VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT on

Database Management Systems (23CS3PCDBM)

Submitted by

JEEVAN R(1BM25CS461)

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   Sep-2024 to Jan-2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

<u>CERTIFICATE</u>

This is to certify that the Lab work entitled "Database Management Systems (23CS3PCDBM)" carried out by **JEEVAN R(1BM25CS461),** who is Bonafede 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 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

| Lab faculty Incharge Name Prof RASHMI H Assistant Professor Department of CSE, BMSCE	Dr. KAVITHA SOODA Professor HOD Department of CSE, BMSCE

Index
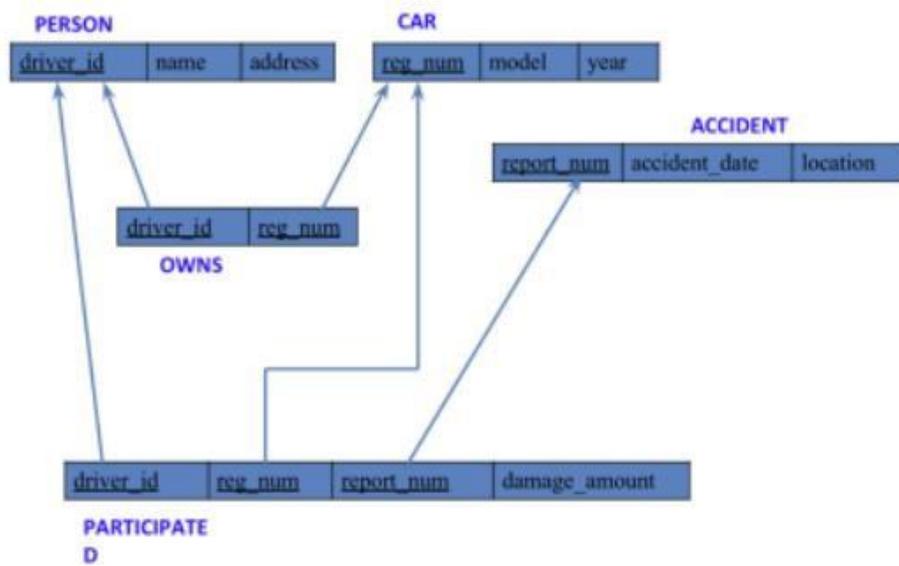
INSURANCE DATABASE

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 "KA053408") for which the accident report number was 12.
- Add a new accident to the database.
- To Do

    o Display Accident date and location o Display driver id who did accident with damage amount greater than or equal to Rs.25000

SCHEMA DIAGRAM

CREATE DATABASES:

```
create database insurance; use
INSURANCE;
```

CREATE TABLES:

```
create table person (Driver_id
varchar(10),
Name varchar(20), Address
varchar(30),
primary key(Driver_id));

create table car (reg_num
varchar(10),        Model
varchar(10), year int,
primary key(reg_num));

create table accident
(Report_num int,
Accident_date date, location
varchar(20),
primary key(Report_num));

create table owns (Driver_id varchar(10), 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 participated
(Driver_id      varchar(10),
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;

	Field	Type	Null	Key	Default	Extra
▶	Driver_id	varchar(20)	NO	PRI	NULL	
	Name	varchar(20)	YES		NULL	
	Address	varchar(30)	YES		NULL	

desc accident;

	Field	Type	Null	Key	Default	Extra
▶	Report_num	int	NO	PRI	NULL	
	Accident_date	date	YES		NULL	
	location	varchar(20)	YES		NULL	

desc participated;

	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(10)	NO	PRI	NULL	
	reg_num	varchar(10)	NO	PRI	NULL	
	Report_num	int	NO	PRI	NULL	
	damage_amount	int	YES		NULL	

desc car;

	Field	Type	Null	Key	Default	Extra
▶	reg_num	varchar(10)	NO	PRI	NULL	
	Model	varchar(10)	YES		NULL	
	year	int	YES		NULL	

desc owns;

	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(10)	NO	PRI	NULL	
	reg_num	varchar(10)	NO	PRI	NULL	

INSERTING VALUES TO THE TABLE:

insert into person values('A01','Richard','Srinivas Nagar'); insert into person values('A02','Pradeep','Rajajinagar'); 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;

Driver_id	Name	Address
A01	Richard	Srinivas Nagar
A02	Pradeep	Rajajinagar
A03	Smith	Ashok Nagar
A04	Venu	N R Colony
A05	John	Hanumanth Nagar
NULL	NULL	NULL

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;

reg_num	Model	year
KA031181	LANCER	1957
KA041702	AUDI	2005
KA052250	INDICA	1990
KA053408	HONDA	2008
KA095477	TOYOTA	1998
NULL	NULL	NULL

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','2005-03-04','Kanakpura Road'); select * from accident;

Report_num	Accident_date	location
11	2003-01-01	Mysore Road
12	2004-02-02	South End Circle
13	2003-01-21	Bull Temple Road
14	2008-02-17	Mysore Road
15	2005-03-04	Kanakpura Road
NULL	NULL	NULL

insert into owns values('A01','KA052250'); insert into owns values('A02','KA053408'); insert into owns values('A03','KA031181'); insert into owns

values('A04','KA095477'); insert into owns
values('A05','KA041702');

select * from owns;

	driver_id	reg_num
▶	A03	KA031181
	A05	KA041702
	A01	KA052250
	A02	KA053408
	A04	KA095477
*	NULL	NULL

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;

	driver_id	reg_num	Report_num	damage_amount
▶	A01	KA052250	11	10000
	A02	KA053408	12	50000
	A03	KA095477	13	25000
	A04	KA031181	14	3000
	A05	KA041702	15	5000
*	NULL	NULL	NULL	NULL

QUERIES

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

        update participated set damage_amount=25000 where
        reg_num='KA053408' and report_num=12;

	driver_id	reg_num	report_num	damage_amount
▶	A02	KA053408	12	25000
	A03	KA095477	13	25000
*	NULL	NULL	NULL	NULL

- **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%';

	CNT
▶	1

- **Display Accident date and location**

    select accident_date, location from accident;

	accident_date	location
▶	2003-01-01	Mysore Road
	2004-02-02	South End Circle
	2003-01-21	Bull Temple Road
	2008-02-17	Mysore Road
	2005-03-04	Kanakpura Road

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

    insert into accident values ('16','2008-03-15','Domlur'); select accident_date, location from accident;

	accident_date	location
▶	2003-01-01	Mysore Road
	2004-02-02	South End Circle
	2003-01-21	Bull Temple Road
	2008-02-17	Mysore Road
	2005-03-04	Kanakpura Road
	2008-03-15	Domlur

- **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;

	driver_id
▶	A02
	A03

Additional Queries on Insurance Database
Questions (Week 2)

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

    select *from car order by year asc;

	reg_num	Model	year
▶	KA031181	LANCER	1957
	KA052250	INDICA	1990
	KA095477	TOYOTA	1998
	KA041702	AUDI	2005
	KA053408	HONDA	2008
*	NULL	NULL	NULL

- **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';

count(report_num)
▶ 1

- **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%';

CNT
▶ 1

- **List the entire participated relation in the descending order of damage amount.**

  select *from participated order by damage_amount desc;

	driver_id	reg_num	Report_num	damage_amount
▶	A02	KA053408	12	50000
	A03	KA095477	13	25000
	A01	KA052250	11	10000
	A05	KA041702	15	5000
	A04	KA031181	14	3000
*	NULL	NULL	NULL	NULL

- **Find the average damage amount**

  select avg(damage_amount) from participated;

avg(damage_amount)
▶ 18600.0000

- **Delete the tuple whose damage amount is below the average damage amount**

  delete from participated where damage_amount<18600;
  select *from participated;

	driver_id	reg_num	Report_num	damage_amount
▶	A02	KA053408	12	50000
	A03	KA095477	13	25000
*	NULL	NULL	NULL	NULL

- **List the name of drivers whose damage is greater than the average damage amount.**

  select name from person a, participated b   WHERE
  a.driver_id = b.driver_id and
 damage_amount > (select avg(damage_amount) from participated);

name
▶ Pradeep
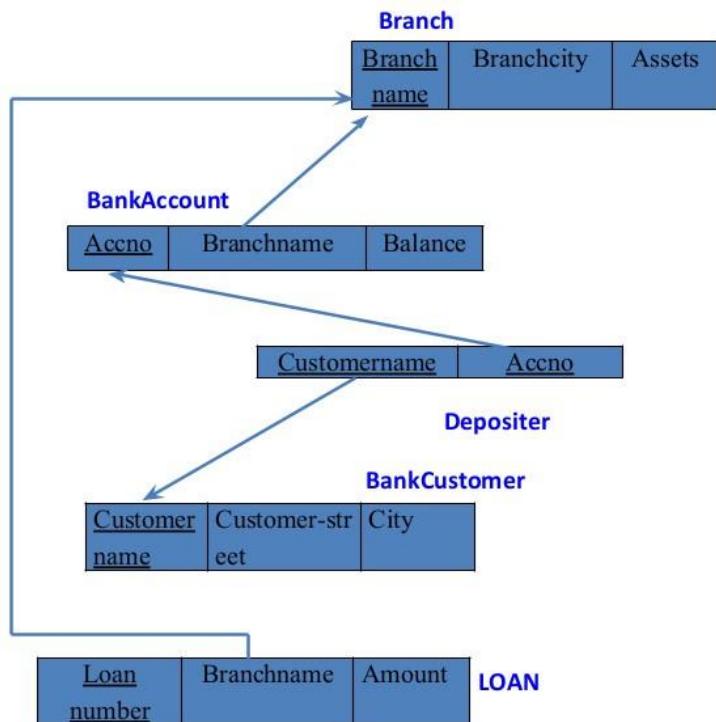
- **Find maximum damage amount.**

  select max(damage_amount) from participated;

max(damage_amount)
▶ 50000

Bank database

Schema Diagram:

Questions(week3)

- 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.

Create Database:

