VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

Dama Yohitesh Naveen Sai(1BM23CS085)

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

CERTIFICATE

This is to certify that the Lab work entitled "Database Management Systems (23CS3PCDBM)" carried out by **Dama Yohitesh Naveen Sai (1BM23CS085)** who is 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. Kayarvizhy Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor HOD Department of CSE, BMSCE

Index

Insurance Database

Question

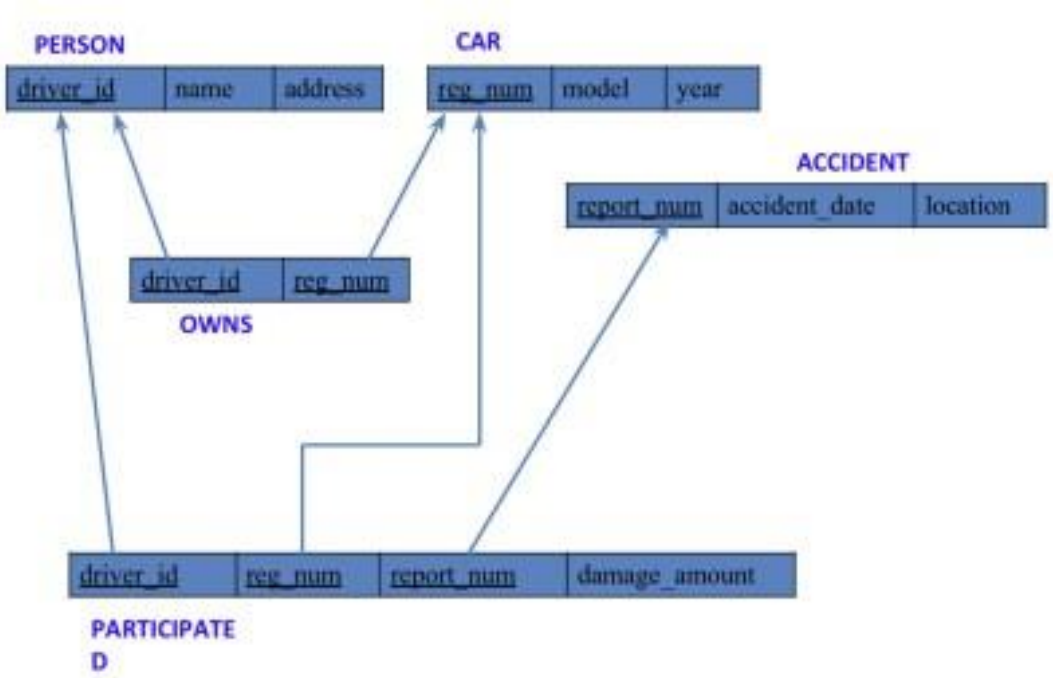
(Week 1)

- 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 'K A053408') for which the accident report number was 12.
- Add a new accident to the database.
- To Do
- Display Accident date and location
- Display driver id who did accident with damage amount greater than or equal to Rs.25000

