**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

### LAB REPORT

**on**

Database Management Systems (23CS3PCDBM)

***Submitted by***

### 

### Eashan Jain V(1BM23CS098)

***in partial fulfilment 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**

**December-2023 to Feb-2024**

**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 **Eashan Jain V(1BM23CS098),** who is a bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

Dr. Joythi S Nayak Professor HOD

Department of CSE, BMSCE

Dr. Kayarvizhy N Associate Professor

Department of CSE, BMSCE

**Index**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| 1 | 20-12-2023 | Insurance Database | 1-8 |
| 2 | 27-12-2023 | More Queries on Insurance Database | 9-12 |
| 3 | 01-01-2024 | Bank Database | 13-20 |
| 4 | 08-01-2024 | More Queries on Bank Database | 21-25 |
| 5 | 16-01-2024 | Employee Database | 26-33 |
| 6 | 22-01-2024 | More Queries on Employee Database | 34-36 |
| 7 | 29-01-2024 | Supplier Database | 37-43 |
| 8 | 05-02-2024 | NoSQL Lab 1 | 44-47 |
| 9 | 14-02-2024 | NoSQL Lab 2 | 48-50 |
| 10 | 4-03-2024 | NoSQL Lab 3 | 51-73 |

# Insurance Database

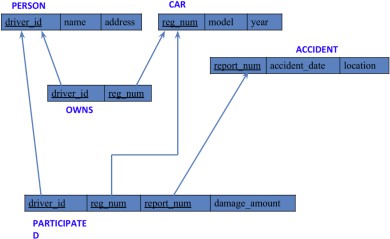
### Question (Week 1)

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

Enter at least five tuples for each relation

* Display Accident date and location
* Update the damage amount to 25000 for the car with a specific reg\_num (example 'K A031181' ) for which the accident report number was 12.
* Add a new accident to the database.
* To Do
* Display Accident date and location
* Display driver\_id who did accident with damage amount greater than or equal to Rs.25000

### Schema Diagram

****

**Create database**

**create database** insurance\_CS098;

**use** insurance\_CS098;

### Create table

**create database** insurance\_CS098;

**use** insurance\_CS098;

**create table** person\_CS098( driver\_id varchar(3) primary **key**, **name** varchar(20) **not null**, address varchar(100)

);

**create table** car\_CS098( reg\_no char(8) primary **key**, **model** varchar(20),

**year** int(4) **not null**

);

**create table** accident\_CS098(

report\_no int(4) primary **key**, accident\_date date,

location varchar(100)

);

**create table** owns\_CS098(

driver\_id varchar(3), reg\_no char(8),

foreign **key**(driver\_id) **references** person\_CS098(driver\_id), foreign **key**(reg\_no) **references** car\_CS098(reg\_no)

);

**create table** participated\_CS098( driver\_id varchar(3),

reg\_no char(8), report\_no int(4), damage\_amt int,

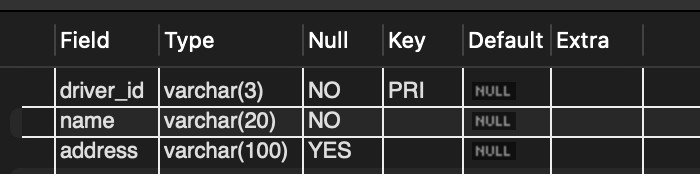
foreign **key**(driver\_id) **references** person\_CS098(driver\_id), foreign **key**(reg\_no) **references** car\_CS098(reg\_no),

foreign **key** (report\_no) **references** accident\_CS098(report\_no)

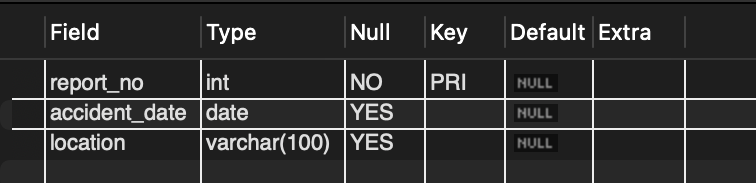
);

### Structure of the table

desc person\_CS098;



desc accident\_CS098;



desc participated\_CS098;



desc car\_CS098;



desc owns\_CS098;



### Inserting Values into the table

**insert into** person\_CS098 **values**

(**"A01"**, **"Richard"**, **"Sri Nagar"**),

(**"A02"**, **"Pradeep"**, **"Raj Nagar"**),

(**"A03"**, **"Smith"**, **"Ashok Nagar"**),

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

(**"A05"**, **"John"**, **"Hanu Nagar"**);

**insert into** car\_CS098 **values**

(**"KA052250"**, **"Indica"**, 1990),

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

(**"KA095477"**, **"Toyota"**, 1998),

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

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

**insert into** owns\_CS098 **values**

(**"A01"**, **"KA052250"**);

**insert into** owns\_CS098 **values**

(**"A02"**, **"KA031181"**);

**insert into** owns\_CS098 **values**

(**"A03"**, **"KA095477"**);

**insert into** owns\_CS098 **values**

(**"A04"**, **"KA053408"**);

**insert into** owns\_CS098 **values**

(**"A05"**, **"KA041702"**);

**insert into** accident\_CS098 **values**

(11, **"01-01-03"**, **"Mysore Rd"**),

(12, **"02-02-04"**, **"SE Circle"**),

(13, **"21-01-03"**, **"Bull Temple Rd"**),

(14, **"17-02-08"**, **"Mysore Rd"**),

(15, **"04-03-05"**, **"KR Puram"**);

**insert into** participated\_CS098 **values**

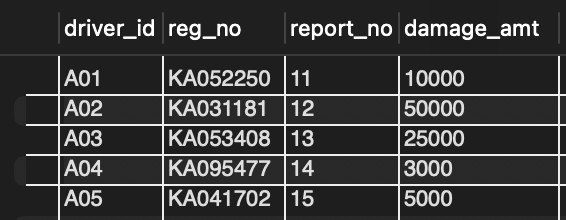
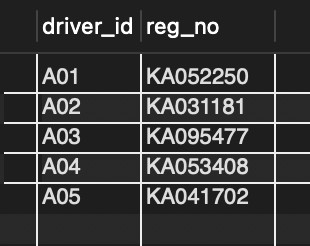
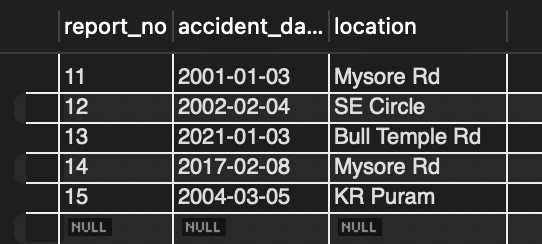
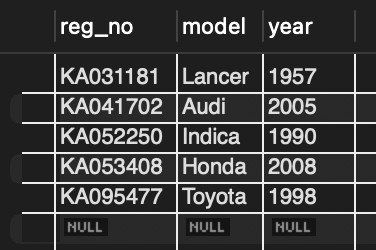
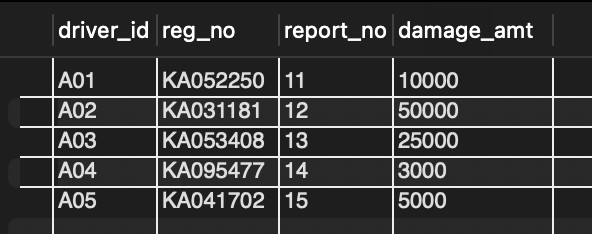
(**"A01"**, **"KA052250"**, 11, 10000), (**"A02"**, **"KA031181"**, 12, 50000),

(**"A03"**, **"KA053408"**, 13, 25000),

(**"A04"**, **"KA095477"**, 14, 3000),

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

**select** \* **from** person\_CS098; **select** \* **from** car\_CS098; **select** \* **from** accident\_CS098; **select** \* **from** owns\_CS098;

**select** \* **from** participated\_CS098;

### Queries

###### - Update the damage amount to 25000 for the car with a specific reg-num (example 'KA031181' ) for which the accident report number was 12.

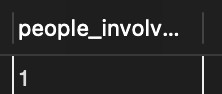
**update** participated\_CS098 **set** damage\_amt = 25000 **where** reg\_no = **"KA031181" and** report\_no = 12;



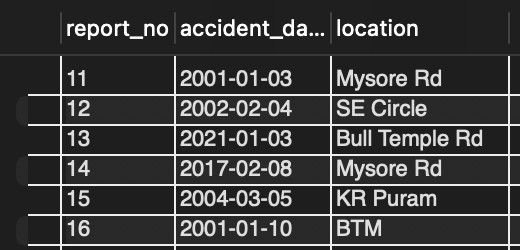
###### Find the total number of people who owned cars that were involved in accidents in 2008.

**select count**(driver\_id) people\_involved **from** participated\_CS098, accident\_CS098 **where**

participated\_CS098.report\_no = accident\_CS098.report\_no **and** accident\_CS098.accident\_date **like "%-08"**;



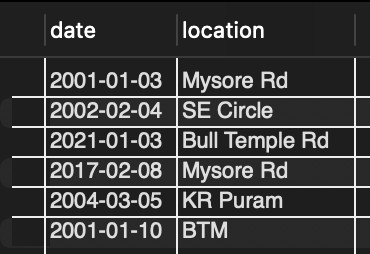
###### Add a new accident to the database.