```
create database bank_247; use
bank_247;
```

Create table:

```
CREATE    TABLE    branch    (
branch_name        VARCHAR(30),
branch_city        VARCHAR(25),
assets INT,
    PRIMARY KEY (branch_name)
);

CREATE TABLE bankaccount (
accno INT,    branch_name
VARCHAR(30),    balance INT,
    PRIMARY KEY (accno),
    FOREIGN KEY (branch_name)
        REFERENCES branch (branch_name)
);

CREATE    TABLE    bankcustomer    (
customername        VARCHAR(20),
customer_street        VARCHAR(30),
customer_city        VARCHAR(35),
PRIMARY KEY (customername)
);

CREATE TABLE depositer (       customername
VARCHAR(20),
```

```
    accno INT,
    PRIMARY KEY (customername , accno),
    FOREIGN KEY (accno)
        REFERENCES bankaccount (accno),
    FOREIGN KEY (customername)
        REFERENCES bankcustomer (customername)
);

CREATE    TABLE    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    (
loan_number INT,      customername
VARCHAR(20),    PRIMARY KEY
(loan_number),
    FOREIGN KEY (loan_number)
        REFERENCES loan (loan_number),
    FOREIGN KEY (customername)
        REFERENCES bankcustomer (customername)
);
```

Structure of the table:

desc branch;

Field	Type	Null	Key	Default	Extra
▶ branch_name	varchar(30)	NO	PRI	NULL	
branch_city	varchar(25)	YES		NULL	
assets	int	YES		NULL	

desc bankaccount;

Field	Type	Null	Key	Default	Extra
▶ accno	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
balance	int	YES		NULL	

desc bankcustomer;

Field	Type	Null	Key	Default	Extra
▶ customername	varchar(20)	NO	PRI	NULL	
customer_street	varchar(30)	YES		NULL	
customer_city	varchar(35)	YES		NULL	

desc depositer;

	Field	Type	Null	Key	Default	Extra
▶	customername	varchar(20)	NO	PRI	NULL	
	accno	int	NO	PRI	NULL	

desc loan;

	Field	Type	Null	Key	Default	Extra
▶	loan_number	int	NO	PRI	NULL	
	branch_name	varchar(30)	YES	MUL	NULL	
	amount	int	YES		NULL	

desc borrower;

	Field	Type	Null	Key	Default	Extra
▶	loan_number	int	NO	PRI	NULL	
	customername	varchar(20)	YES	MUL	NULL	

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_Parliamentroad","Delhi",10000); insert into branch values("SBI_Jantarmantar","Delhi",20000);

select * from branch;

	branch_name	branch_city	assets
▶	SBI_Chamrajpet	Bangalore	50000
	SBI_Jantarmantar	Delhi	20000
	SBI_Parliamentroad	Delhi	10000
	SBI_ResidencyRoad	Bangalore	10000
	SBI_ShivajiRoad	Bombay	20000
*	NULL	NULL	NULL

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_Parliamentroad",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_Parliamentroad",3000); insert into bankaccount values(10,"SBI_ResidencyRoad",5000); insert into bankaccount values(11,"SBI_Jantarmantar",2000);

select * from bankaccount;

accno	branch_name	balance
1	SBI_Chamrajpet	2000
2	SBI_ResidencyRoad	5000
3	SBI_ShivajiRoad	6000
4	SBI_Parliamentroad	9000
5	SBI_Jantarmantar	8000
6	SBI_ShivajiRoad	4000
8	SBI_ResidencyRoad	4000
9	SBI_Parliamentroad	3000
10	SBI_ResidencyRoad	5000
11	SBI_Jantarmantar	2000

insert into bankcustomer values("Avinash","BUll_temple_Road","Bangalore"); insert into bankcustomer values("Dinesh","Bannergatta_Road","Bangalore"); insert into bankcustomer values("Mohan","NationaCollege_Road","Bangalore"); insert into bankcustomer values("Nikil","Akbar_Road","Delhi"); insert into bankcustomer values("Ravi","Prithviraj_Road","Delhi");

select * from bankcustomer;

customername	customer_street	customer_city
Avinash	BUll_temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationaCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

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;

customername	accno
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11
NULL	NULL

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_Parliamentroad",4000);
insert into loan values(5,"SBI_Jantarmantar",5000);

select * from loan;

loan_number	branch_name	amount
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_Parliamentroad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

insert into borrower values(1,"Mohan"); insert into
borrower values(2,"Avinash"); insert into
