int,

PRIMARY KEY(flno)

);

create table aircraft(

aid int,

aname varchar(20),

cruisingRange int,

PRIMARY KEY(aid)

);

create table employee(

eid int,

ename varchar(20),

salary int,

PRIMARY KEY(eid)

);

create table certified(

eid int,

aid int,

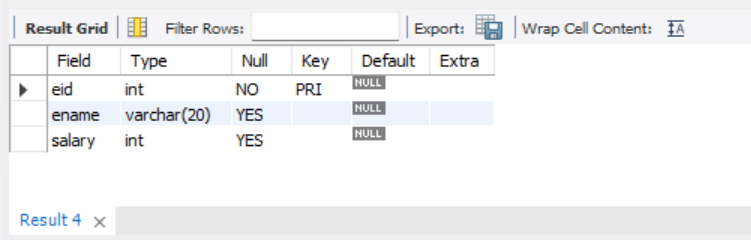
FOREIGN KEY(eid) REFERENCES employee(eid) on update cascade on delete cascade,

FOREIGN KEY(aid) REFERENCES aircraft(aid) on update cascade on delete cascade

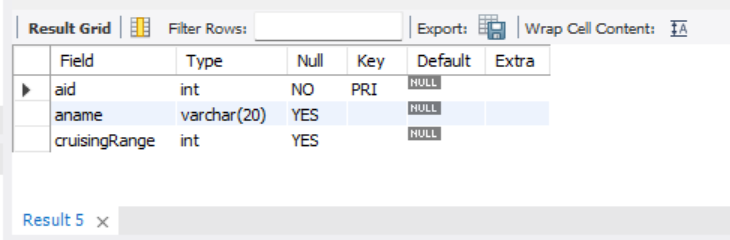
);

**Structure of the table**

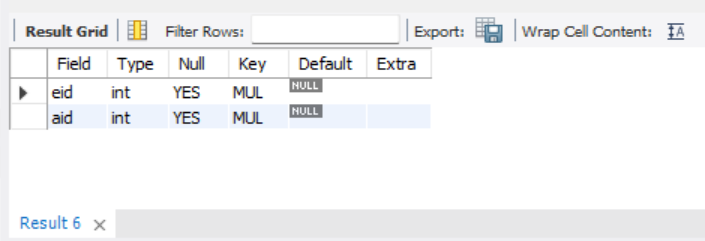
desc employee;



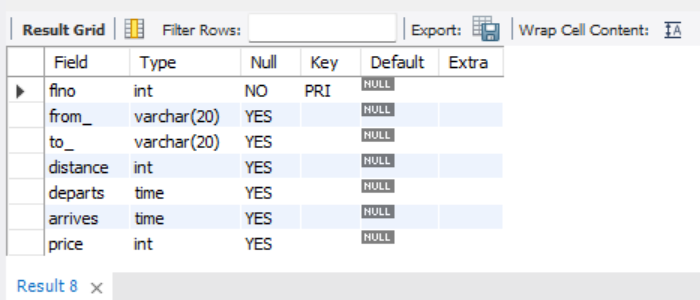
desc aircraft;



desc certified;



desc flights;



**Inserting Values to the table**

insert into employee values(101,'Avinash',50000);

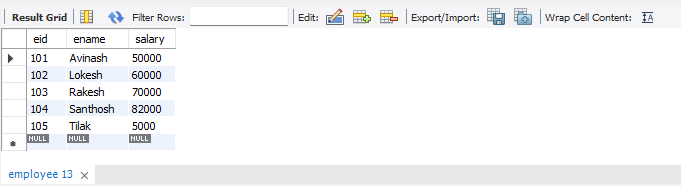
insert into employee values(102,'Lokesh',60000);

insert into employee values(103,'Rakesh',70000);

insert into employee values(104,'Santhosh',82000);

insert into employee values(105,'Tilak',5000);

select \* from employee;



insert into aircraft values(1,'Airbus',2000);

insert into aircraft values(2,'Boeing',700);

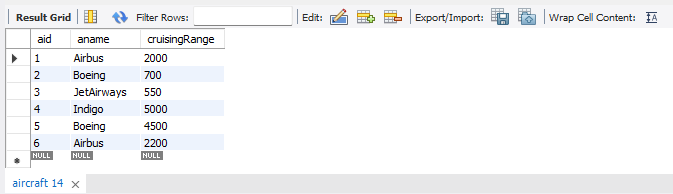
insert into aircraft values(3,'JetAirways',550);

insert into aircraft values(4,'Indigo',5000);

insert into aircraft values(5,'Boeing',4500);

insert into aircraft values(6,'Airbus',2200);

select \* from aircraft;



insert into certified values(101,2);

insert into certified values(101,4);

insert into certified values(101,5);

insert into certified values(101,6);

insert into certified values(102,1);

insert into certified values(102,3);

insert into certified values(102,5);

insert into certified values(103,2);

insert into certified values(103,3);

insert into certified values(103,5);

insert into certified values(103,6);