Schema Diagram



Create database

```
create database insurance;

use insurance;
```

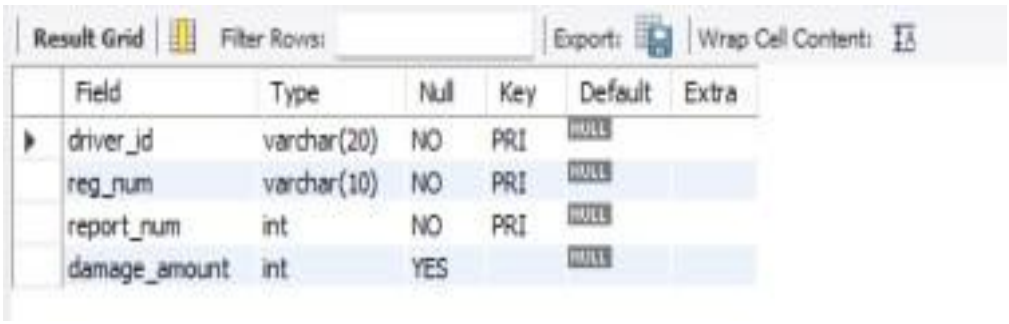
Create table

```
create table insurance.person(

driver_id varchar(20),

name varchar(30),

address varchar(50),

PRIMARY KEY(driver_id)

);

create table  car(

reg_num varchar(15),

model varchar(10),
year int,

PRIMARY KEY(reg_num)

);

create table insurance.owns(
```

```
driver_id varchar(20),
reg_num varchar(10),
PRIMARY KEY(driver_id, reg_num),
FOREIGN KEY(driver_id) REFERENCES person(driver_id),
FOREIGN KEY(reg_num) REFERENCES car(reg_num)
);
create table insurance.accident(
report_num int,
accident_date date,
location varchar(50),
PRIMARY KEY(report_num)
);
create table insurance.participated(
driver_id varchar(20),
reg_num varchar(10),
report_num int,
damage_amount int,
PRIMARY KEY(driver_id,reg_num,report_num),
FOREIGN KEY(driver_id) REFERENCES person(driver_id),
FOREIGN KEY(reg_num) REFERENCES car(reg_num),
FOREIGN KEY(report_num) REFERENCES accident(report_num)
);
```

Structure of the table

```
desc person;
```

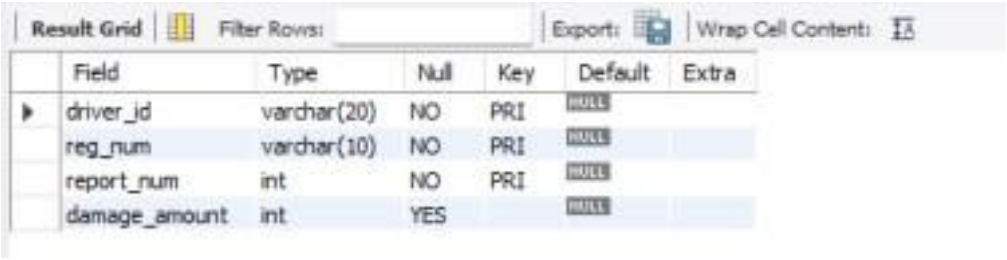


Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

```
desc accident;
```

Field	Type	Null	Key	Default	Extra
report_num	int	NO	PRI	NULL	
accident_date	date	YES		NULL	
location	varchar(50)	YES		NULL	

desc participated;



Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	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(15)	NO	PRI	NULL	
model	varchar(10)	YES		NULL	
year	int	YES		NULL	

desc owns;

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	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", "Rajaji nagar");

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

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

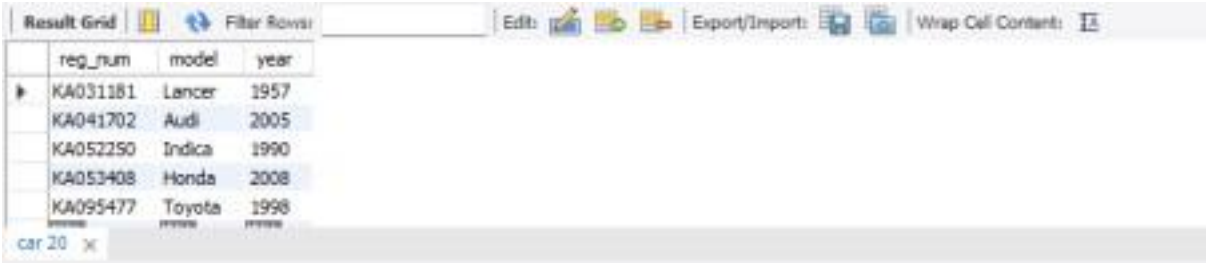
insert into person values("A05","John", "Hanumanth nagar");

select * from person;

driver_id	name	address
A01	Richard	Srinivas nagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	N R Colony
A05	John	Hanumanth nagar