**insert into** accident\_CS098 **values** (16, **"01-01-10"**, **"BTM"**); **select** \* **from** accident\_CS098;

**TO DO:**

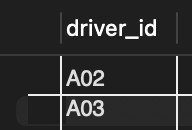
#### DISPLAY ACCIDENT DATE AND LOCATION

**select** accident\_date **as** date, location **from** accident\_CS098;



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

**select** participated\_CS098.driver\_id **as** driver\_id **from** accident\_CS098, participated\_CS098 **where** accident\_CS098.report\_no = participated\_CS098.report\_no **and** participated\_CS098.damage\_amt >= 25000;

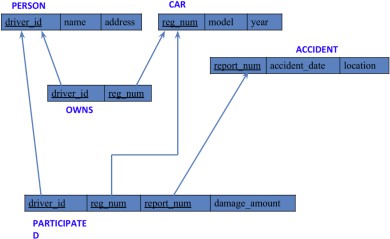


# More Queries on Insurance Database

### Question (Week 2)

* PERSON (driver\_id: String, name: String, address: String)
* CAR (reg\_num: String, model: String, year: int)
* ACCIDENT (report\_num: int, accident\_date: date, location: String)
* OWNS (driver\_id: String, reg\_num: String)
* PARTICIPATED (driver\_id: String,reg\_num: String, report\_num: int, damage\_amount: int)
* 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.

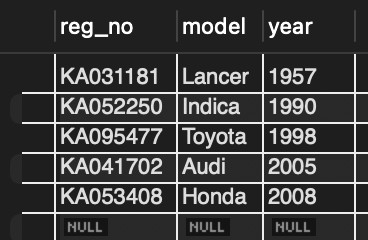
### Schema Diagram

****

**Queries**

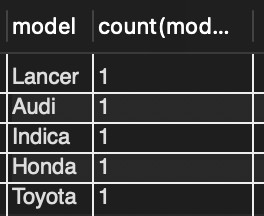
###### Display the entire CAR relation in the ascending order of manufacturing year.

select \* from car\_CS098 order by year asc;



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

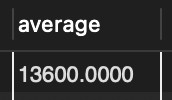
**select model**, **count**(**model**) **from** participated\_CS098, car\_CS098 **where** participated\_CS098.reg\_no = car\_CS098.reg\_no **group by model**;



**TO DO:**

#### FIND THE AVERAGE DAMAGE AMOUNT

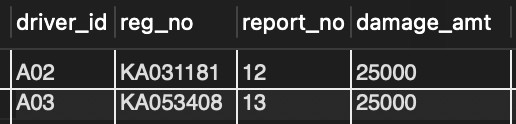
**select avg**(damage\_amt) **as** average **from** participated\_CS098;



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

**delete from** participated\_CS098 **where** damage\_amt < (**select** \* **from** (**select avg**(damage\_amt) **from**

participated\_CS098) **as** average);



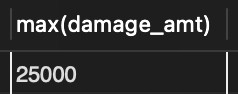
##### LIST THE NAME OF DRIVERS WHOSE DAMAGE IS GREATER THAN THE AVERAGE DAMAGE AMOUNT.

**select name from** person\_CS098, participated\_CS098 **where** person\_CS098.driver\_id = participated\_CS098.driver\_id **and** participated\_CS098.damage\_amt > (**select avg**(damage\_amt) **from** participated\_CS098);



##### FIND MAXIMUM DAMAGE AMOUNT.

**select max**(damage\_amt) **from** participated\_CS098;



# Bank Database

### Question (Week 3)

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

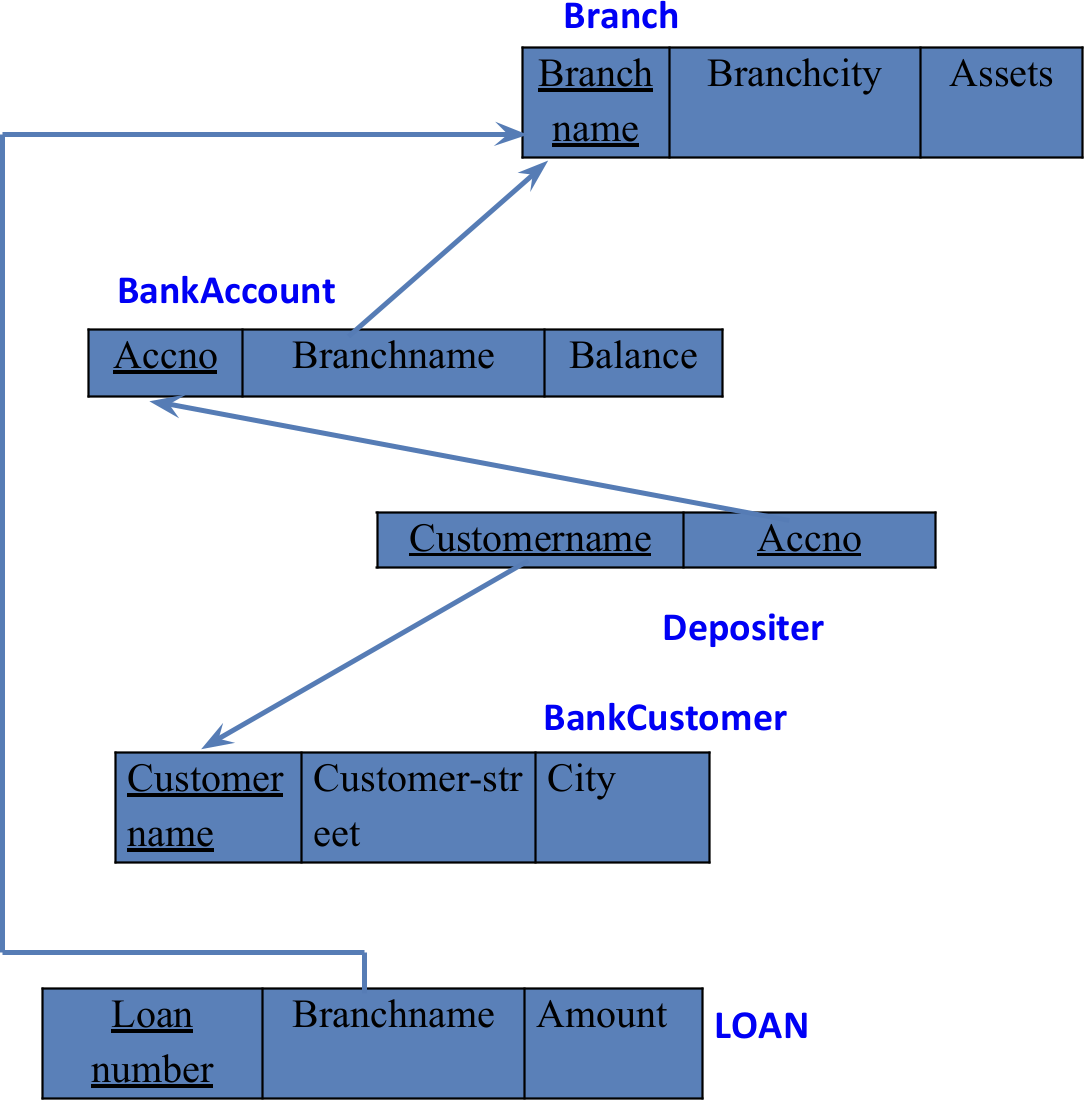
Depositer(customer-name: String, accno: int)

* LOAN (loan-number: int, branch-name: String, amount: real)
* Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation.
* Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
* Find all the customers who have at least two accounts at the same branch (ex.

SBI\_ResidencyRoad).

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

### Schema Diagram

****

**Create database**

**create database** bank\_CS098;

**use** bank\_CS098;

### Create table

**create table** branch\_CS098(

branch\_name varchar(20) primary **key**, branch\_city varchar(20),

assets float

);

**create table** bank\_account\_CS098(

acc\_no int primary **key**, branch\_name varchar(20),

balance float,

foreign **key**(branch\_name) **references** branch\_CS098(branch\_name)

);

**create table** deposits\_CS098(

customer\_name varchar(20), acc\_no int,

foreign **key**(acc\_no) **references** bank\_account\_CS098(acc\_no),

foreign **key**(customer\_name) **references** bank\_customer\_CS098(customer\_name)

);

**create table** bank\_customer\_CS098( customer\_name varchar(20) primary **key**,

customer\_street varchar(50), city varchar(15)

);

**create table** loans\_CS098(

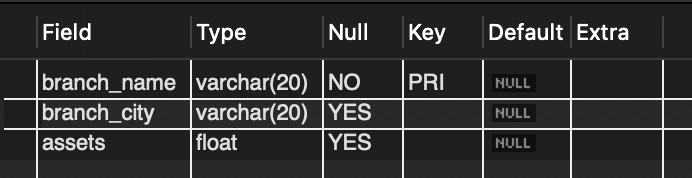
loan\_no int primary **key**, branch\_name varchar(20), amt float,

foreign **key**(branch\_name) **references** branch\_CS098(branch\_name)

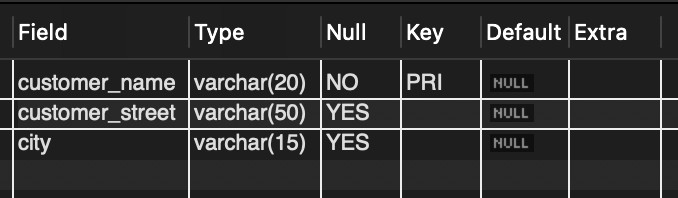
);

### Structure of the table

desc branch\_CS098;



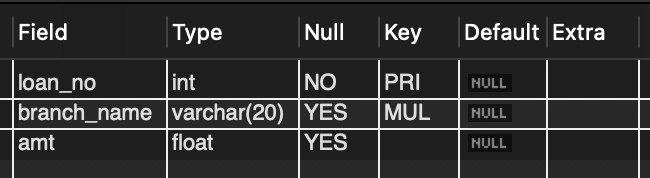
desc bank\_customer\_CS098;



desc deposits\_CS098;



desc loans\_CS098;



desc bank\_account\_CS098;



### Inserting Values to the table

insert into branch\_CS098 values ("Chamrajpet","Banglore",50000), ("ResideRoad","Banglore",10000), ("ShivaRoad","Bombay",20000),