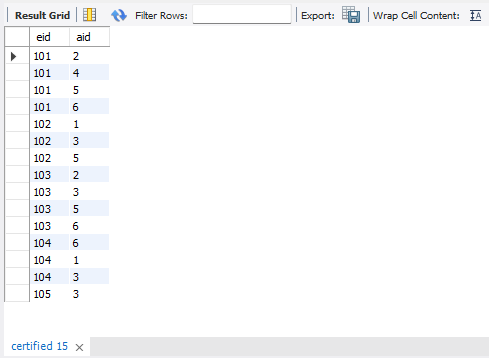
insert into certified values(104,6);

insert into certified values(104,1);

insert into certified values(104,3);

insert into certified values(105,3);

select \* from certified;



insert into flights values(1,'Bengaluru','NewDelhi',500,'06:00','09:00',5000);

insert into flights values(2,'Bengaluru','Chennai',300,'07:00','08:30',3000);

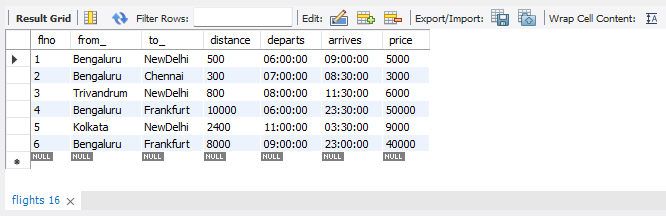
insert into flights values(3,'Trivandrum','NewDelhi',800,'08:00','11:30',6000);

insert into flights values(4,'Bengaluru','Frankfurt',10000,'06:00','23:30',50000);

insert into flights values(5,'Kolkata','NewDelhi',2400,'11:00','03:30',9000);

insert into flights values(6,'Bengaluru','Frankfurt',8000,'09:00','23:00',40000);

select \* from flights;

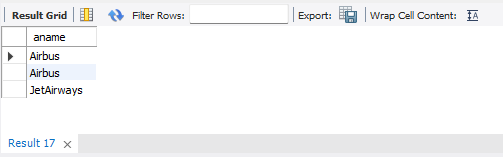


**Queries**

* **Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.**

select (a.aname) from employee e inner join certified c

on e.eid=c.eid and e.salary>80000 inner join aircraft a on a.aid=c.aid;

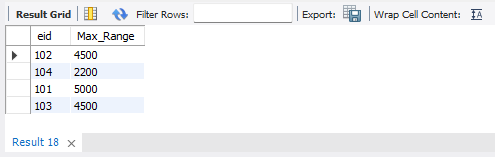


* **For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.**

select c.eid, max(a.cruisingRange) as Max\_Range

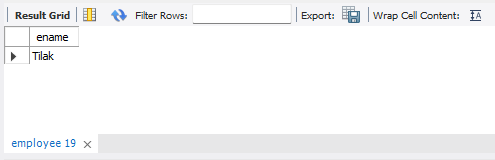
from aircraft a, certified c

where c.aid=a.aid group by c.eid having count(\*)>=3;

****

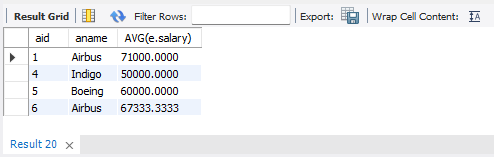
* **Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.**

select ename from employee where salary<some(select price from flights where from\_='Bengaluru' and to\_='Frankfurt');



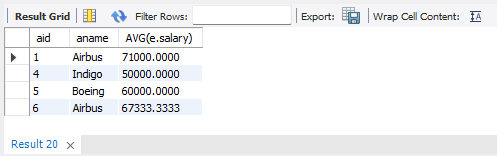
* **For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft.**

select c.aid,a.aname, AVG(e.salary) from certified c, aircraft a, employee e where a.cruisingRange>1000 and e.eid=c.eid and a.aid=c.aid group by c.aid;



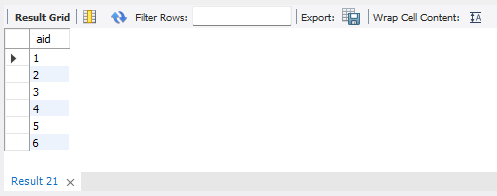
* **Find the names of pilots certified for some Boeing aircraft.**

select distinct e.ename from employee e, certified c, aircraft a where a.aid=c.aid and e.eid=c.eid and a.aname='Boeing';



* **Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.**

select a.aid from flights f, aircraft a where (f.from\_='Bengaluru' and f.to\_='NewDelhi') and f.distance<=a.cruisingRange ;



NoSQL Lab 1

**Question**

**(Week 9)**

Perform the following DB operations using MongoDB.

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

2. Insert appropriate values

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

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

5. Export the created table into local file system

6. Drop the table

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

mongodb collection.