person 19

```
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

car 20

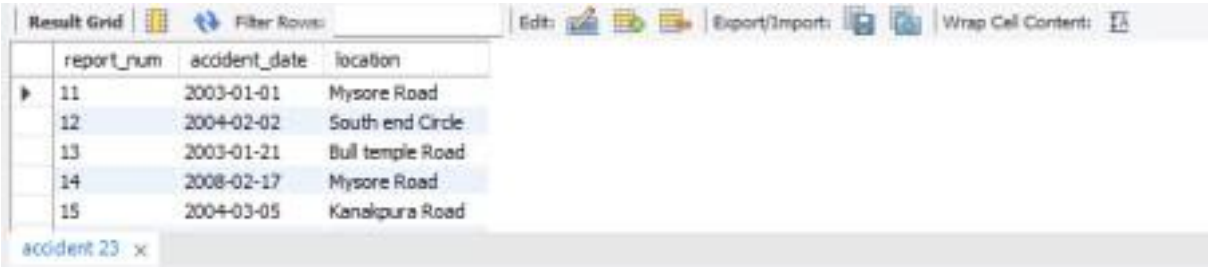
```
insert into owns values("A01","KA052250");
insert into owns values("A02","KA031181");
insert into owns values("A03","KA095477");
insert into owns values("A04","KA053408");
insert into owns values("A05","KA041702");
select * from owns;
```

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

owns 22