("Parliament","Delhi",10000),

("JMantar","Delhi",20000);

insert into bank\_account\_CS098 values (1,"Chamrajpet",2000),

(2,"ResideRoad",5000),

(3,"ShivaRoad",6000),

(4,"Parliament",9000),

(5,"JMantar",8000),

(6,"ShivaRoad",4000),

(8,"ResideRoad",4000),

(9,"Parliament",3000),

(10,"ResideRoad",5000),

(11,"JMantar",2000);

insert into bank\_customer\_CS098 values ("Avinash","BulTemple","Banglore"),

("Dinesh","Banrgutta","Banglore"), ("Mohan","National college","Banglore"), ("Nikhil","Akbar road","Delhi"),

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

insert into deposits\_CS098 values ("Avinash",1),

("Dinesh",2),

("Nikhil",4),

("Ravi",5),

("Avinash",8),

("Nikhil",9),

("Dinesh",10),

("Nikhil",11);

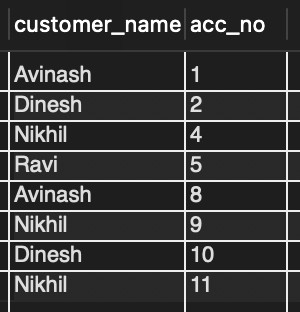
insert into loans\_CS098 values (1,"Chamrajpet",1000),

(2,"ResideRoad",2000),

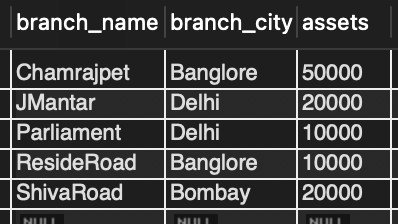
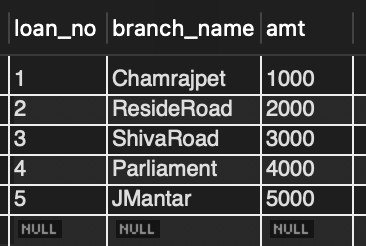
(3,"ShivaRoad",3000),

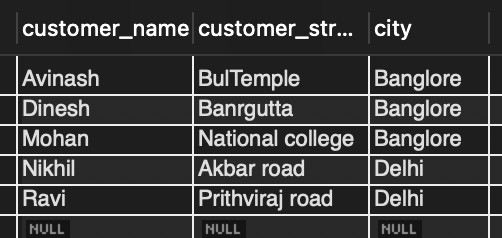
(4,"Parliament",4000),

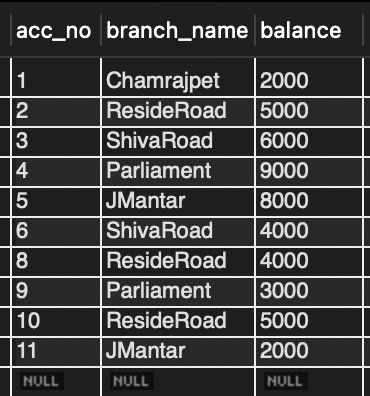
(5,"JMantar",5000);

**select** \* **from** branch\_CS098; **select** \* **from** deposits\_CS098; **select** \* **from** loans\_CS098;

**select** \* **from** bank\_customer\_CS098;

**select** \* **from** bank\_account\_CS098;





### Queries

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

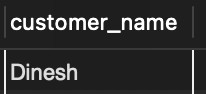
**alter table** branch\_CS098 **rename column** assets **to** assets\_in\_lks;

**select** branch\_name, assets\_in\_lks **from** branch\_CS098;



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

**select** d.customer\_name **from** deposits\_CS098 d, bank\_account\_CS098 b **where** b.branch\_name=**'ResideRoad' and** d.acc\_no=b.acc\_no **group by** d.customer\_name **having count**(d.acc\_no)>=2;



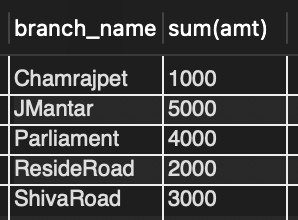
###### Create a view which gives each branch the sum of the amount of all the loans at the branch.

**create view** loansum **as** (

**select** branch\_name, **sum**(amt) **from** loans\_CS098 **group by** branch\_name

);

**select** \* **from** loansum;



# More Queries on Bank Database

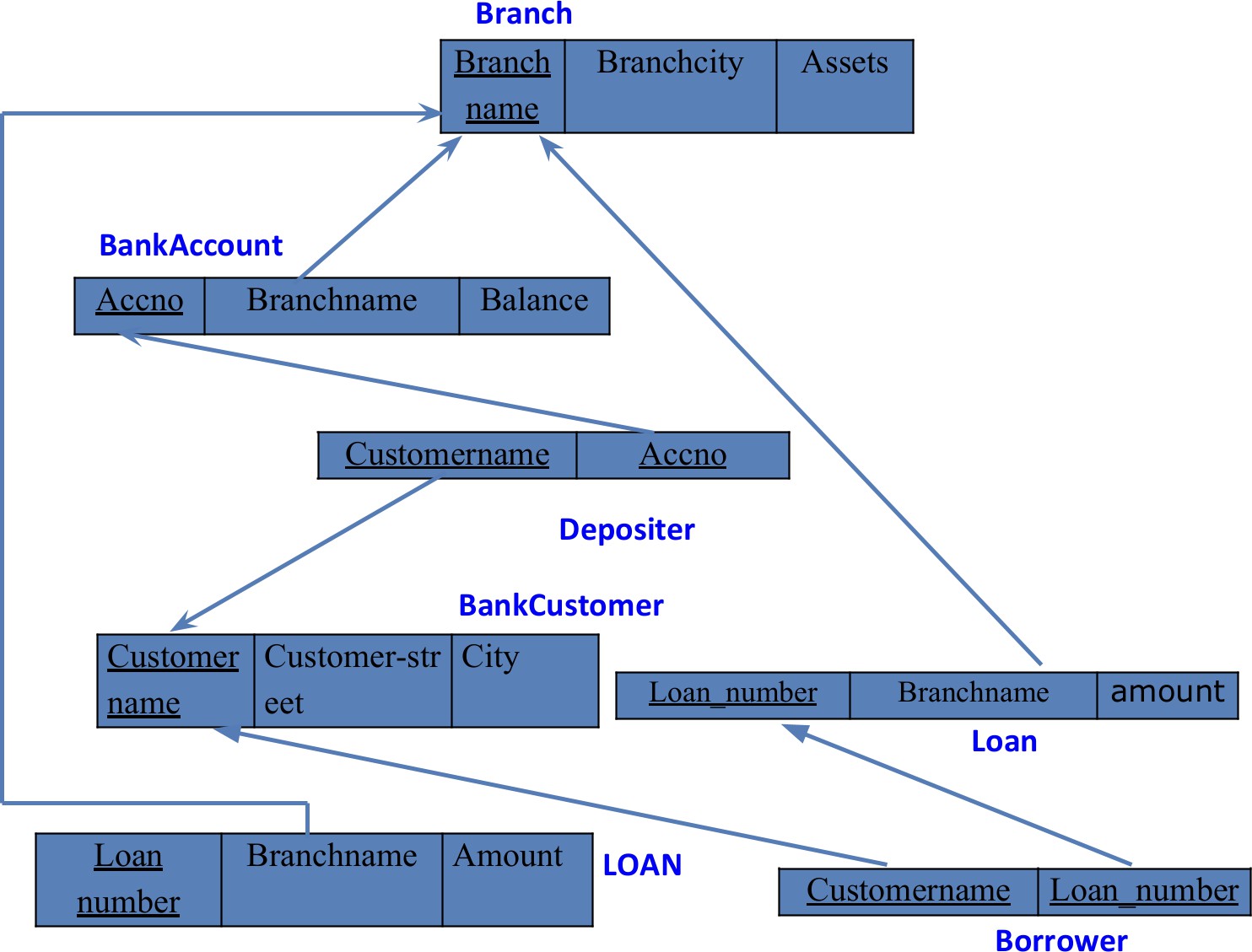
### Question (Week 4)

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

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

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

****

**Creating Table:**

**create table** borrower\_CS098( customer\_name varchar(20),

loan\_no int,

foreign **key**(customer\_name) **references** bank\_customer\_CS098(customer\_name), foreign **key**(loan\_no) **references** loans\_CS098(loan\_no)

);

### Inserting values:

**insert into** branch\_CS098 **values** (**"SBI\_MantriMarg"**, **"Delhi"**, 200000); **insert into** bank\_account\_CS098 **values** (12, **"SBI\_MantriMarg"**, 2000); **insert into** deposits\_CS098 **values**(**"Nikhil"**, 12);

**insert into** borrower\_CS098 **values**

(**"Avinash"**, 1),

(**"Dinesh"**, 2),

(**"Mohan"**, 3),

(**"Nikhil"**, 4),

(**"Ravi"**, 5);

### Queries

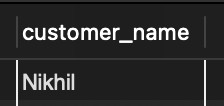
###### Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).