**Create database**

db.createCollect(“Student”);

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

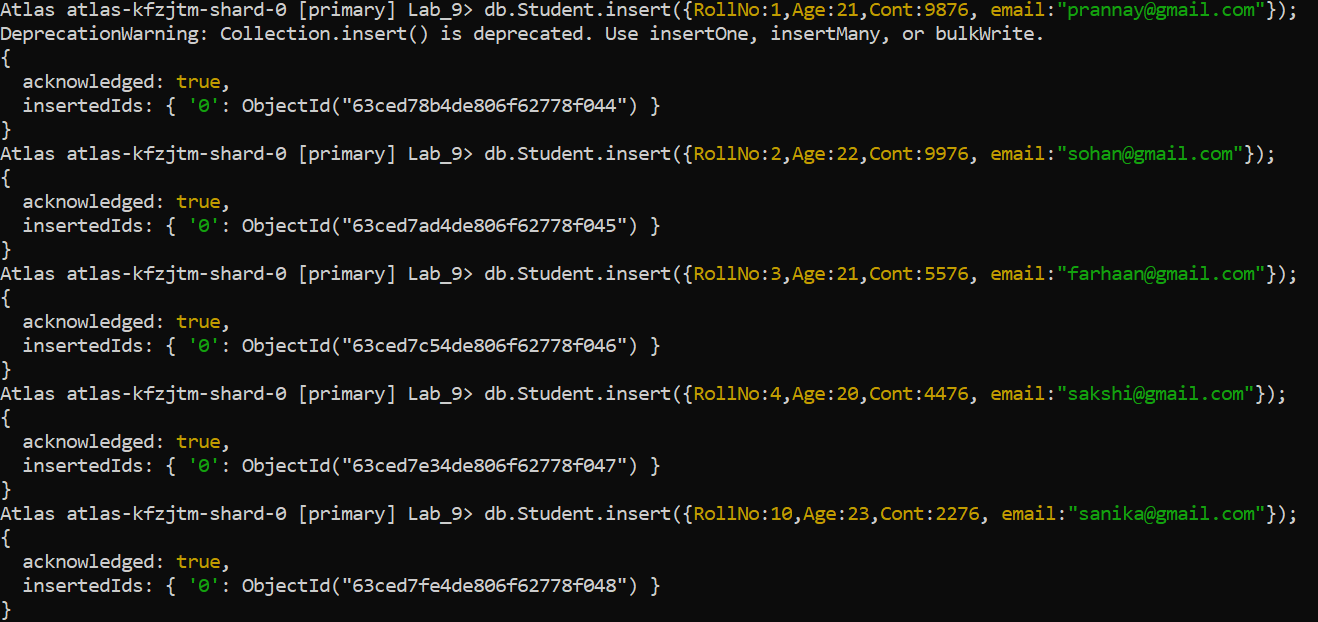
db.Student.insert({rollno:1,age:21,cont:9876,email:"prannay@gmail.com"});

db.Student.insert({rollno:2,age:22,cont:9976,email:"sohan@gmail.com"});

db.Student.insert({rollno:3,age:21,cont:5576,email:"farhaan@gmail.com"});

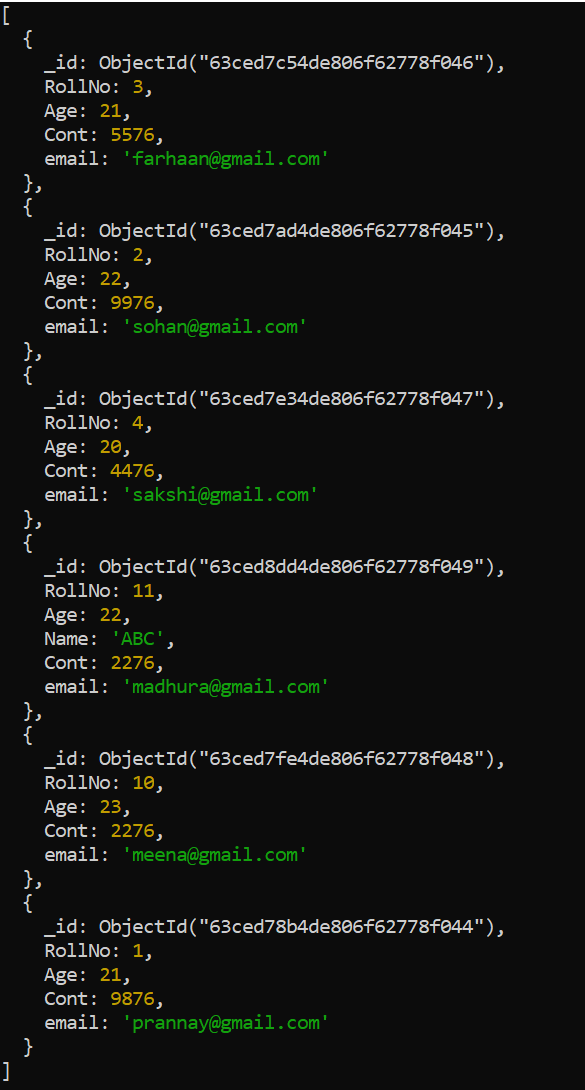
db.Student.insert({rollno:4,age:20,cont:4476,email:"sakshi@gmail.com"});

db.Student.insert({rollno:5,age:23,cont:2276,email:"[sanika@gmail.com](mailto:sanika@gmail.com)"});

****

**Structure of the table**

db.Student.find();

****

**Queries**

* **Create a database “Student” with the following attributes**

**Rollno, age, contactNo, Email-Id.**

db.createCollection("Student");