borrower values(3,"Dinesh"); insert into borrower
values(4,"Mohan");
insert into borrower values(5,"Nikil");

select * from borrower;

loan_number	customername
2	Avinash
3	Dinesh
1	Mohan
4	Mohan
5	Nikil
NULL	NULL

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,(assets/100000) as assets_in_lakhs from branch;

branch_name	assets_in_lakhs
SBI_Chamrajpet	0.5000
SBI_Jantarmantar	0.2000
SBI_Parliamentroad	0.1000
SBI_ResidencyRoad	0.1000
SBI_ShivajiRoad	0.2000

- **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;

customername
▶ Dinesh

- **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(amount) from loan group by
        branch_name;

        select * from sum_of_loan;

branch_name	sum(amount)
▶ SBI_Chamrajpet	1000
SBI_Jantarmantar	5000
SBI_Parliamentroad	4000
SBI_ResidencyRoad	2000
SBI_ShivajiRoad	3000

Additional Queries on Bank Database

Questions (week 4)
- Branch (branch-name: String, branch-city: String, assets: real)
- Bankaccount(accno: int, branch-name: String, balance: real)
- Bankcustomer (customer-name: String, customer-street: String, customer-city: String)
- Depositer(customer-name: String, accno: int)
- Loan (loan-number: int, branch-name: String, amount: real)

- Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).
- Find all customers who have a loan at the bank but do not have an account.
- Find all customers who have both an account and a loan at the Bangalore branch
- Find the names of all branches that have greater assets than all branches located in Bangalore.
- Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay). -   Update the Balance of all accounts by 5%

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

select distinct s.customername from depositer s
    where not exists ((select branch_name from branch where branch_city="Delhi") except (select r.branch_name from depositer t, bankaccount r where t.accno=r.accno and s.customername=t.customername));

customername
▶ Nikil

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

    select distinct customername from borrower
    where customername not in (select customername from depositer);

customername
▶ Mohan

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

    SELECT DISTINCT b.customername FROM borrower b, loan l, depositer d, branch br
    WHERE b.loan_number = l.loan_number
    AND l.branch_name = br.branch_name
    AND br.branch_city = 'Bangalore'
    AND b.customername IN (SELECT customername FROM depositer);

customername
▶ Avinash

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

  SELECT branch_name FROM branch
  WHERE assets > ALL (SELECT assets FROM branch
  WHERE branch_city = 'Bangalore');

branch_name
NULL

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

  DELETE FROM bankaccount
  WHERE branch_name IN (SELECT branch_name FROM branch
  WHERE branch_city = "Bombay");

accno	branch_name	balance
1	SBI_Chamrajpet	2000
2	SBI_ResidencyRoad	5000
4	SBI_Parliamentroad	9000
5	SBI_Jantarmantar	8000
8	SBI_ResidencyRoad	4000
9	SBI_Parliamentroad	3000
10	SBI_ResidencyRoad	5000
11	SBI_Jantarmantar	2000
NULL	NULL	NULL

- **Update the Balance of all accounts by 5%**
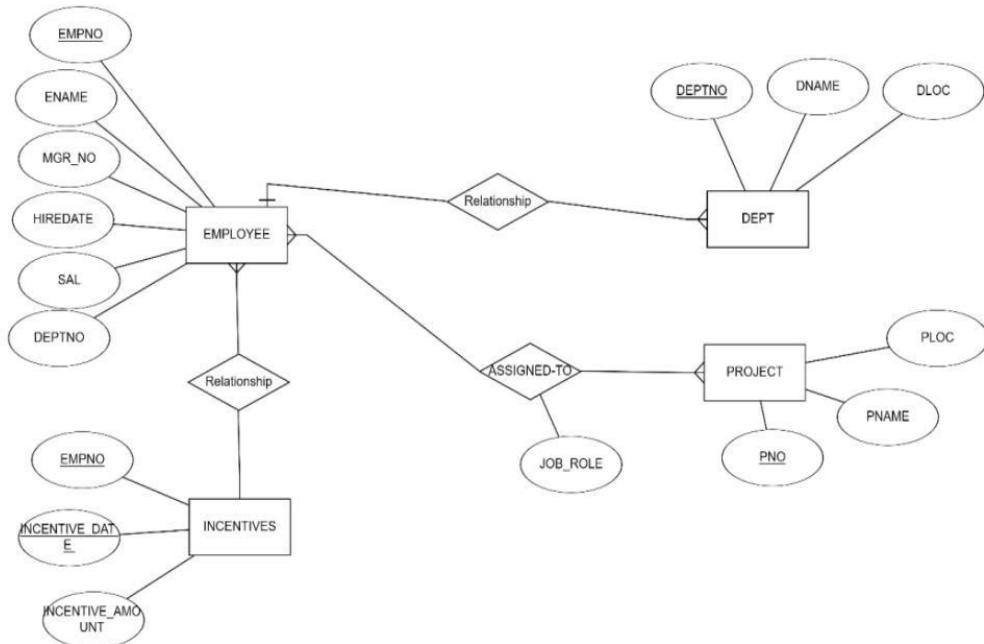
  UPDATE bankaccount
 SET balance = balance * 1.05;

  select * from bankaccount;
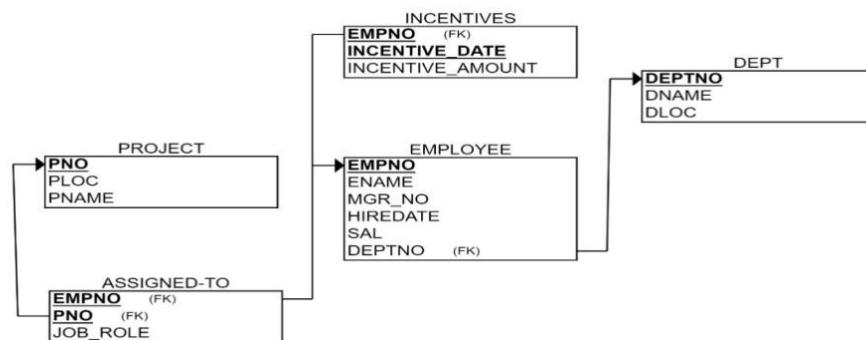
accno	branch_name	balance
1	SBI_Chamrajpet	2100
2	SBI_ResidencyRoad	5250
4	SBI_Parliamentroad	9450
5	SBI_Jantarmantar	8400
8	SBI_ResidencyRoad	4200
9	SBI_Parliamentroad	3150
10	SBI_ResidencyRoad	5250
11	SBI_Jantarmantar	2100
NULL	NULL	NULL

Employee database

ER Diagram:

Schema Diagram:

Questions (week 5):

- Using Scheme diagram, create tables by properly specifying the primary keys and the foreign keys.
- Enter greater than five tuples for each table.
- Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru
- Get Employee ID's of those employees who didn't receive incentives
- Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location. -   increase income of all employees by 5% in a table.
- Find the names of employee starting with "A"

Create database:

```
create   database   emp;   use
emp;
```

Create tables:

```
create table dept (      deptno decimal(2 , 0
) primary key,          dname  varchar(14)
default null,
   loc varchar(13) default null
);

create table emp (      empno decimal(4 , 0
) primary key,    ename varchar(10) default
null,    mgr_no decimal(4 , 0 ) default null,
hiredate date default null,     sal decimal(7
, 2 ) default null,
   deptno  decimal(2 , 0 ) references dept (deptno)        on
delete cascade on update cascade
);

create table incentives (
   empno decimal(4 , 0 ) references emp (empno)      on
delete cascade on update cascade,    incentive_date date,
incentive_amount decimal(10 , 2 ),
   primary key (empno , incentive_date)
);

create table project (       pno int
primary key,    pname varchar(30)
not null,    ploc varchar(30)
);
```

create table assigned_to (empno decimal(4 , 0)
references emp (empno)   on delete cascade on update
cascade,   pno int references project (pno)   on delete
cascade on update cascade,
    job_role varchar(30),
    primary key (empno , pno));

Structure of table:

desc dept;

	Field	Type	Null	Key	Default	Extra
▶	deptno	decimal(2,0)	NO	PRI	NULL	
	dname	varchar(14)	YES		NULL	
	loc	varchar(13)	YES		NULL	

desc emp;

	Field	Type	Null	Key	Default	Extra
▶	empno	decimal(4,0)	NO	PRI	NULL	
	ename	varchar(10)	YES		NULL	
	mgr_no	decimal(4,0)	YES		NULL	
	hiredate	date	YES		NULL	
	sal	decimal(7,2)	YES		NULL	
	deptno	decimal(2,0)	YES		NULL	

desc incentives;

	Field	Type	Null	Key	Default	Extra
▶	empno	decimal(4,0)	NO	PRI	NULL	
	incentive_date	date	NO	PRI	NULL	
	incentive_amount	decimal(10,2)	YES		NULL	

desc project;

	Field	Type	Null	Key	Default	Extra
▶	pno	int	NO	PRI	NULL	
	pname	varchar(30)	NO		NULL	
	ploc	varchar(30)	YES		NULL	

desc assigned_to;

	Field	Type	Null	Key	Default	Extra
▶	empno	decimal(4,0)	NO	PRI	NULL	
	pno	int	NO	PRI	NULL	
	job_role	varchar(30)	YES		NULL	

Inserting values into tables:

insert into dept values (10, 'accounting', 'mumbai'); insert into
dept values (20, 'research', 'bengaluru'); insert into dept values
(30, 'sales', 'delhi');
insert into dept values (40, 'operations', 'chennai');

select * from dept;

	deptno	dname	loc
▶	10	ACCOUNTING	MUMBAI
	20	RESEARCH	BENGALURU
	30	SALES	DELHI
	40	OPERATIONS	CHENNAI
*	NULL	NULL	NULL

insert into emp values (7369, 'adarsh', 7902, '2012-12-17', '80000.00', '20'); insert
into emp values (7499, 'shruthi', 7698, '2013-02-20', '16000.00', '30'); insert into
emp values (7521, 'anvitha', 7698, '2015-02-22', '12500.00', '30'); insert into emp
values (7566, 'tanvir', 7839, '2008-04-02', '29750.00', '20'); insert into emp values
(7654, 'ramesh', 7698, '2014-09-28', '12500.00', '30'); insert into emp values (7698,
'kumar', 7839, '2015-05-01', '28500.00', '30'); insert into emp values (7782, 'clark',
7839, '2017-06-09', '24500.00', '10'); insert into emp values (7788, 'scott', 7566,
'2010-12-09', '30000.00', '20'); insert into emp values (7839, 'king', null, '2009-11-
17', '90000', '10'); insert into emp values (7844, 'turner', 7698, '2010-09-08',
'15000.00', '30'); insert into emp values (7876, 'adams', 7788, '2013-01-12',
'11000.00', '20'); insert into emp values (7900, 'james', 7698, '2017-12-03',
'9500.00', '30'); insert into emp values (7902, 'ford', 7566, '2010-12-03', '30000.00',
'20'); select * from emp;

	empno	ename	mgr_no	hiredate	sal	deptno
▶	7369	Adarsh	7902	2012-12-17	80000.00	20
	7499	Shruthi	7698	2013-02-20	16000.00	30
	7521	Anvitha	7698	2015-02-22	12500.00	30
	7566	Tanvir	7839	2008-04-02	29750.00	20
	7654	Ramesh	7698	2014-09-28	12500.00	30
	7698	Kumar	7839	2015-05-01	28500.00	30
	7782	CLARK	7839	2017-06-09	24500.00	10
	7788	SCOTT	7566	2010-12-09	30000.00	20
	7839	KING	NULL	2009-11-17	90000.00	10
	7844	TURNER	7698	2010-09-08	15000.00	30
	7876	ADAMS	7788	2013-01-12	11000.00	20
	7900	JAMES	7698	2017-12-03	9500.00	30
	7902	FORD	7566	2010-12-03	30000.00	20
*	NULL	NULL	NULL	NULL	NULL	NULL

insert into incentives values(7499, '2019-02-01', 5000.00); insert into incentives values(7521, '2019-03-01', 2500.00); insert into incentives values(7566, '2022-02-01', 5070.00); insert into incentives values(7654, '2020-02-01', 2000.00); insert into incentives values(7654, '2022-04-01', 879.00); insert into incentives values(7521, '2019-02-01', 8000.00); insert into incentives values(7698, '2019-03-01', 500.00); insert into incentives values(7698, '2020-03-01', 9000.00); insert into incentives values(7698, '2022-04-01', 4500.00);

select * from incentives;

	empno	incentive_date	incentive_amount
▶	7499	2019-02-01	5000.00
	7521	2019-02-01	8000.00
	7521	2019-03-01	2500.00
	7566	2022-02-01	5070.00
	7654	2020-02-01	2000.00
	7654	2022-04-01	879.00
	7698	2019-03-01	500.00
	7698	2020-03-01	9000.00
	7698	2022-04-01	4500.00
*	NULL	NULL	NULL

insert into project values(101, 'ai project', 'bengaluru'); insert into project values(102, 'iot', 'hyderabad'); insert into project values(103, 'blockchain', 'bengaluru'); insert into project values(104, 'data science', 'mysuru');