**select** d.customer\_name **from** branch\_CS098 b, deposits\_CS098 d, bank\_account\_CS098 ba **where**

b.branch\_city=**'Delhi' and** d.acc\_no=ba.acc\_no **and** b.branch\_name=ba.branch\_name

**group by** d.customer\_name **having count**(**distinct** b.branch\_name)= (**select count**(**distinct**

b.branch\_name) **from** branch\_CS098 b **where** b.branch\_city=**'Delhi'**;



* **Find all customers who have a loan at the bank but do not have an account. select** customer\_name, loans\_CS098.loan\_no

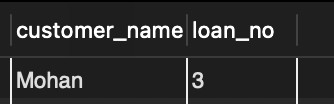
**from** (borrower\_CS098 **right outer join** loans\_CS098

**on** loans\_CS098.loan\_no = borrower\_CS098.loan\_no)

**where** customer\_name **not in** (**select** customer\_name

**from** deposits\_CS098, bank\_account\_CS098 **where** deposits\_CS098.acc\_no = bank\_account\_CS098.acc\_no

**group by** customer\_name, branch\_name);

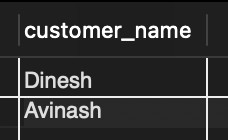


###### Find all customers who have both an account and a loan at the Bangalore branch.

**select distinct** customer\_name **from** deposits\_CS098

**where** customer\_name **in** (**select** deposits\_CS098.customer\_name **from** branch\_CS098, bank\_account\_CS098, deposits\_CS098

**where** branch\_CS098.branch\_city = **"Banglore" and** branch\_CS098.branch\_name = bank\_account\_CS098.branch\_name **and** bank\_account\_CS098.acc\_no = deposits\_CS098.acc\_no) **and** customer\_name **in** (**select** customer\_name **from** borrower\_CS098, loans\_CS098 **where** branch\_name **in** (**select** branch\_name **from** branch\_CS098 **where** branch\_city = **"Banglore"**));



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

select branch\_name from branch\_CS098 where assets\_in\_lks > all(select assets\_in\_lks from branch\_CS098 where branch\_city = "Banglore");



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

update bank\_account\_CS098 set balance = 1.05\*balance;

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

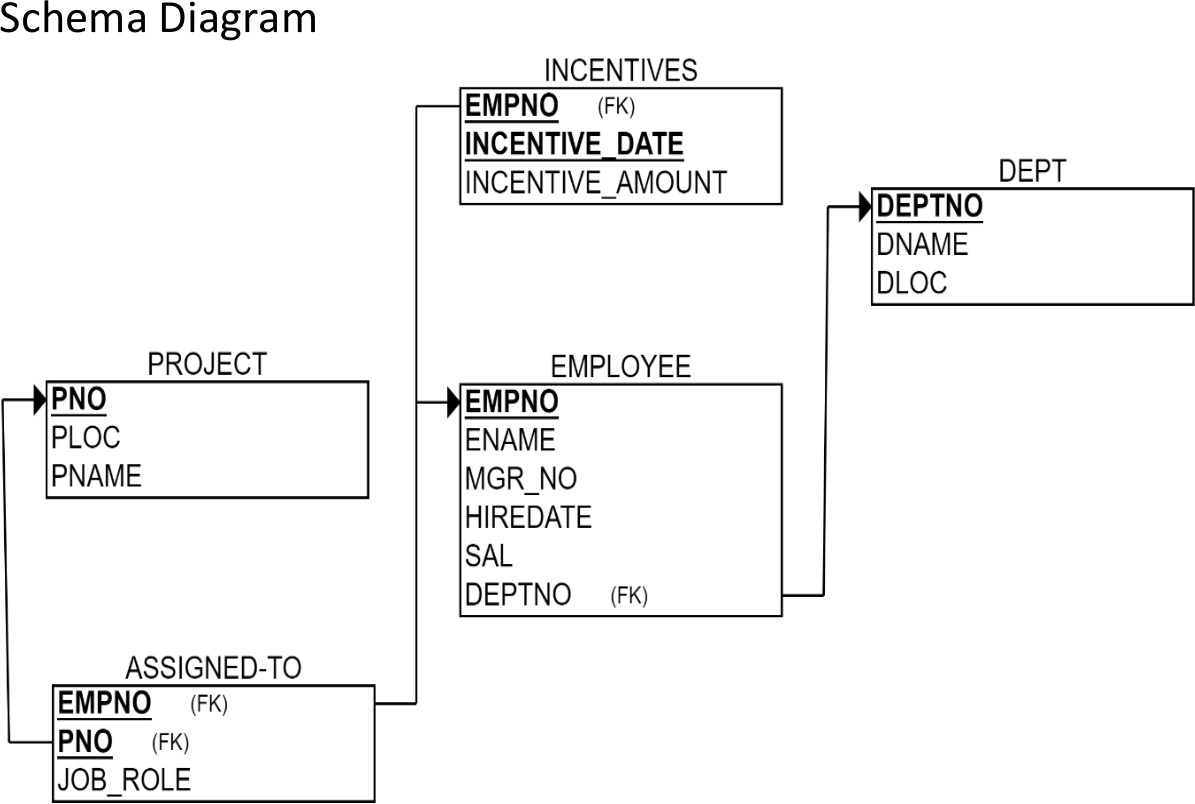
delete from bank\_account\_CS098 where branch\_name in (select branch\_name from branch\_CS098 where branch\_city = "Bombay");

# 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** employee\_database\_CS098;

**use** employee\_database\_CS098;

### Create table

**create table** project\_CS098(

pno int primary **key**, ploc varchar(20), pname varchar(20)

);

**create table** dept\_CS098(

deptno int primary **key**, dname varchar(30), dloc varchar(30)

);

**create table** employee\_CS098( empno int primary **key**, ename varchar(20), mgr\_no int,

hiredate date, sal **double**, deptno int,

foreign **key**(deptno) **references** dept\_CS098(deptno)

);

**create table** assigned\_to\_CS098( empno int primary **key**, pno int,

job\_role varchar(20),

foreign **key**(empno) **references** employee\_CS098(empno), foreign **key**(pno) **references** project\_CS098(pno)

);

**create table** incentives\_CS098( empno int,

incentive\_date date primary **key**, incentive\_amount **double**,

foreign **key**(empno) **references** employee\_CS098(empno)

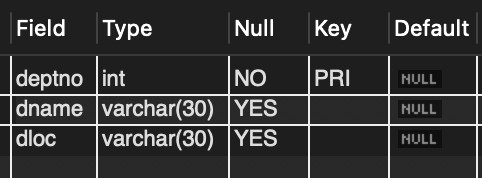
);

### Structure of the table

desc project;

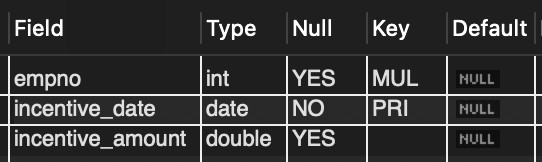


desc dept;

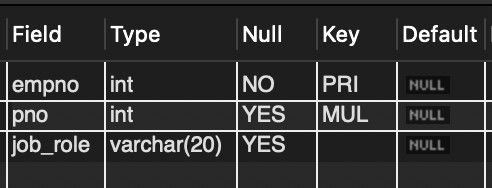
 \ desc employee;



desc incentives;



desc assigned\_to;



### Inserting Values to the table

**insert into** project\_CS098 **values**

(1,**"bengaluru"**,**"abcd"**),

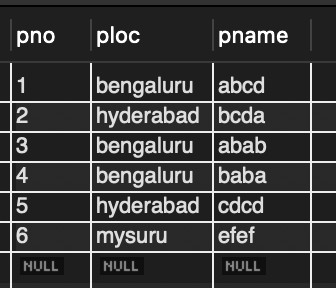
(2,**"hyderabad"**,**"bcda"**),

(3,**"bengaluru"**,**"abab"**),

(4,**"bengaluru"**,**"baba"**),

(5,**"hyderabad"**,**"cdcd"**),

(6, **"mysuru"**, **"efef"**); **select** \* **from** project\_CS098;



**insert into** dept\_CS098 **values**

(1,**"cse"**,**"bengaluru"**),

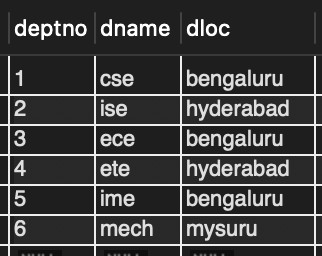
(2,**"ise"**,**"hyderabad"**),

(3,**"ece"**,**"bengaluru"**),

(4,**"ete"**,**"hyderabad"**),

(5,**"ime"**,**"bengaluru"**),

(6, **"mech"**, **"mysuru"**); **select** \* **from** dept\_CS098;



**insert into** employee\_CS098 **values**

(1,**"a"**,**null**,**"2023-11-9"**,70000,1),

(2,**"b"**,2,**"2023-8-9"**,70000,1),

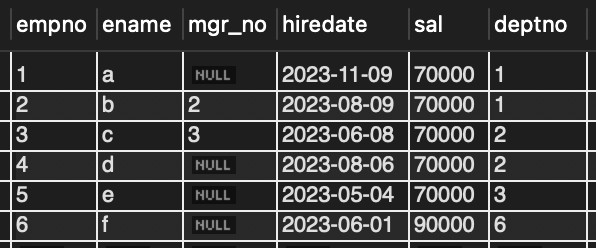
(3,**"c"**,3,**"2023-6-8"**,70000,2),

(4,**"d"**,**null**,**"2023-8-6"**,70000,2),

(5,**"e"**,**null**,**"2023-5-4"**,70000,3),

(6, **"f"**, **null**, **"2023-6-1"**, 90000, 6);

**select** \* **from** employee\_CS098;



**insert into** incentives\_CS098 **values**

(1,**"2023-12-9"**,10000),

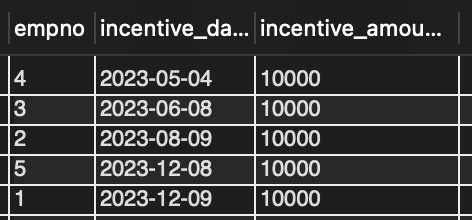
(2,**"2023-8-9"**,10000),

(3,**"2023-6-8"**,10000),

(4,**"2023-5-4"**,10000),

(5,**"2023-12-8"**,10000);

**select** \* **from** incentives\_CS098;



**insert into** assigned\_to\_CS098 **values**

(1,1, **"employee"**),

(2,1, **"manager"**),

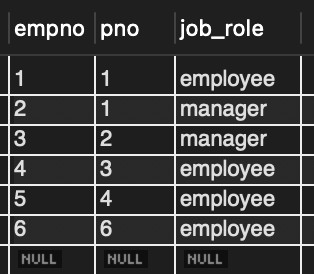
(3,2, **"manager"**),

(4,3, **"employee"**),

(5,4, **"employee"**),

(6, 6, **"employee"**);