```
insert into accident values(11,'2003-01-01',"Mysore Road");
insert into accident values(12,'2004-02-02',"South end Circle");
insert into accident values(13,'2003-01-21',"Bull temple Road");
insert into accident values(14,'2008-02-17',"Mysore Road");
insert into accident values(15,'2004-03-05',"Kanakpura Road");
```

select * from accident;



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	2004-03-05	Kanakpura Road

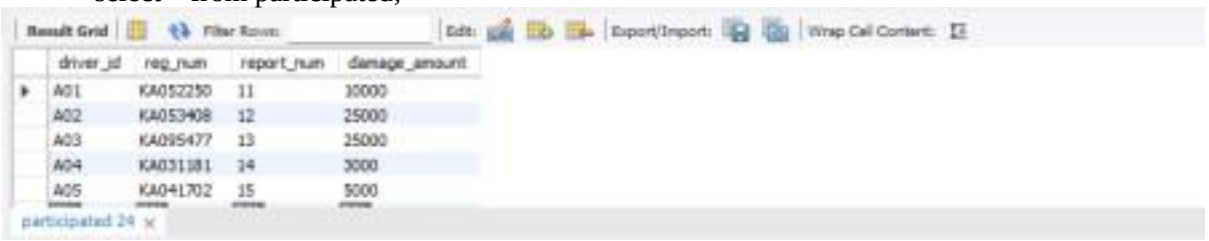
insert into participated values("A01","KA052250",11,10000);
insert into participated values("A02","KA053408",12,50000);

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	25000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000

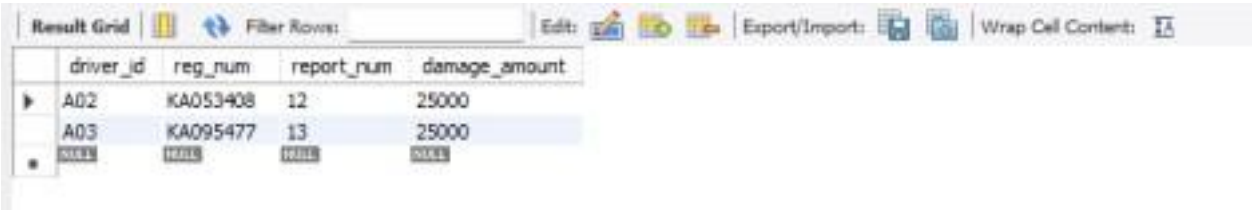
Queries

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

update participated

set damage_amount=25000

where reg_num='KA053408' and report_num=12;



driver_id	reg_num	report_num	damage_amount
A02	KA053408	12	25000
A03	KA095477	13	25000

- **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%';

	people_involved
▶	1

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

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

	report_no	accident_date	location
▶	11	2001-01-03	Mysore Rd
	12	2002-02-04	SE Circle
	13	2021-01-03	Bull Temple Rd
	14	2017-02-08	Mysore Rd
	15	2004-03-05	KR Puram
	16	2008-03-08	Domlur
*	NULL	NULL	NULL

TO DO:

● DISPLAY ACCIDENT DATE AND LOCATION

select accident_date as date, location from accident;

	accident_date	location
▶	2003-01-01	mysore road
	2004-02-02	south end
	2003-01-21	bull temple road
	2003-02-17	mysore road
	2003-03-15	kanakpura ROAD

accident_170 36 ×

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

select participated_204.driver_id as driver_id from accident_204, participated_204 where accident_204.report_no = participated_204.report_no and participated_204.damage_amt >= 25000;

	driver_id
▶	A02
	A03

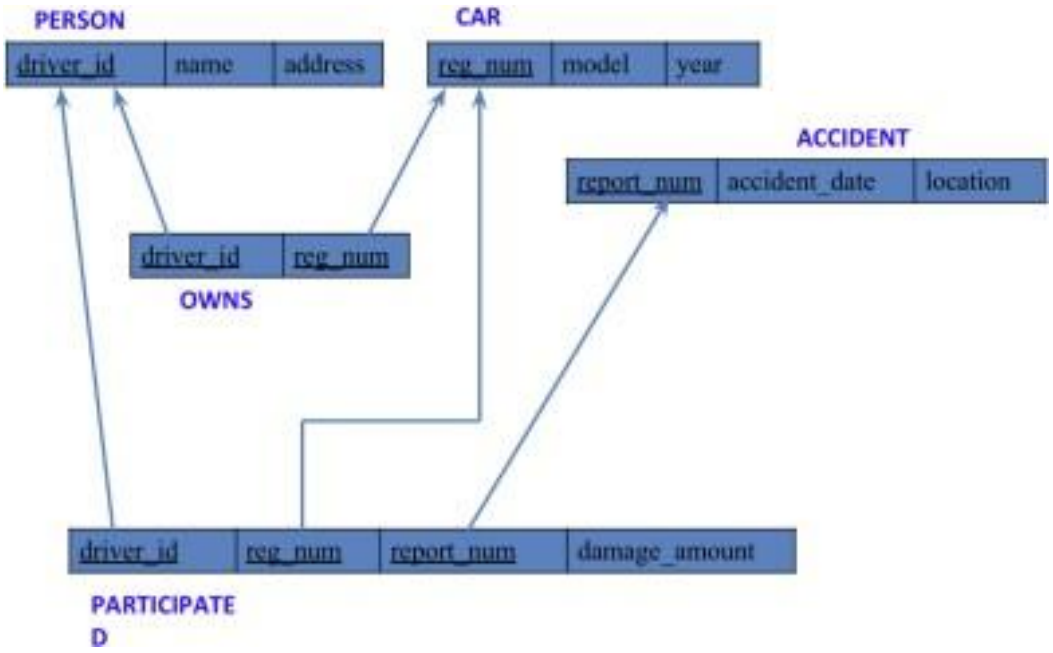
More Queries on Insurance Database

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

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

SELECT * FROM PARTICIPATED ORDER BY DAMAGE_AMOUNTT DESC;

	driver_id	reg_num	report_num	damage_amount
▶	A02	KA053408	12	25000
	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_AMOUNTT) FROM PARTICIPATED;

	avg(damage_amount)
▶	13600.0000

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
	Smith

FIND MAXIMUM DAMAGE AMOUNT.

SELECT MAX(DAMAGE_AMOUNT) FROM PARTICIPATED;

	MAX(DAMAGE_AMOUNT)
▶	25000

Bank Database

Question

(Week 3)

- Branch (branch-name: String, branch-city: String, assets: real)

- BankAccount(accno: int, branch-name: String, balance: real)

- BankCustomer (customer-name: String, customer-street: String, customer-city: String) - Depositer(customer-name: String, accno: int)

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

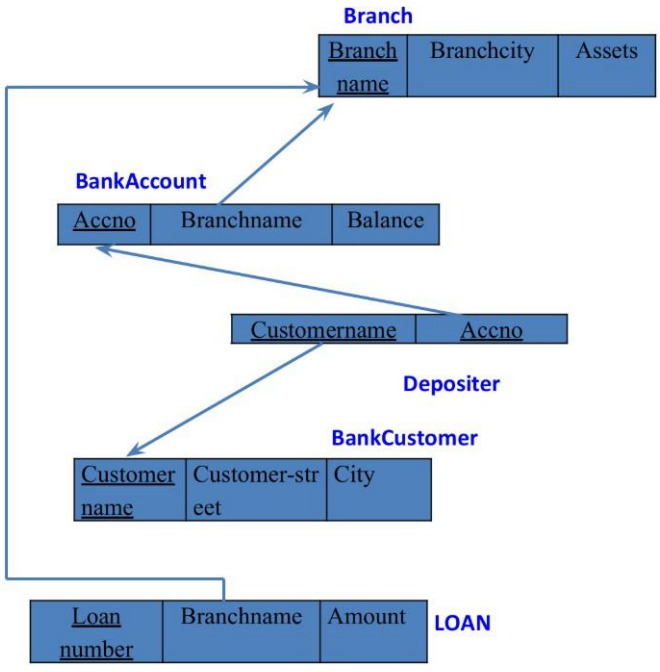
- Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation.

- 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 DB:

```
create database bank170;
show databases;
use bank170;
```

create tables:

```
create table BRANCH(
NAME VARCHAR(20),
CITY VARCHAR(20),
ASSETS VARCHAR(20),
PRIMARY KEY(NAME));

create table bankaccount(
accno VARCHAR(20),
nAME VARCHAR(20),
BALANCE VARCHAR(20),
PRIMARY KEY (ACCNO,NAME),FOREIGN KEY (NAME) REFERENCES BRANCH (NAME));

CREATE TABLE CUSTOMER (NAME VARCHAR(20),
STREET VARCHAR(20),
CITY VARCHAR(20),
PRIMARY KEY (NAME));

CREATE TABLE  LOAN(
LOAN_NO VARCHAR(20),
NAME  VARCHAR(20),
AMOUNT VARCHAR(20),
PRIMARY KEY (NAME ),
foreign key (NAME ) references BRanch(name));
```

inserting values:

```
insert into branch values ("SBI_chmarajpet","banglore",50000);
insert into branch values ("SBI_residencyroad","banglore",10000);
insert into branch values ("SBI_shivajiroad","bombay",20000);
insert into branch values ("SBI_parlimentroad","delhi",10000);
insert into branch values ("SBI_jantarmantar","delhi",20000);

insert into bankaccount values (1,"SBI_chmarajpet",2000);
insert into bankaccount values (2,"SBI_residencyroad",5000);
insert into bankaccount values (3,"SBI_shivajiroad",6000);
insert into bankaccount values (4,"SBI_parlimentroad",9000);
```

```
insert into bankaccount values (5,"SBI_jantarmantar",8000);
insert into bankaccount values (6,"SBI_shivajiroad",4000);
insert into bankaccount values (8,"SBI_residencyroad",4000);
insert into bankaccount values (9,"SBI_parlimentroad",3000);
insert into bankaccount values (10,"SBI_residencyroad",5000);
insert into bankaccount values (11,"SBI_jantarmantar",2000);

insert into customer values ("avinash","bull temple road","banglore");
insert into customer values ("dinesh","bannerghatta road","banglore");
insert into customer values ("mohan","national clg road","banglore");
insert into customer values ("nikhil","akbar road ","delhi");
insert into customer values ("ravi","prithvi raj road","delhi");