* **Insert appropriate values**

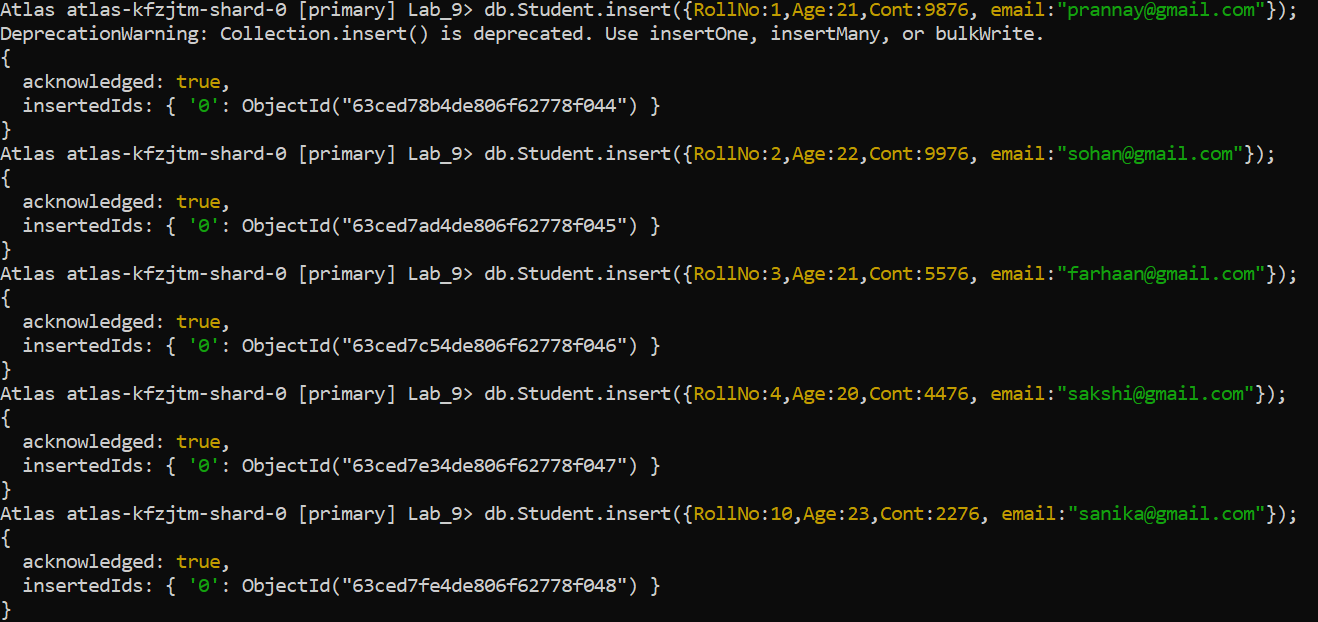
db.Student.insert({rollno:1,age:21,cont:9876,email:"prannay@gmail.com"});

db.Student.insert({rollno:2,age:22,cont:9976,email:"sohan@gmail.com"});

db.Student.insert({rollno:3,age:21,cont:5576,email:"farhaan@gmail.com"});

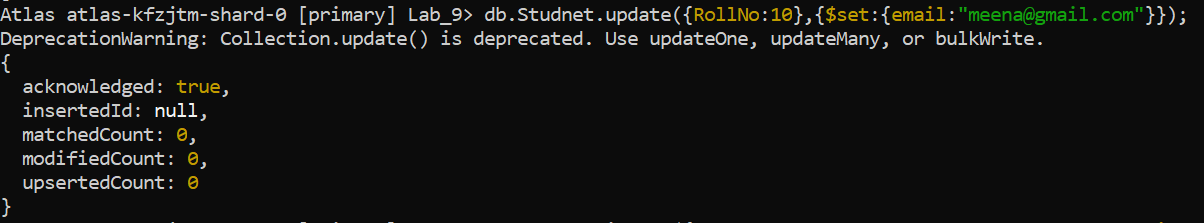
db.Student.insert({rollno:4,age:20,cont:4476,email:"sakshi@gmail.com"});

db.Student.insert({rollno:5,age:23,cont:2276,email:"sanika@gmail.com"});

****

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

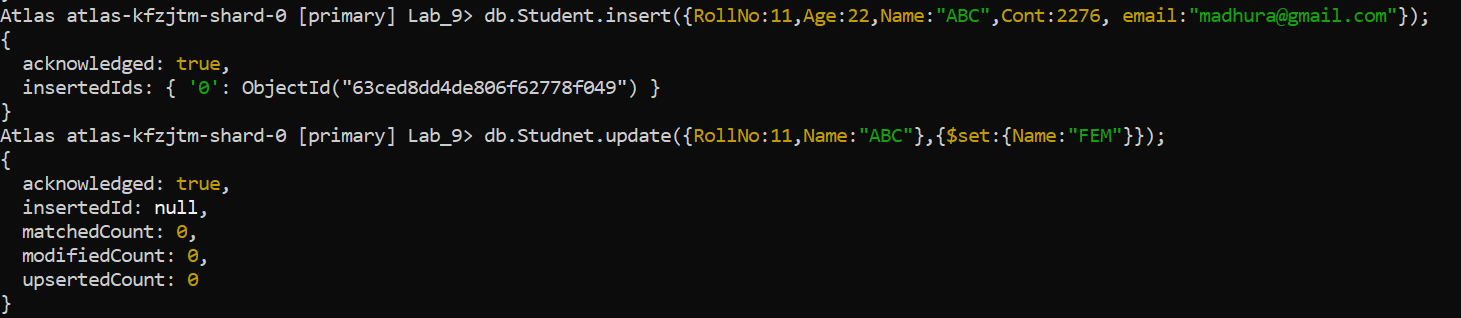
db.Student.update({rollno:5},{$set:{email:"abhinav@gmail.com"}})