**select** \* **from** assigned\_to\_CS098;

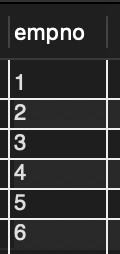


### Queries

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

**select** assigned\_to\_CS098.empno **from** assigned\_to\_CS098, project\_CS098

**where** assigned\_to\_CS098.pno = project\_CS098.pno **and** project\_CS098.ploc **in** (**"bengaluru"**, **"mysuru"**, **"hyderabad"**);



* **Get Employee ID’s of those employees who didn’t receive incentives** select empno from employee\_CS098 where empno not in (select empno from incentives\_CS098);



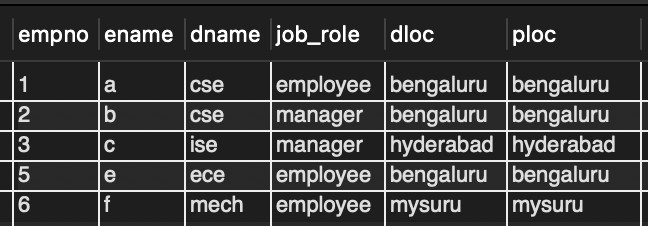
###### Write a SQL query to find the employees name, number, dept, job\_role, department location and project location who are working for a project location same as his/her department location.

**select** employee\_CS098.empno, ename, dname, job\_role, dloc, ploc

**from** employee\_CS098, assigned\_to\_CS098, project\_CS098, dept\_CS098

**where** ploc = dloc **and** assigned\_to\_CS098.empno = employee\_CS098.empno

**and** employee\_CS098.deptno = dept\_CS098.deptno **and** project\_CS098.pno = assigned\_to\_CS098.pno;

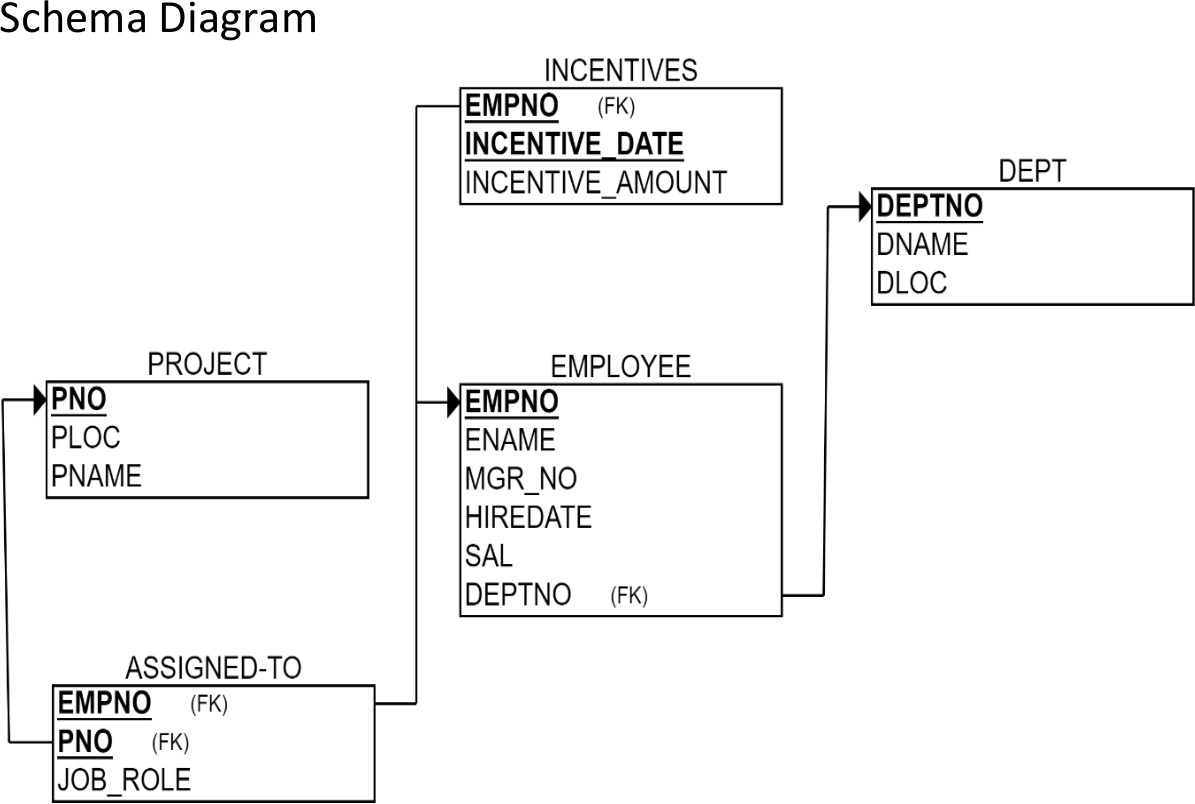


# More Queries on Employee Database

### 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

****

**Queries**

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

**select** e1.ename

**from** employee\_CS098 e1, employee\_CS098 e2

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

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

**from** employee\_CS098 e1, employee\_CS098 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\_CS098 m

**where** m.empno **in**

(**select** mgr\_no **from** employee\_CS098)

**and** m.sal>(**select avg**(n.sal) **from** employee\_CS098 n

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



* + **Find the employee details who got second maximum incentive in January 2019.** select \* from employee\_CS098 where empno=

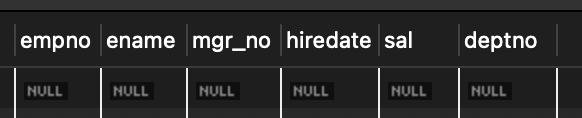
(select i.empno from incentives\_CS098 i

where i.incentive\_amount= (select max(n.incentive\_amount) from incentives\_CS098 n where n.incentive\_amount < (select max(inc.incentive\_amount) from incentives\_CS098

inc

where inc.incentive\_date between 2023-01-01 and 2023-12-31) and incentive\_date

between 2023-01-01 and 2023-12-31));

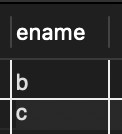


###### Display those employees who are working in the same department where his manager is working.

**select** e2.ename

**from** employee\_CS098 e1, employee\_CS098 e2

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

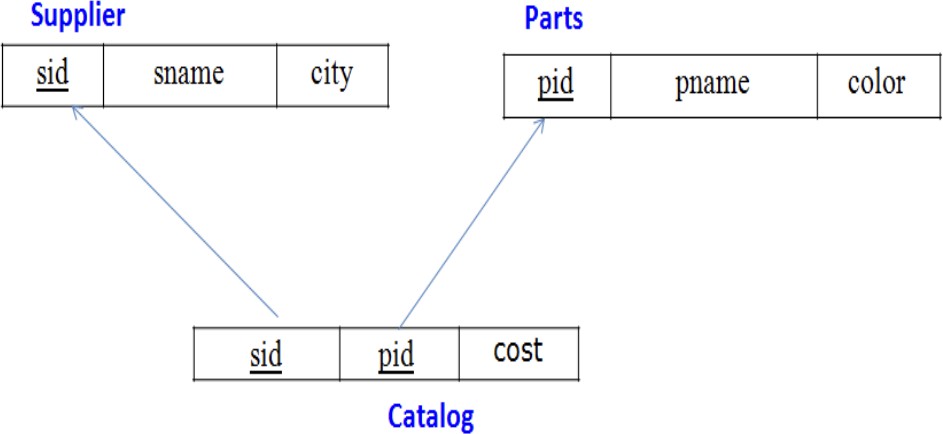


# Supplier Database

### 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** supply\_CS098;

**use** supply\_CS098;

### Create table

**create table** supplier\_CS098(

**sid** int primary **key**, sname varchar(20),

city varchar(30)

);

**create table** parts\_CS098(

pid int primary **key**, pname varchar(20),

color varchar(20)

);

**create table** catalog\_CS098(

**sid** int, pid int,

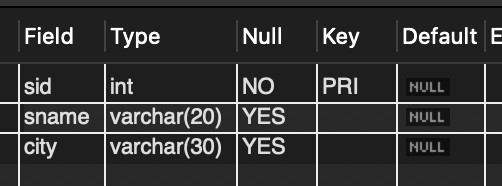
**cost** int,

foreign **key**(**sid**) **references** supplier\_CS098(**sid**), foreign **key**(pid) **references** parts\_CS098(pid)

);

### Structure of the table

desc Supplierr;



desc Parts;



desc Catalog;



### Inserting Values to the table

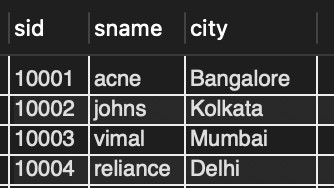
**insert into** supplier\_CS098 **values**

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

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

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

(10004, **"reliance"**, **"Delhi"**); **select** \* **from** supplier\_CS098;



**insert into** parts\_CS098 **values**

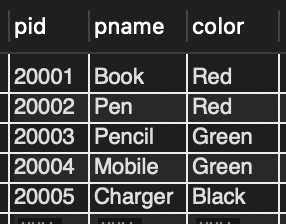
(20001, **"Book"**, **"Red"**),

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

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

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

(20005, **"Charger"**, **"Black"**); **select** \* **from** parts\_CS098;



**insert into** catalog\_CS098 **values**

(10001, 20001, 10),

(10001, 20002, 10),

(10001, 20003, 30),

(10001, 20004, 10),

(10001, 20005, 10),

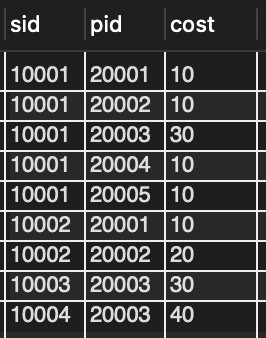
(10002, 20001, 10),

(10002, 20002, 20),

(10003, 20003, 30),

(10004, 20003, 40);