insert into depositer values ("avinash",1);
insert into depositer values ("dinesh",2);
insert into depositer values ("mohan",3);
insert into depositer values ("nikhil",4);
insert into depositer values ("ravi",5);
insert into depositer values ("avinash",8);
insert into depositer values ("nikhil",9);
insert into depositer values ("dinesh",10);
insert into depositer values ("nikhil",11);

insert into loan values (1,"SBI_chmarajpet",1000);
insert into loan values (2,"SBI_residencyroad",2000);
insert into loan values (3,"SBI_shivajiroad",3000);
insert into loan values (4,"SBI_parlimentroad",4000);
insert into loan values (5,"SBI_jantarmantar",5000);
select * from branch;
```

NAME	CITY	assets_inlakhs
SBI_chmarajpet	banglore	50000
SBI_jantarmantar	delhi	20000
SBI_parlimentroad	delhi	10000
SBI_residencyroad	banglore	10000
SBI_shivajiroad	bombay	20000
NULL	NULL	NULL

branch 22 ×

```
select * from bankaccount;
```

accno	nAME	BALANCE
1	SBI_chmarajpet	2000
10	SBI_residencyroad	5000
11	SBI_jantarmantar	2000
2	SBI_residencyroad	5000
3	SBI_shivajiroad	6000
4	SBI_parlimentroad	9000

bankaccount 23 ×

select * from customer;

NAME	STREET	CITY
avinash	bull temple road	banglore
dinesh	bannerghatta road	banglore
mohan	national clg road	banglore
nikhil	akbar road	delhi
ravi	prithvi raj road	delhi
NULL	NULL	NULL

customer 24 ×

select * from depositer;

NAME	ACCNO
avinash	1
dinesh	10
nikhil	11
dinesh	2
mohan	3
nikhil	4

select * from loan;

NAME	ACCNO
avinash	1
dinesh	10
nikhil	11
dinesh	2
mohan	3
nikhil	4

Queries:

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

select name,assets_inlakhs from branch;
alter table branch rename column assets to assets_inlakhs;

name	assets_inlakhs
SBI_chmarajpet	50000
SBI_jantarmantar	20000
SBI_parlimentroad	10000
SBI_residencyroad	10000
SBI_shivajiroad	20000

- **Find all the customers who have at least two accounts at the same branch (ex.SBI_ResidencyRoad).**

select d.name from Depositer d, BankAccount b where b.name='SBI_ResidencyRoad' and d.Accno=b.Accno group by d.name having count(d.Accno)>=2;

	name
▶	dinesh

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

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

select * from br;

	name	sum(amount)
▶	SBI_chmarajpet	1000
	SBI_jantarmantar	5000
	SBI_parlimentroad	4000
	SBI_residencyroad	2000
	SBI_shivajiroad	3000

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).

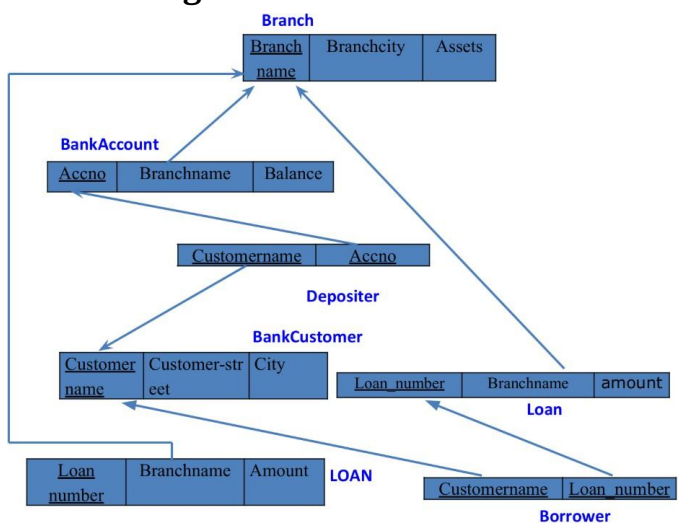
- Find all customers who have a loan at the bank but do not have an account. - Find all customers who have both an account and a loan at the Bangalore branch

- 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(
customer_name varchar(20),