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

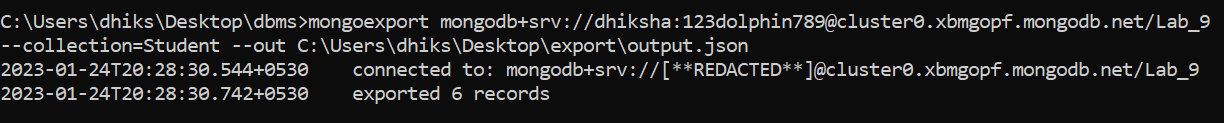
db.Student.insert({rollno:11,age:22,name:"ABC",cont:2276,email:"madhura@gmail.com"});

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



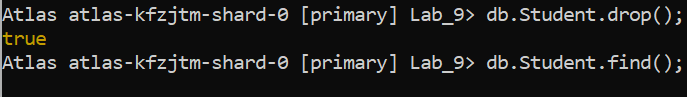
* **Export the created table into local file system**

mongoexport mongodb+srv://dhiksha:<password>@cluster0.xbmgopf.mongodb.net/Lab\_9 --collection=Student -- out C:\Users\dhiks\Desktop\export\output.json

****

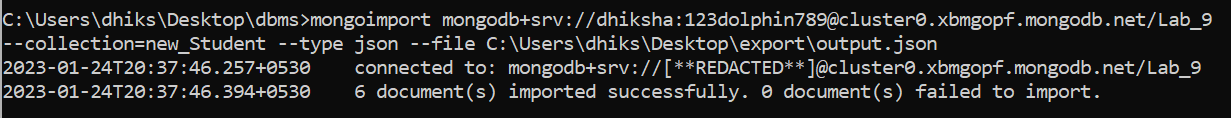
* **Drop the table**

db.Student.drop();



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

mongoimport mongodb+srv://dhiksha:<password>@cluster0.xbmgopf.mongodb.net/Lab\_9 --collection=new\_Student – type json –file C:\Users\dhiks\Desktop\export\output.json

****

NoSQL Lab 2

**Question**

**(Week 10)**

Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes Cust\_id, Acc\_Bal, Acc\_Type

2. Insert at least 5 values into the table

3. Write a query to display those records whose total account balance is greater than 1200 of account type ‘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 database**

db.createCollect(“Customer”);

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

db.Customer.insert({Cust\_id:1,Acc\_bal:2000,Acc\_type:"Z"});

db.Customer.insert({Cust\_id:2,Acc\_bal:1000,Acc\_type:"Z"});

db.Customer.insert({Cust\_id:3,Acc\_bal:1500,Acc\_type:"A"});

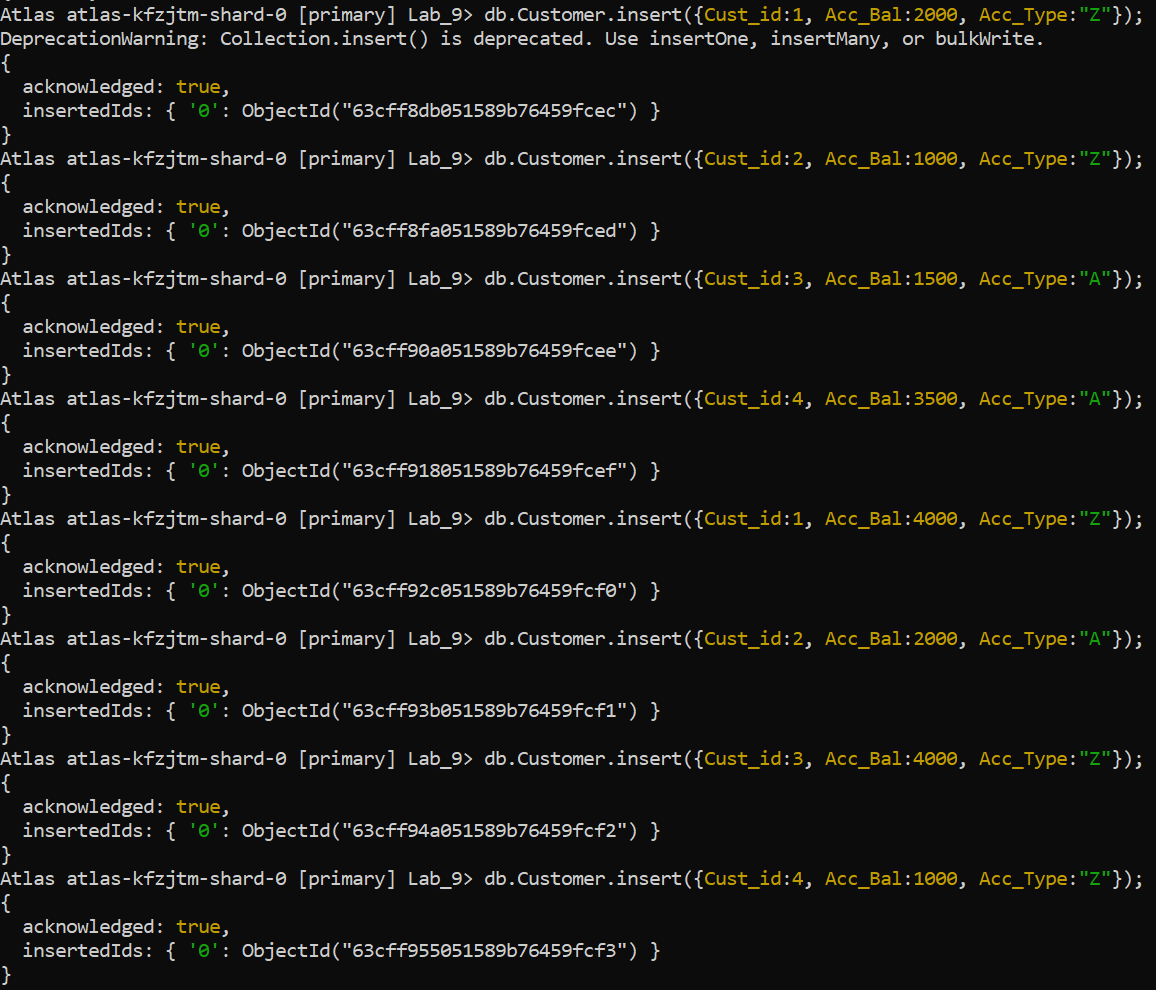
db.Customer.insert({Cust\_id:4,Acc\_bal:3500,Acc\_type:"A"});