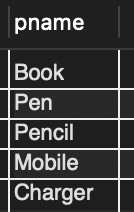
**select** \* **from** catalog\_CS098;



### Queries

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

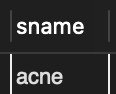
**select** pname **from** parts\_CS098 **where** pid **in** (**select** pid **from** catalog\_CS098);



###### Find the snames of suppliers who supply every part. select sname from supplier\_CS098 where sid in

(**select sid from** catalog\_CS098 **group by sid having count**(**distinct** pid) = (**select count**(**distinct**

pid) **from** parts\_CS098));

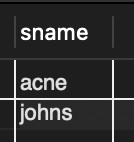


###### Find the snames of suppliers who supply every red part.

**select distinct** sname **from** supplier\_CS098, parts\_CS098, catalog\_CS098

**where** supplier\_CS098.sid = catalog\_CS098.sid **and** parts\_CS098.pid = catalog\_CS098.pid **and**

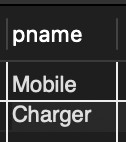
parts\_CS098.color=**"Red"**;



###### Find the pnames of parts supplied by Acme Widget Suppliers and by no one else. select pname from parts\_CS098 where pid not in

(**select** pid **from** catalog\_CS098 **where sid in** (**select sid from** supplier\_CS098 **where** sname !=

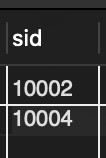
**"acne"**));



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

**select sid from** catalog\_CS098 a **where** a.cost > (**select avg**(b.cost) **from** catalog\_CS098 b **where** a.pid

= b.pid **group by** b.pid);

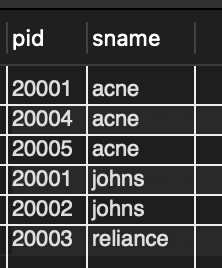


###### For each part, find the sname of the supplier who charges the most for that part.

**select** pid, sname **from** catalog\_CS098 a, supplier\_CS098

**where** a.cost = (**select max**(b.cost) **from** catalog\_CS098 b **where** a.pid = b.pid **group by** b.pid) **and**

supplier\_CS098.sid = a.sid;



# NoSQL Lab 1

### Question (Week 8)

Perform the following DB operations using MongoDB.

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

### Create database

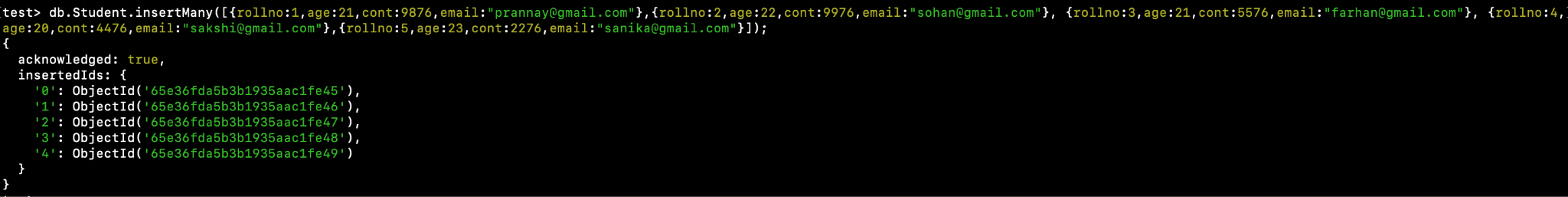
db.createCollection(**"Student"**)*;*

### Create table & Inserting Values to the table

db.Student.insertMany([{**rollno:**1,**age:**21,**cont:**9876,[**email:"prannay@gmail.com"**](mailto:prannay@gmail.com)},{**rollno:**2,**a ge:**22,**cont:**9976,[**email:"sohan@gmail.com"**](mailto:sohan@gmail.com)},

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

{**rollno:**4,**age:**20,**cont:**4476,[**email:"sakshi@gmail.com"**](mailto:sakshi@gmail.com)},{**rollno:**5,**age:**23,**cont:**2276,**email:"sa** [**nika@gmail.com"**](mailto:nika@gmail.com)}]);



**Structure of the table**

db.Student.find();



**Queries**

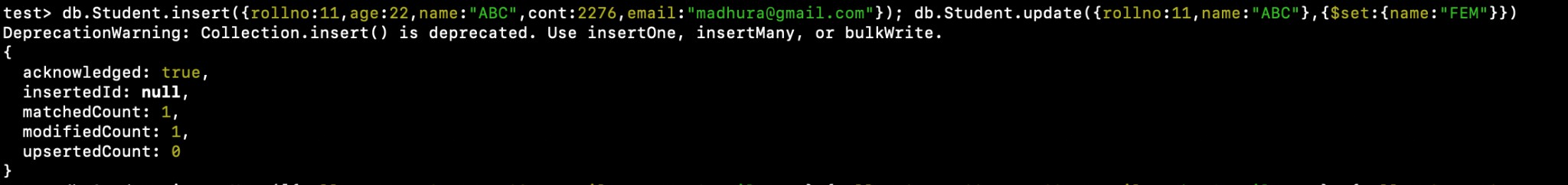
## Write a query to update the Email-Id of a student with rollno 5.

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



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

**db.Student.insert**({**rollno**:11,**age**:22,**name**:**"ABC"**,**cont**:2276,**email**:[**"madhura@gmail.com"**](mailto:madhura@gmail.com)}); **db.Student.update**({**rollno**:11,**name**:**"ABC"**},{$**set**:{**name**:**"FEM"**}})



## Export the created table into local file system

mongoexport mongodb+srv://CS098:<password>@cluster0.xbmgopf.mongodb.net/test

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

## Drop the table

db.Student.drop();



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

mongoimport mongodb+srv://CS098:<password>@cluster0.xbmgopf.mongodb.net/test

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

# NoSQL Lab 2

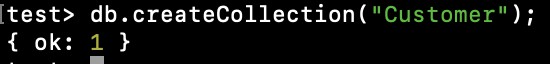
### Question (Week 9)

Perform the following DB operations using MongoDB.

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

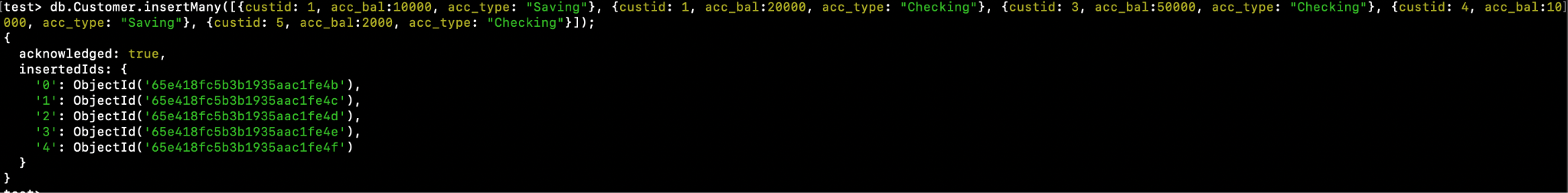
### Create Table:

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

**

### Inserting Values:

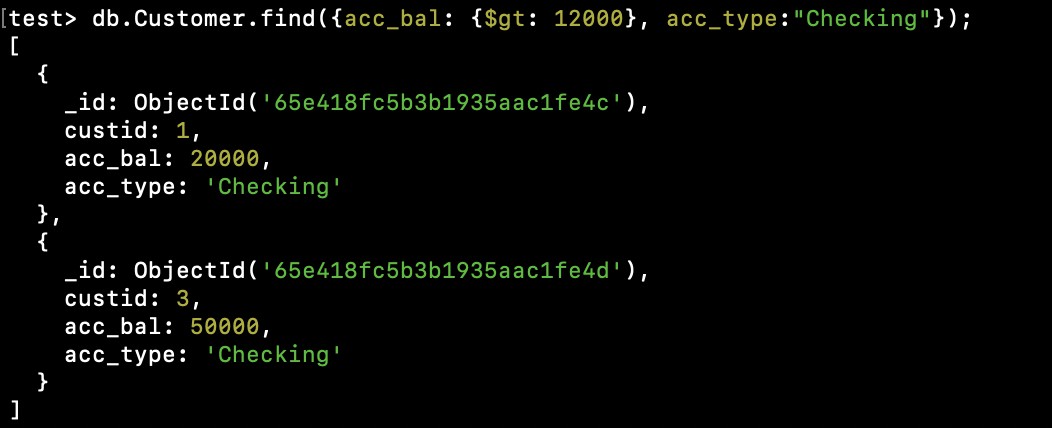
db.Customer.insertMany([{**custid:** 1, **acc\_bal:**10000, **acc\_type: "Saving"**}, {**custid:** 1, **acc\_bal:**20000, **acc\_type: "Checking"**}, {**custid:** 3, **acc\_bal:**50000, **acc\_type: "Checking"**}, {**custid:** 4, **acc\_bal:**10000, **acc\_type: "Saving"**}, {**custid:** 5, **acc\_bal:**2000,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



### Queries:

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

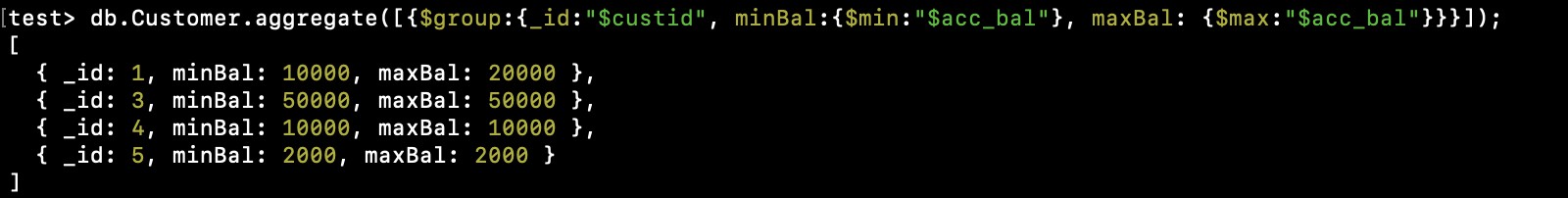
db.Customer.find({**acc\_bal**: {$gt: 12000}, **acc\_type**:"Checking"});