```
    loan_no int,
    foreign key(customer_name) references bank_customer_204(customer_name),
    foreign key(loan_no) references loans_204(loan_no)
);
```

Inserting values:

```
insert into branch values ("SBI_MantriMarg", "Delhi", 200000);
insert into bank_account values (12, "SBI_MantriMarg", 2000);
insert into deposits values("Nikhil", 12);

insert into borrower 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 distinct d.customer_name
from depositer d,bankAccount ba,branch b
where d.accno=ba.accno and ba.branch_name=b.branch_name and b.city='delhi'
group by d.customer_name having count(b.branch_name)>1;
```

customer_name
nikil

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

```
select b.customer_name
from borrower b
where b.loan_no not in(select d.accno from depositer d where b.loan_no=d.accno);
```

customer_name
Mohan

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

```
select b.customer_name
from borrower b
where b.loan_no in (select d.accno from depositer d,bankAccount ba,branch b
where b.loan_no=d.accno and d.accno=ba.accno and ba.branch_name=b.branch_name and b.city='Bangalore');
```

	customer_name
▶	Avinash
	Dinesh

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

```
select branch_name
from branch
where assets_inlakhs>all(select assets_inlakhs from branch where city='Bangalore');
```

	branch_name
▶	SBI_jantarmantar
	SBI_MantriMarg
	SBI_shivajiRoad
*	NULL

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

```
delete from bankAccount ba
where ba.branch_name=(select b.branch_name from branch b where city='Bombay');
select * from  bankAccount;
```

	accno	branch_name	balance
▶	1	SBI_chamrajpet	2000
	2	SBI_residencyRoad	5000
	4	SBI_parlimentRoad	9000
	5	SBI_jantarmantar	8000
	8	SBI_residencyRoad	4000
	9	SBI_parlimentRoad	3000
	10	SBI_residencyRoad	5000
	11	SBI_jantarmantar	2000
	12	SBI_MantriMarg	2000
*	NULL	NULL	NULL

● Update the Balance of all accounts by 5%