db.Customer.insert({Cust\_id:1,Acc\_bal:4000,Acc\_type:"Z"});

db.Customer.insert({Cust\_id:2,Acc\_bal:2000,Acc\_type:"A"});

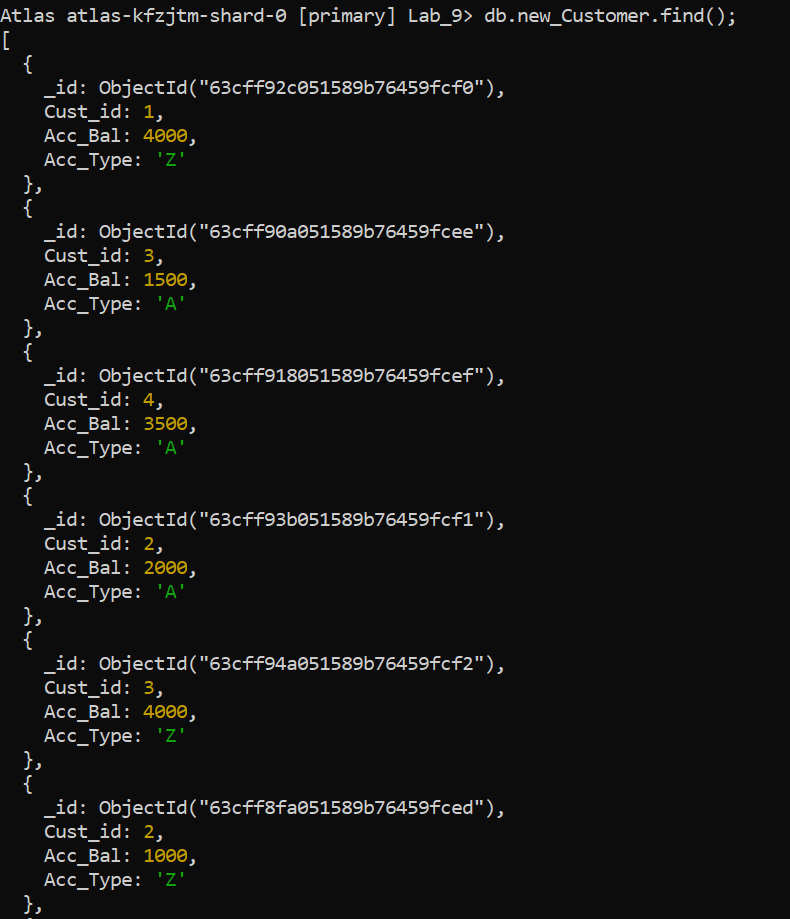
db.Customer.insert({Cust\_id:3,Acc\_bal:4000,Acc\_type:"Z"});

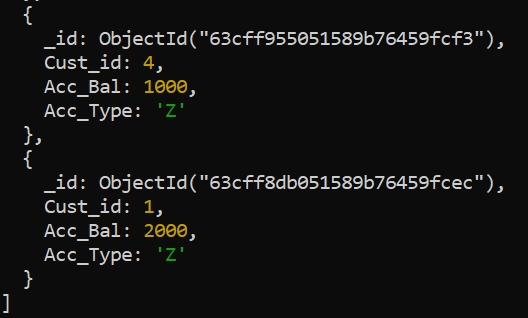
db.Customer.insert({Cust\_id:4,Acc\_bal:1000,Acc\_type:"Z"});

****

**Structure of the table**

db.Customer.find();

****

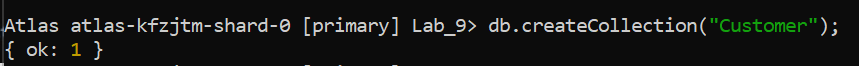
****

**Queries**

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

**Cust\_id, Acc\_Bal, Acc\_Type**

db.createCollection("Customer");