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

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

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



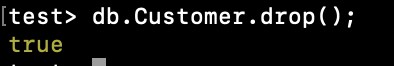
###### Exporting the collection to a json file

mongoexport mongodb+srv://CS098:<password>@cluster0.xbmgopf.mongodb.net/test

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

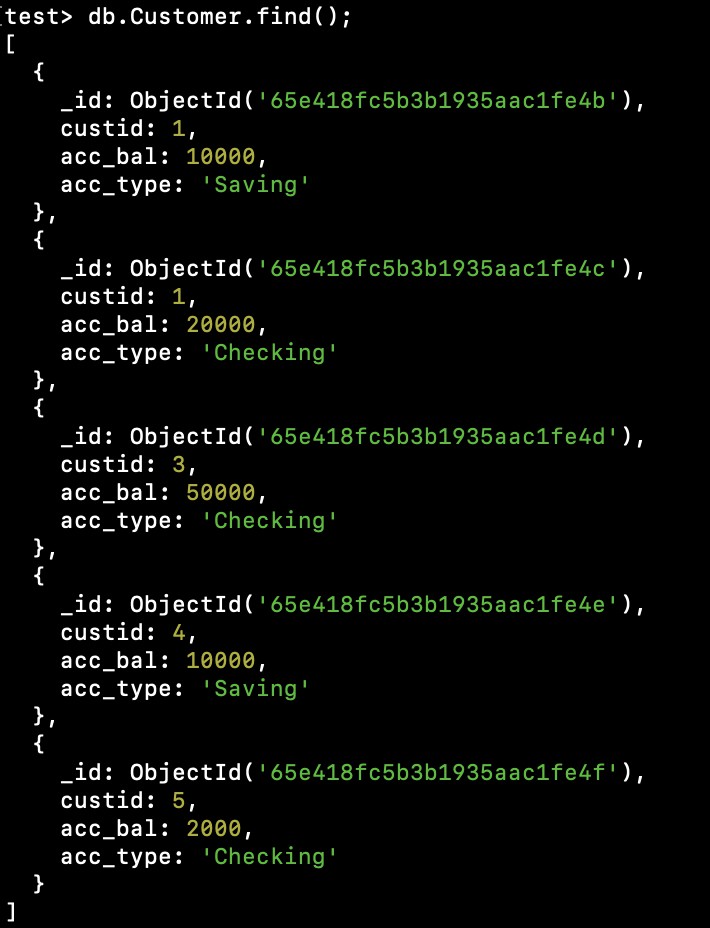
###### Dropping collection “Customer”

db.Customer.drop();



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

mongoimport mongodb+srv://CS098:<password>@cluster0.xbmgopf.mongodb.net/test

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

# NoSQL Lab 3

### Question (Week 10)

1. Write a MongoDB query to display all the documents in the collection restaurants.
2. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.
3. Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not

more than 10.

1. Write a MongoDB query to find the average score for each restaurant.
2. Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

### Creating Table:

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



### Inserting Values:

db.Restraunt.insertMany([

{

"address": { "building": "1007",

"coord": [-73.856077, 48.848447],

"street": "Morris Park Ave", "zipcode": "18462", "borough": "Bronx"

},

"cuisine": "Bakery", "grades": [

{"date": new Date("2014-03-03"), "grade": "A", "score": 2},

{"date": new Date("2013-09-11"), "grade": "A", "score": 6},

{"date": new Date("2013-01-24"), "grade": "A", "score": 10},

{"date": new Date("2011-11-23"), "grade": "A", "score": 9},

{"date": new Date("2011-03-10"), "grade": "B", "score": 14}

],

"name": "Morris Park Bake Shop", "restaurant\_id": "30075445"

},

{

"address": { "building": "2001",

"coord": [-74.005941, 40.712776],

"street": "Broadway", "zipcode": "10001", "borough": "Manhattan"

},

"cuisine": "Italian", "grades": [

{"date": new Date("2015-08-20"), "grade": "A", "score": 8},

{"date": new Date("2014-06-10"), "grade": "B", "score": 4},

{"date": new Date("2013-12-15"), "grade": "A", "score": 11},

{"date": new Date("2012-09-30"), "grade": "A", "score": 9},

{"date": new Date("2011-05-12"), "grade": "A", "score": 12}

],

"name": "Pasta Paradise", "restaurant\_id": "40092138"

},

{

"address": { "building": "3003",

"coord": [-118.243685, 34.052235],

"street": "Hollywood Blvd", "zipcode": "90028", "borough": "Los Angeles"

},

"cuisine": "Mexican", "grades": [

{"date": new Date("2016-04-15"), "grade": "A", "score": 9},

{"date": new Date("2015-12-05"), "grade": "B", "score": 6},

{"date": new Date("2014-09-20"), "grade": "A", "score": 11},

{"date": new Date("2013-06-18"), "grade": "A", "score": 8},

{"date": new Date("2012-02-10"), "grade": "A", "score": 10}

],

"name": "Sizzling Tacos", "restaurant\_id": "50065432"

},

{

"address": { "building": "4004",

"coord": [77.209021, 28.613939],

"street": "Connaught Place", "zipcode": "110001", "borough": "New Delhi"

},

"cuisine": "Indian", "grades": [

{"date": new Date("2019-10-25"), "grade": "A", "score": 8},

{"date": new Date("2018-07-15"), "grade": "B", "score": 5},

{"date": new Date("2017-04-30"), "grade": "A", "score": 10},

{"date": new Date("2016-01-12"), "grade": "A", "score": 9},

{"date": new Date("2015-05-20"), "grade": "A", "score": 12}

],

"name": "Spice Delight", "restaurant\_id": "60098765"

},

{

"address": { "building": "5005",

"coord": [76.780253, 30.728592],

"street": "Balle Balle Lane", "zipcode": "160022", "borough": "Chandigarh"

},

"cuisine": "Punjabi", "grades": [

{"date": new Date("2020-12-10"), "grade": "A", "score": 9},

{"date": new Date("2019-08-25"), "grade": "B", "score": 7},

{"date": new Date("2018-04-15"), "grade": "A", "score": 11},

{"date": new Date("2017-01-22"), "grade": "A", "score": 8},

{"date": new Date("2016-06-30"), "grade": "A", "score": 10}

],

"name": "Pind Flavors", "restaurant\_id": "70087654"

},

{

"address": { "building": "6006",

"coord": [77.594562, 12.971598],

"street": "Vidyarthi Bhavan Road", "zipcode": "560004",

"borough": "Bangalore"

},

"cuisine": "Kannadiga", "grades": [

{"date": new Date("2021-09-18"), "grade": "A", "score": 8},

{"date": new Date("2020-05-12"), "grade": "B", "score": 6},

{"date": new Date("2019-02-28"), "grade": "A", "score": 10},

{"date": new Date("2018-11-15"), "grade": "A", "score": 9},

{"date": new Date("2017-07-05"), "grade": "A", "score": 12}

],

"name": "Namma Oota", "restaurant\_id": "80076543"

},

{

"address": {

"building": "7007",

"coord": [73.856743, 18.5CS09830],

"street": "Pune-Nashik Highway", "zipcode": "411001",

"borough": "Pune"

},

"cuisine": "Maharashtrian", "grades": [

{"date": new Date("2022-05-20"), "grade": "A", "score": 9},

{"date": new Date("2021-01-15"), "grade": "B", "score": 7},

{"date": new Date("2020-08-10"), "grade": "A", "score": 11},

{"date": new Date("2019-04-25"), "grade": "A", "score": 8},

{"date": new Date("2018-10-12"), "grade": "A", "score": 10}

],

"name": "Misal Junction", "restaurant\_id": "90065432"

},

{

"address": { "building": "7007",

"coord": [73.856743, 18.5CS09830],

"street": "Shivaji Road", "zipcode": "411001",

"borough": "Pune"

},

"cuisine": "Maharashtrian", "grades": [

{"date": new Date("2022-04-30"), "grade": "A", "score": 9},

{"date": new Date("2021-10-15"), "grade": "B", "score": 7},

{"date": new Date("2020-06-28"), "grade": "A", "score": 12},

{"date": new Date("2019-03-12"), "grade": "A", "score": 8},

{"date": new Date("2018-08-20"), "grade": "A", "score": 10}

],

"name": "Vyanjan Vihar", "restaurant\_id": "90065432"

},

{

"address": { "building": "8008",

"coord": [79.312929, 9.288536],

"street": "Temple Road", "zipcode": "623526", "borough": "Rameshwaram"

},

"cuisine": "Cafe", "grades": [

{"date": new Date("2021-07-22"), "grade": "A", "score": 8},

{"date": new Date("2020-02-10"), "grade": "B", "score": 5},

{"date": new Date("2019-09-05"), "grade": "A", "score": 10},

{"date": new Date("2018-04-18"), "grade": "A", "score": 9},

{"date": new Date("2017-11-30"), "grade": "A", "score": 12}

],

"name": "Rameshwaram Retreat", "restaurant\_id": "10076543"

},

{

"address": { "building": "9009",

"coord": [80.270718, 13.082680],

"street": "Anna Salai", "zipcode": "600002", "borough": "Chennai"