insert into project values(105, 'autonomous systems', 'pune'); select * from project;

	pno	pname	ploc
▶	101	AI Project	BENGALURU
	102	IOT	HYDERABAD
	103	BLOCKCHAIN	BENGALURU
	104	DATA SCIENCE	MYSURU
	105	AUTONOMOUS SYSTEMS	PUNE
*	NULL	NULL	NULL

insert into assigned_to values(7499, 101, 'software engineer'); insert into assigned_to values(7521, 101, 'software architect'); insert into assigned_to values(7566, 101, 'project manager'); insert into assigned_to values(7654, 102, 'sales'); insert into assigned_to values(7521, 102, 'software engineer'); insert into assigned_to values(7499, 102, 'software engineer'); insert into assigned_to values(7654, 103, 'cyber security'); insert into assigned_to values(7698, 104, 'software engineer'); insert into assigned_to values(7900, 105, 'software engineer'); insert into assigned_to values(7839, 104, 'general manager');

select * from assigned_to;

	empno	pno	job_role
▶	7499	101	Software Engineer
	7499	102	Software Engineer
	7521	101	Software Architect
	7521	102	Software Engineer
	7566	101	Project Manager
	7654	102	Sales
	7654	103	Cyber Security
	7698	104	Software Engineer
	7839	104	General Manager
	7900	105	Software Engineer
*	NULL	NULL	NULL

Queries:

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

    select e.empno from emp e, project p,assigned_to a where
    e.empno = a.empno AND a.pno = p.pno and p.ploc in
    ('BENGALURU' , 'HYDERABAD', 'MYSURU');

	empno
▶	7499
	7499
	7521
	7521
	7566
	7654
	7654
	7698
	7839

- **Get Employee ID's of those employees who didn't receive incentives.**

    select empno from emp
    where empno not in (select empno from incentives);

	empno
▶	7369
	7782
	7788
	7839
	7844
	7876
	7900
	7902
*	NULL

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

  select e.empno , e.ename,d.dname,d.loc,a.job_role,p.ploc from
  emp e,dept d,assigned_to a,project p where e.deptno=d.deptno
  and e.empno=a.empno and
  a.pno=p.pno and d.loc=p.ploc;

empno	ename	dname	loc	job_role	ploc
7566	Tanvir	RESEARCH	BENGALURU	Project Manager	BENGALURU

- **Increase income of all employees by 5% in a table.**

  update emp set sal = 1.05*sal;
  select * from emp;

empno	ename	mgr_no	hiredate	sal	deptno
7369	Adarsh	7902	2012-12-17	84000.00	20
7499	Shruthi	7698	2013-02-20	16800.00	30
7521	Anvitha	7698	2015-02-22	13125.00	30
7566	Tanvir	7839	2008-04-02	31237.50	20
7654	Ramesh	7698	2014-09-28	13125.00	30
7698	Kumar	7839	2015-05-01	29925.00	30
7782	CLARK	7839	2017-06-09	25725.00	10
7788	SCOTT	7566	2010-12-09	31500.00	20
7839	KING	NULL	2009-11-17	94500.00	10
7844	TURNER	7698	2010-09-08	15750.00	30
7876	ADAMS	7788	2013-01-12	11550.00	20
7900	JAMES	7698	2017-12-03	9975.00	30
7902	FORD	7566	2010-12-03	31500.00	20
NULL	NULL	NULL	NULL	NULL	NULL

- **Find the names of employee starting with "A"**

  select ename from emp where
  ename like 'A%';

ename
Adarsh
Anvitha
ADAMS

More Queries on Employee Database

Questions (Week 6)
- Using Scheme diagram (under Program-5), Create tables by properly specifying the primary keys and the foreign keys.
- Enter greater than five tuples for each table.
- List the name of the managers with the maximum employees

- Display those managers name whose salary is more than average salary of his employee.
- Find the name of the second top level managers of each department.
- Find the employee details who got second maximum incentive in January 2019.
- Display those employees who are working in the same department where his manager is working.

Queries:

- **Enter greater than five tuples for each table.**

  select e.ename as manager_name, count(emp.empno) as num_employees from emp e join emp as
  emp on e.empno = emp.mgr_no group by e.ename
  having count(emp.empno) = (select max(employee_count) from (select
  count(empno) as employee_count from emp
      where mgr_no is not null
  group by mgr_no   ) as counts);

manager_name	num_employees
▶ Kumar	5

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

  select e.empno,e.ename as manager_name, e.sal as manager_salary from emp e
  where e.empno in (select mgr_no from emp
      where mgr_no is not null   group
  by mgr_no
      having e.sal > avg(sal)
  );

empno	manager_name	manager_salary
▶ 7698	Kumar	29925.00
7788	SCOTT	31500.00
7839	KING	94500.00
NULL	NULL	NULL

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

  select e2.ename as second_level_manager, d.dname as department_name from emp
  e1 join emp e2 on e1.empno = e2.mgr_no join dept d on e2.deptno = d.deptno where
  e1.mgr_no is null;

empno	second_level_manager	department_name
▶ 7782	CLARK	ACCOUNTING
7566	Tanvir	RESEARCH
7698	Kumar	SALES

- **SQL Query to find the name of the top-level manager of each department.**

  select e.empno, e.ename, d.dname from emp e, dept d where
  d.deptno = e.deptno and e.empno in (select mgr_no from
  emp) and e.deptno not in (select m.deptno from emp m where
  e.mgr_no = m.empno);

empno	ename	dname
7839	KING	ACCOUNTING
7566	Tanvir	RESEARCH
7698	Kumar	SALES

- **SQL Query to find the employee details who got second maximum incentive in February 2019.**