* **Insert at least 5 values into the table**

db.Customer.insert({Cust\_id:1,Acc\_bal:2000,Acc\_type:"Z"});

db.Customer.insert({Cust\_id:2,Acc\_bal:1000,Acc\_type:"Z"});

db.Customer.insert({Cust\_id:3,Acc\_bal:1500,Acc\_type:"A"});

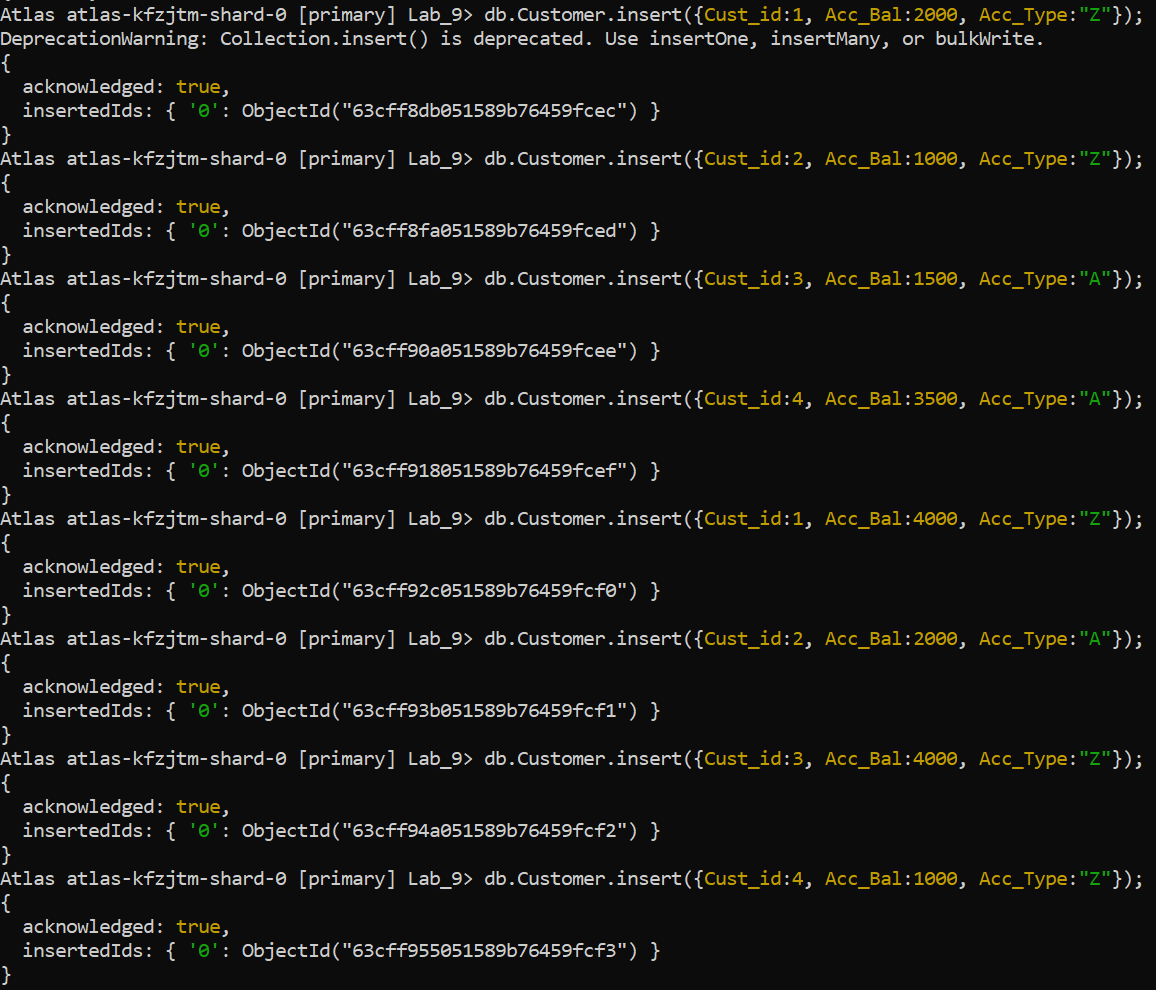
db.Customer.insert({Cust\_id:4,Acc\_bal:3500,Acc\_type:"A"});

db.Customer.insert({Cust\_id:1,Acc\_bal:4000,Acc\_type:"Z"});

db.Customer.insert({Cust\_id:2,Acc\_bal:2000,Acc\_type:"A"});

db.Customer.insert({Cust\_id:3,Acc\_bal:4000,Acc\_type:"Z"});