},

"cuisine": "Tamil", "grades": [

{"date": new Date("2022-01-15"), "grade": "A", "score": 8},

{"date": new Date("2021-06-05"), "grade": "B", "score": 6},

{"date": new Date("2020-11-20"), "grade": "A", "score": 11},

{"date": new Date("2019-08-12"), "grade": "A", "score": 9},

{"date": new Date("2018-03-25"), "grade": "A", "score": 10}

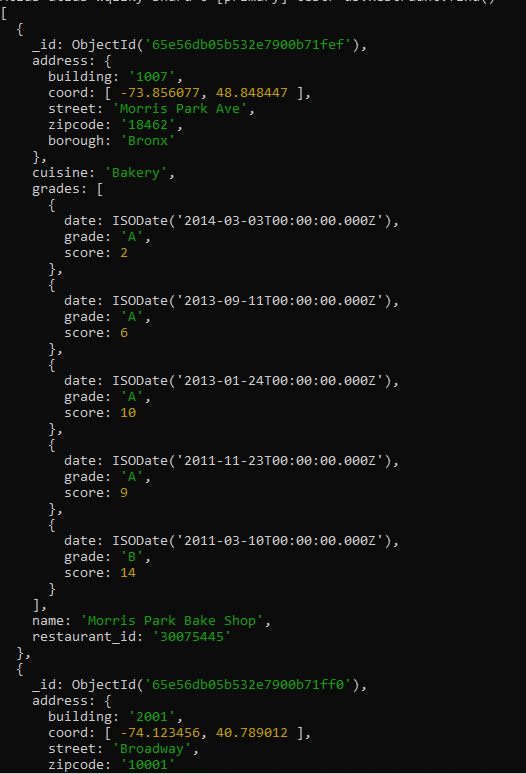
],

"name": "Tamil Delicacies", "restaurant\_id": "11076543"

}]);

##### QUERIES

1. db.Restraunt.find()



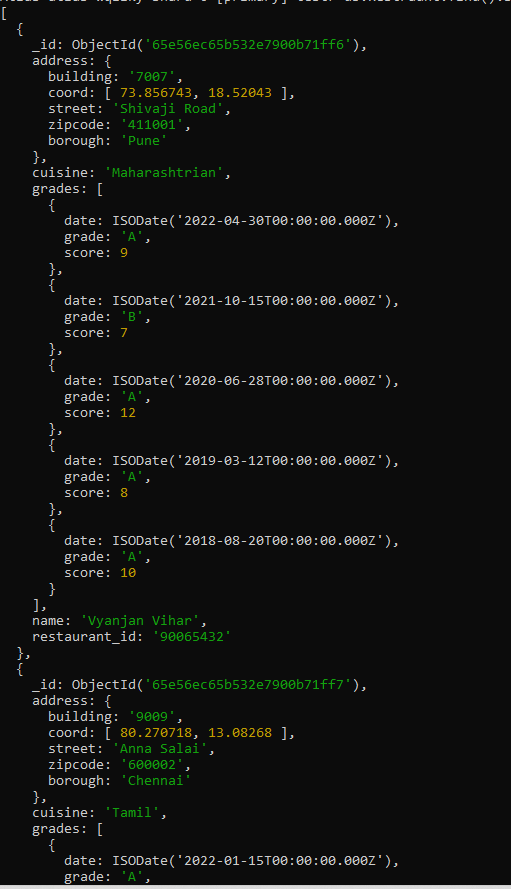


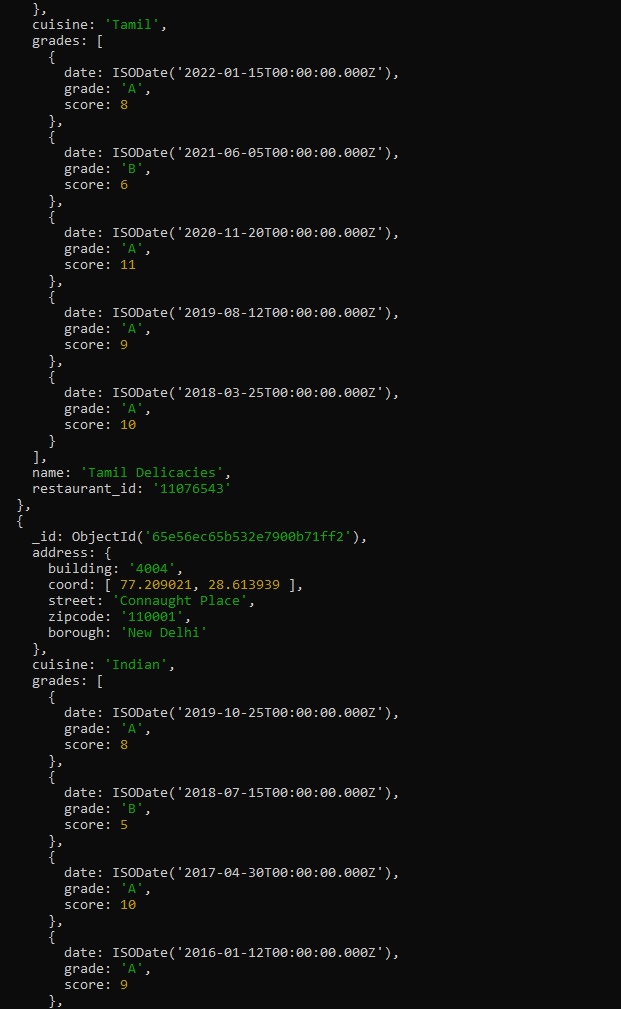


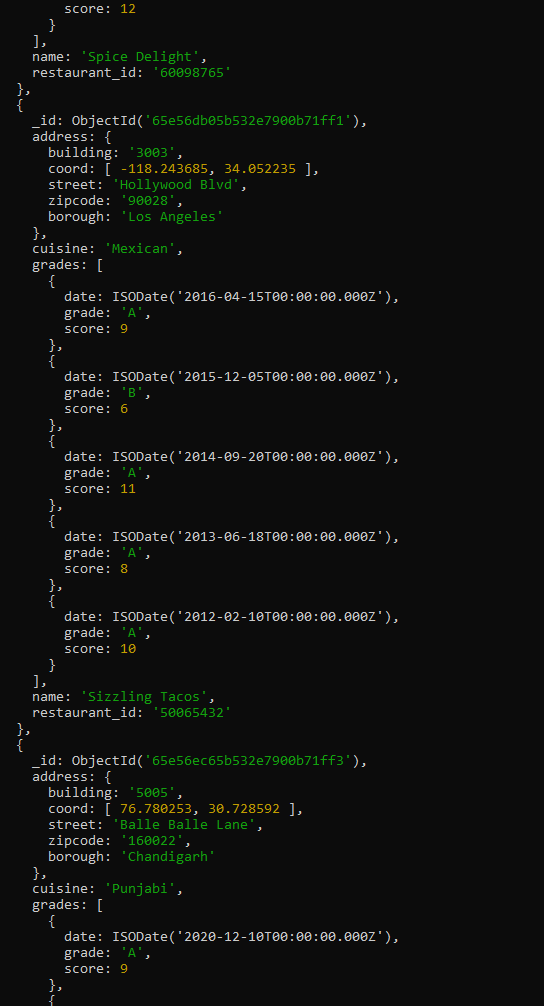




1. db.Restraunt.find().sort({ "name": -1 });











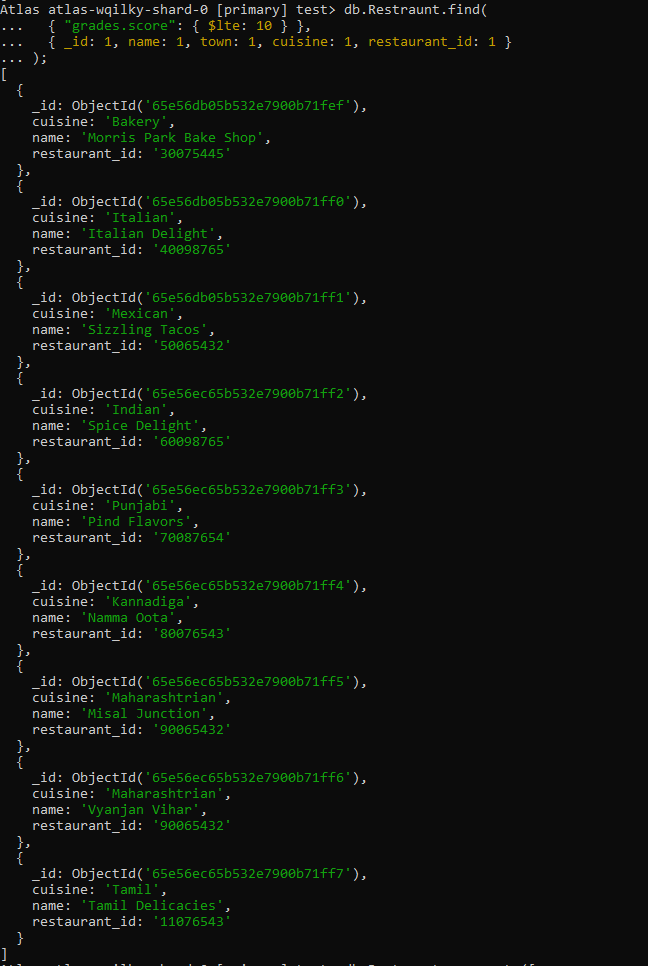


1. db.Restraunt.find(

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

{ \_id: 1, name: 1, town: 1, cuisine: 1, restaurant\_id: 1 }

);



1. db.Restraunt.aggregate([

{

$unwind: "$grades"

},

{

}

},

{

}

}

]);

$group: {

\_id: "$restaurant\_id", name: { $first: "$name" },

averageScore: { $avg: "$grades.score" }

$project: {

\_id: 1,

name: 1,

averageScore: 1



1. db.Restraunt.find(

{ "address.zipcode": { $regex: /^10/ } },

{ \_id: 0, name: 1, "address.street": 1, "address.zipcode": 1 }

);