```
select e.ename, i.incentive_amount from emp e join
incentives  i  on  e.empno  =  i.empno  where
i.incentive_date    like    '2019-02%'         and
i.incentive_amount = (      select incentive_amount
from incentives    where incentive_date like '2019-
02%'
  order by incentive_amount desc    limit 1
offset 1
);
```

ename	incentive_amount
Shruthi	5000.00

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

```
select e.ename as employee_name, e.deptno as employee_dept,
m.ename as manager_name, m.deptno as manager_dept from emp e join emp
m on e.mgr_no = m.empno where e.deptno = m.deptno;
```

| employee_name	employee_dept	manager_name	manager_dept
Adarsh	20	FORD	20
Shruthi	30	Kumar	30
Anvitha	30	Kumar	30
Ramesh	30	Kumar	30
CLARK	10	KING	10
SCOTT	20	Tanvir	20
TURNER	30	Kumar	30
ADAMS	20	SCOTT	20
JAMES	30	Kumar	30
FORD	20	Tanvir	20
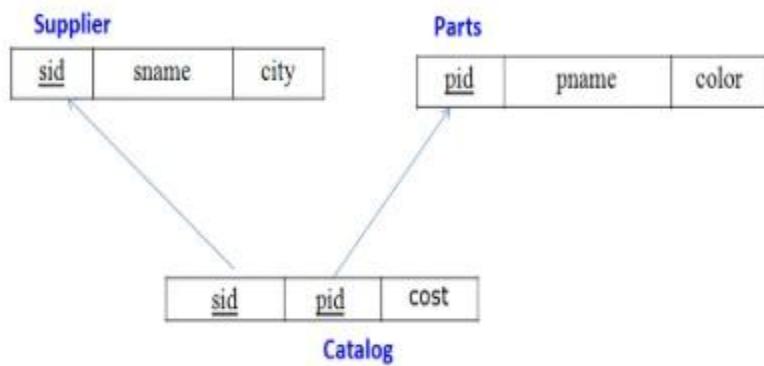
- **Write a SQL query to find those employees whose net pay are higher than or equal to the salary of any other employee in the company.**

```
select e.empno, e.ename, e.sal from emp e where e.sal
>= any (select m.sal from emp m
where m.empno not in (select p.empno from emp p where p.empno = e.empno));
```

empno	ename	sal
7369	Adarsh	84000.00
7499	Shruthi	16800.00
7521	Anvitha	13125.00
7566	Tanvir	31237.50
7654	Ramesh	13125.00
7698	Kumar	29925.00
7782	CLARK	25725.00
7788	SCOTT	31500.00
7839	KING	94500.00
7844	TURNER	15750.00
7876	ADAMS	11550.00
7902	FORD	31500.00
NULL	NULL	NULL

Supplier Database

Schema Diagram

Supplier

sid	sname	city

Parts

pid	pname	color

sid	pid	cost

Catalog

Queries (Week 7):

- Using Scheme diagram, create tables by properly specifying the primary keys and the foreign keys. Insert appropriate records in each table.
- Find the pnames of parts for which there is some supplier.
- Find the snames of suppliers who supply every part.
- Find the snames of suppliers who supply every red part.
- Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
- 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).
- For each part, find the sname of the supplier who charges the most for that part.

Create Database:

```
create   database   supplier;   use
supplier;
```

Create tables:

```
CREATE  TABLE  Supplier  (
sid  INT   PRIMARY   KEY,
sname VARCHAR(50),    city
VARCHAR(50)
);
CREATE TABLE Parts (    pid
INT PRIMARY KEY,
pname VARCHAR(50),
   color VARCHAR(20)
);

CREATE TABLE Catalog (
   sid INT,    pid INT,    cost
DECIMAL(10,          2),
PRIMARY KEY (sid, pid),
   FOREIGN KEY (sid) REFERENCES Supplier(sid),
   FOREIGN KEY (pid) REFERENCES Parts(pid)
);
```

Structure of tables:

desc supplier;

Field	Type	Null	Key	Default	Extra
sid	int	NO	PRI	NULL	
sname	varchar(50)	YES		NULL	
city	varchar(50)	YES		NULL	

desc parts;

Field	Type	Null	Key	Default	Extra
pid	int	NO	PRI	NULL	
pname	varchar(50)	YES		NULL	
color	varchar(20)	YES		NULL	

desc catalog;

Field	Type	Null	Key	Default	Extra
sid	int	NO	PRI	NULL	
pid	int	NO	PRI	NULL	
cost	decimal(10,2)	YES		NULL	

Inserting Values into tables:

INSERT INTO Supplier VALUES (10001, 'Acme Widget', 'Bangalore');
INSERT INTO Supplier VALUES (10002, 'Johns', 'Kolkata');
INSERT INTO Supplier VALUES (10003, 'Vimal', 'Mumbai');
INSERT INTO Supplier VALUES (10004, 'Reliance', 'Delhi');
INSERT INTO Supplier VALUES (10005, 'Mahindra', 'Mumbai');
select * from supplier;

sid	sname	city
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi
10005	Mahindra	Mumbai
NULL	NULL	NULL

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;

pid	pname	color
20001	Book	Red
20002	Pen	Red
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black
NULL	NULL	NULL

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;

sid	pid	cost
▶ 10001	20001	10.00
10001	20002	10.00
10001	20003	30.00
10001	20004	10.00
10001	20005	10.00
10002	20001	10.00
10002	20002	20.00
10003	20003	30.00
10004	20003	40.00
* NULL	NULL	NULL

Queries:

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

        select distinct pname from parts where pid
        in (select pid from catalog);

	pname
▶	Book
	Pen
	Pencil
	Mobile
	Charger

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

        select sname from supplier where not exists
        (select pid from parts where pid not in (select
        pid from catalog where catalog.sid =
        supplier.sid));

	sname
▶	Acme Widget

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

        select sname from supplier where not exists
        (select pid from parts
        where color = 'red' and pid not in (select pid from catalog where
        catalog.sid = supplier.sid));

	sname
▶	Acme Widget
	Johns

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

        select pname from parts where pid in (
        select pid from catalog where sid =
        (select sid from supplier where sname =
        'acme widget')) and pid not in (select pid
        from catalog

where sid != (select sid from supplier where sname = 'acme widget'));

	pname
▶	Mobile
	Charger

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