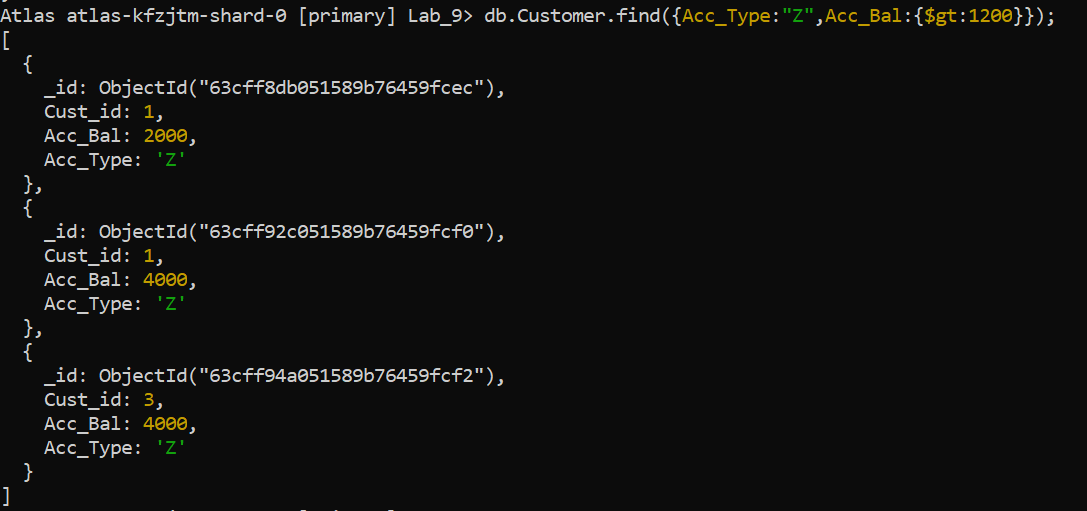
db.Customer.insert({Cust\_id:4,Acc\_bal:1000,Acc\_type:"Z"});

****

* **Write a query to display those records whose total account balance**

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

db.Customer.find({Acc\_Type:”Z”, Acc\_Bal:{$gt:1200}});

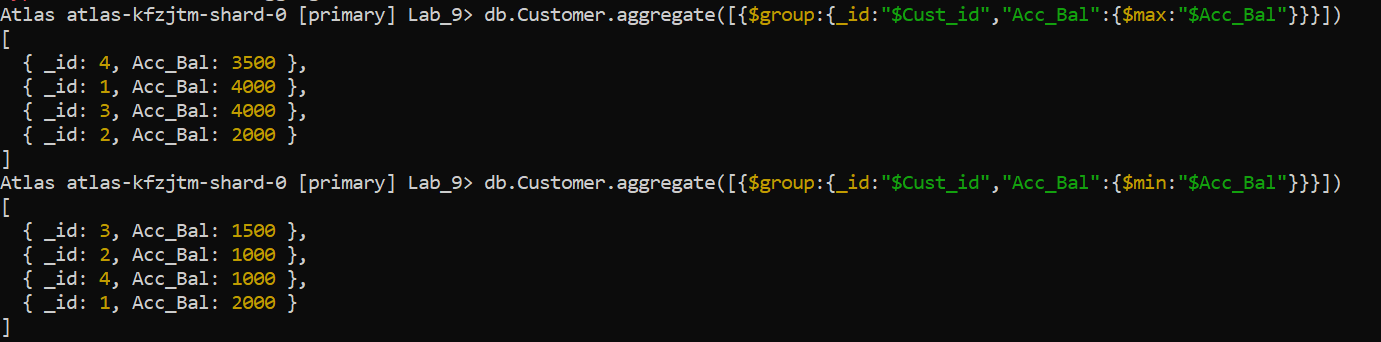


* **Determine Minimum and Maximum account balance for each**

**customer\_id.**

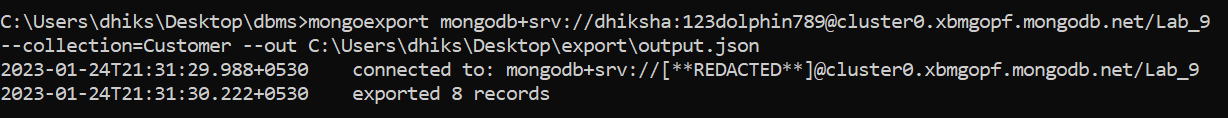
db.Customer.aggregrate([{$group:{\_id:"$Cust\_id","Acc\_Bal":{$max:"$Acc\_Bal"}}}])

db.Customer.aggregrate([{$group:{\_id:"$Cust\_id","Acc\_Bal":{$min:"$Acc\_Bal"}}}])



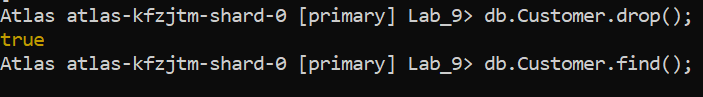
* **Export the created collection into local file system**

mongoexport mongodb+srv://dhiksha:<password>@cluster0.xbmgopf.mongodb.net/Lab\_9 --collection=Customer -- out C:\Users\dhiks\Desktop\export\output.json

****

* **Drop the table**

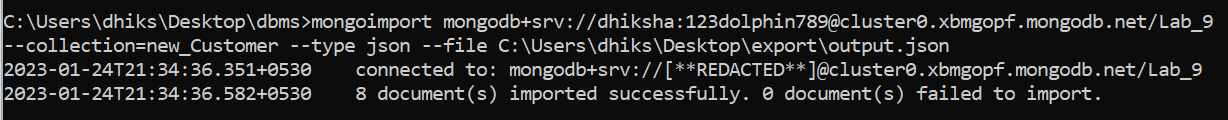
db.Customer.drop();



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

**collection.**

mongoimport mongodb+srv://dhiksha:<password>@cluster0.xbmgopf.mongodb.net/Lab\_9 --collection=new\_Customer – type json –file C:\Users\dhiks\Desktop\export\output.json

****