```
select distinct c.sid from catalog c join (select pid,
avg(cost) as avg_cost from catalog group by pid)
avg_table on c.pid = avg_table.pid where c.cost >
avg_table.avg_cost;
```

	sid
▶	10002
	10004

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

```
select p.pname,p.pid, s.sname from parts p
join catalog c on p.pid = c.pid join
supplier s on c.sid = s.sid
where (p.pid, c.cost) in (select pid, max(cost) from catalog group by pid);
```

	pname	pid	sname
▶	Book	20001	Acme Widget
	Book	20001	Johns
	Pen	20002	Johns
	Pencil	20003	Reliance
	Mobile	20004	Acme Widget
	Charger	20005	Acme Widget

Structure of the tabl

NoSQLLab1

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 datasheet from local file system inro collection.

Create database

db.createCollection("Student");

Create table & Inserting Values to the table
db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});

Structure of the table

db.Student.find();

```
Atlas atlas-13yfay-shard-0 [primary] test> db.Student.insert({Rollno:11,Age:21,Cont:3376,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746ba0f207c5c04af227804") }
}
Atlas atlas-13yfay-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("6746b8ff207c5c04af2277ff"),
    Rollno: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b941207c5c04af227800"),
    Rollno: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b980207c5c04af227801"),
    Rollno: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b999207c5c04af227802"),
    Rollno: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9b2207c5c04af227803"),
    Rollno: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  },
  {
    _id: ObjectId("6746ba0f207c5c04af227804"),
    Rollno: 11,
    Age: 21,
    Cont: 3376,
    email: 'pani.de9@gmail.com'
  }
]
```

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"}});

```
  },
  {
    _id: ObjectId("6746b9b2207c5c04af227803"),
    Rollno: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  },
```
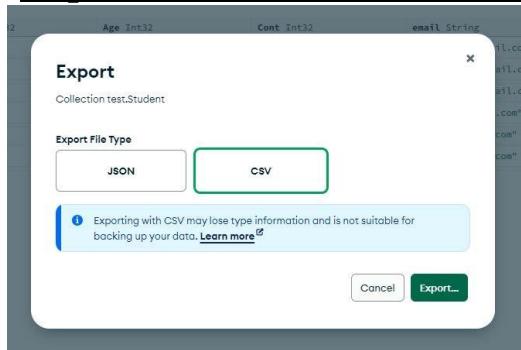
- **Replace the student name from "ABC" to "FEM" of rollno 11.**

  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"}})

```
{
  _id: ObjectId("6746ba0f207c5c04af227804"),
  Rollno: 11,
  Age: 21,
  Cont: '2276',
  email: 'rea.de9@gmail.com',
  name: 'FEM'
}
```

● **Export the created table into local files**

Export

Collection test.Student

Export File Type

| JSON | CSV |

ℹ Exporting with CSV may lose type information and is not suitable for backing up your data. **Learn more**

Cancel   Export...

	A	B	C	D	E	F
1	_id	RollNo	Age	Cont	email	Name
2	67613bdde754bbf059d14a0e	1	21	9876	antara.de9@gmail.com	
3	67613bf5e754bbf059d14a0f	2	22	9976	anushka.de9@gmail.com	
4	67613bfce754bbf059d14a10	3	21	5576	anubhav.de9@gmail.com	
5	67613c00e754bbf059d14a11	4	20	4476	pani.de9@gmail.com	
6	67613c07e754bbf059d14a12	10	23	2276	Abhinav@gmail.com	
7	67613cd2e754bbf059d14a13	11	22	2276	rea.de9@gmail.com	FEM

● **Drop the table**

db.Student.drop()

```
Atlas atlas-uyucz2-shard-0 [primary] test> db.Student.drop();
true
```

● **Import a given csv datasheet from local file system into collection.**

	A	B	C	D	E	F
1	_id	RollNo	Age	Cont	email	Name
2	67613bdde754bbf059d14a0e	1	21	9876	antara.de9@gmail.com	
3	67613bf5e754bbf059d14a0f	2	22	9976	anushka.de9@gmail.com	
4	67613bfce754bbf059d14a10	3	21	5576	anubhav.de9@gmail.com	
5	67613c00e754bbf059d14a11	4	20	4476	pani.de9@gmail.com	
6	67613c07e754bbf059d14a12	10	23	2276	Abhinav@gmail.com	
7	67613cd2e754bbf059d14a13	11	22	2276	rea.de9@gmail.com	FEM

Student

	_id ObjectId	RollNo Int32	Age Int32	Cont Int32	email String	Name String	
1	ObjectId('67613bdde754bb...	1	21	9876	"antara.de9@gmail.com"	No field	
2	ObjectId('67613bf5e754bb...	2	22	9976	"anushka.de9@gmail.com"	No field	
3	ObjectId('67613bfce754bb...	3	21	5576	"anubhav.de9@gmail.com"	No field	
4	ObjectId('67613c00e754bb...	4	20	4476	"pani.de9@gmail.com"	No field	
5	ObjectId('67613c07e754bb...	10	23	2276	"Abhinav@gmail.com"	No field	
6	ObjectId('67613cd2e754bb...	11	22	2276	"rea.de9@gmail.com"	"FEM"	

NoSQLLab2

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 balanceis

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"}]);

db.Customer.find()

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.Customer.find()
[
  {
    _id: ObjectId("6751fde06a59c75535ff9949"),
    custid: 1,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994a"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994b"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994c"),
    custid: 4,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994d"),
    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"});

Atlas atlas-13yfay-shard-0 [primary] test> db.customer.find({acc_bal:{$gt:12000},acc_type:"Checking"});
[
  {
    _id: ObjectId("674ff86c8df86e77109b3e37"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("674ff8b78df86e77109b3e38"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  }
]
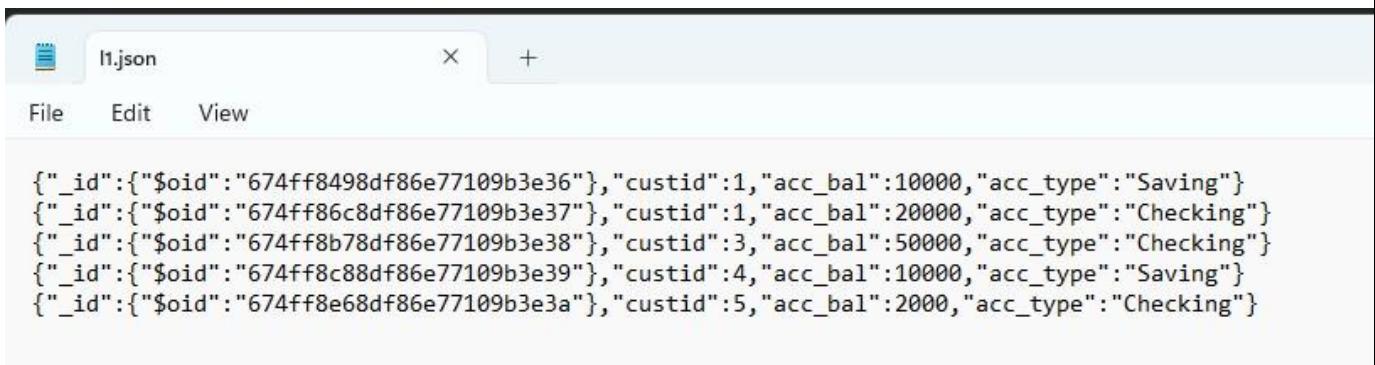
● **Finding the maximum and minimum balance of each customer**

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

Atlas atlas-13yfay-shard-0 [primary] test> db.customer.aggregate([{$group:{_id:"$custid",minBal:{$min:"$acc_bal"},maxBal:{$max:"$acc_bal"}}}]);
[
  { _id: 4, minBal: 10000, maxBal: 10000 },
  { _id: 1, minBal: 10000, maxBal: 20000 },
  { _id: 3, minBal: 50000, maxBal: 50000 },
  { _id: 5, minBal: 2000, maxBal: 2000 }
]
Atlas atlas-13yfay-shard-0 [primary] test>

● **Exporting the collection to a json file**

mongoexport mongodb+srv://Likith:@cluster0.xbmgopf.mongodb.net/test --collection=Customer -- out D:\1BM23CS171\l1.json

l1.json

File   Edit   View

{"_id":{"$oid":"674ff8498df86e77109b3e36"},"custid":1,"acc_bal":10000,"acc_type":"Saving"}
{"_id":{"$oid":"674ff86c8df86e77109b3e37"},"custid":1,"acc_bal":20000,"acc_type":"Checking"}
{"_id":{"$oid":"674ff8b78df86e77109b3e38"},"custid":3,"acc_bal":50000,"acc_type":"Checking"}
{"_id":{"$oid":"674ff8c88df86e77109b3e39"},"custid":4,"acc_bal":10000,"acc_type":"Saving"}
{"_id":{"$oid":"674ff8e68df86e77109b3e3a"},"custid":5,"acc_bal":2000,"acc_type":"Checking"}

● **Dropping collection "Customer"** db.Customer.drop();

```
Atlas atlas-13yfay-shard-0 [primary] test> db.customer.drop();
true
Atlas atlas-13yfay-shard-0 [primary] test>
```

● **Exporting from a json file to the collection**

mongoimport mongodb+srv://Likith:@cluster0.xbmgopf.mongodb.net/test --collection=Customer --file D:\1BM23CS171\l1.json

```
db.Customer.find();
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.Customer.find()
[
  {
    _id: ObjectId("6751fde06a59c75535ff9949"),
    custid: 1,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994a"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994b"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994c"),
    custid: 4,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994d"),
    custid: 5,
    acc_bal: 2000,
    acc_type: 'Checking'
  }
]
```

51

NoSQLLab3

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 indescending along with all the columns.

3.   Write a MongoDB query to find the restaurant Id, name, town andcuisine for those restaurants which achieved a score which is not more than 10.

4.   Write a MongoDB query to find the average score for each restaurant.

5.   Write a MongoDB query to find the name and address of therestaurants that have a zipcode that starts with '10'.

Creating Table:
```
db.createCollection("restaurants");
```

Inserting Values:
```
db.restaurants.insertMany([
{ name: "Meghna Foods", town: "Jayanagar", cuisine: "Indian", score: 8, address: { zipcode:
"10001", street: "Jayanagar“ } },
{ name: "Empire", town: "MG Road", cuisine: "Indian", score: 7, address: { zipcode: "10100", street: "MG
Road" } },
{ name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: { zipcode: "20000",
street: "Indiranagar" } },
{ name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode: "10300", street:
"Majestic" } },
{ name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400",
street: "Malleshwaram" }
} ])
```

QUERIES

1) db.Restraunt.find()

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({})
[
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'jayanagar' }
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'M G Road' }
  },
  {
    _id: ObjectId("675002dbf345f747889620bb"),
    name: 'Chinese Wok',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

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

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

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({}).sort({name:-1})
[
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'jayanagar' }
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'M G Road' }
  },
  {
    _id: ObjectId("675002dbf345f747889620bb"),
    name: 'Chinese Wok',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

3) Query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10

db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })

```
{
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
},
{
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian'
},
{
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese'
},
{
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Malleshwaram',
    cuisine: 'Indian'
}
```

4)Query to find the average score for each restaurant

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

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }
... ])
[
  { _id: 'WOW Momo', average_score: 5 },
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Chinese Wok', average_score: 12 },
  { _id: 'Empire', average_score: 7 }
]
```

5) Query to find the name and address of the restaurants that have a zipcode that starts with '10'.

db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
[
  { name: 'Meghna Foods', address: { street: 'jayanagar' } },
  { name: 'Empire', address: { street: 'M G Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momo', address: { street: 'Malleshwaram' } }
]
]
```