```
update bankAccount
set balance=balance+((5*balance)/100);
select * from bankAccount;
```

	accno	name	balance
▶	1	SBI_Chamrajpet	2100
	2	SBI_Residency road	5250
	3	SBI_Shivaji road	6300
	4	SBI_Parliament road	9450
	5	SBI_Jantarmantar	8400
	6	SBI_Shivaji road	4200
	8	SBI_Residency road	4200
	9	SBI_Parliament road	3150
	10	SBI_Residency road	5250
	11	SBI_Jantarmantar	2100
	12	SBI_MantriMarg	2100
*	NULL	NULL	NULL

6300

Employee Database

Question

(Week 5)

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

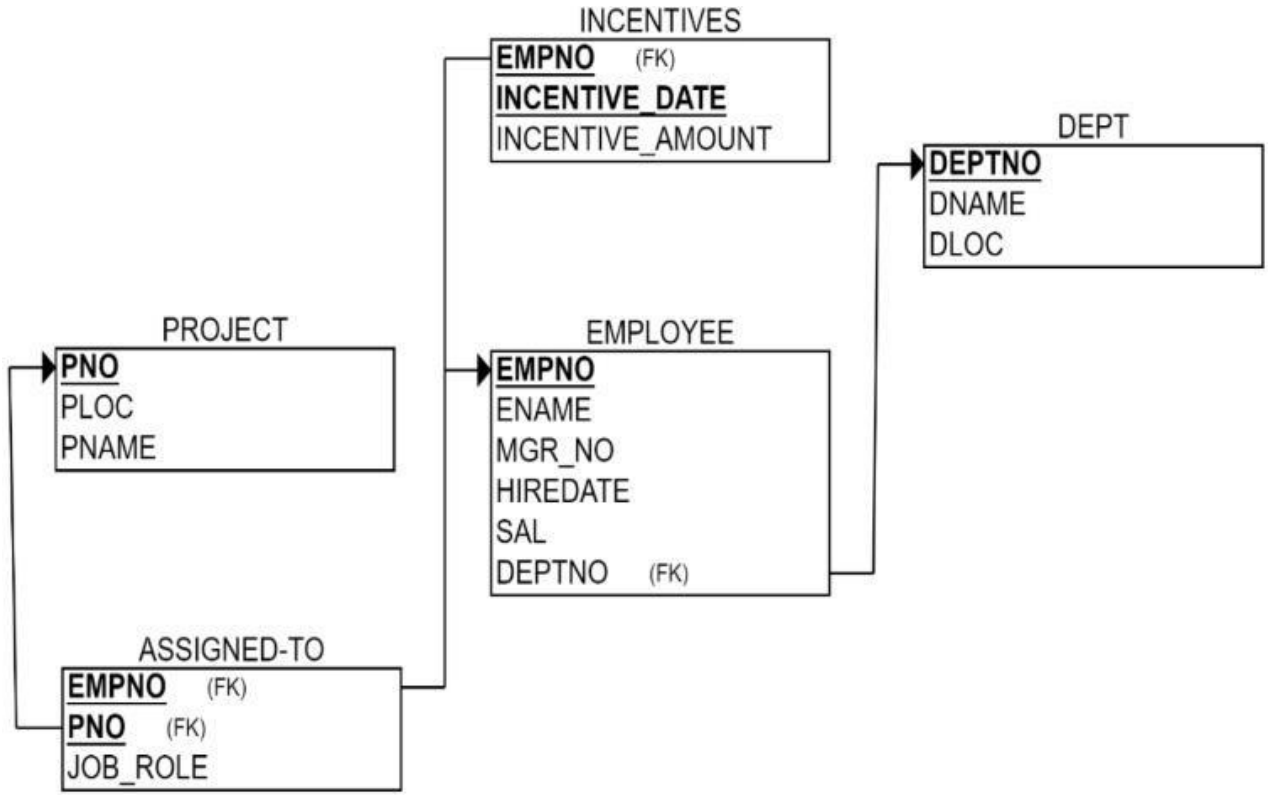
2. Enter greater than five tuples for each table.

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;
use employee_database;

Create table

```
create table project(
        pno int primary key,
        ploc varchar(20),
        pname varchar(20)
);
create table dept(
        deptno int primary key,
        dname varchar(30),
        dloc varchar(30)
);
create table employee(
        empno int primary key,
        ename varchar(20),
        mgr_no int,
        hiredate date,
        sal double,
        deptno int,
        foreign key(deptno) references dept(deptno)
);
create table assigned_to(
        empno int primary key,
        pno int,
        job_role varchar(20),
        foreign key(empno) references employee(empno),
        foreign key(pno) references projec(pno)
);
create table incentives(
        empno int,
        incentive_date date primary key,
        incentive_amount double,
        foreign key(empno) references employee(empno));
```

insert values:

```
insert into project values(
        (1,"bengaluru","abcd"),
        (2,"hyderabad","bcda"),
        (3,"bengaluru","abab"),
        (4,"bengaluru","baba"),
        (5,"hyderabad","cdcd"),
        (6, "mysuru", "efef"));

insert into dept values (
        (1,"cse","bengaluru"),
```

(2,"**ise**","**hyderabad**"),
(3,"**ece**","**bengaluru**"),
(4,"**ete**","**hyderabad**"),
(5,"**ime**","**bengaluru**"),
(6, "**mech**", "**mysuru**"));

insert into employee **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));

insert into incentives **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));

insert into assigned_to **values (**
(1,1, "**employee**"),
(2,1, "**manager**"),
(3,2, "**manager**"),
(4,3, "**employee**"),
(5,4, "**employee**"),
(6, 6, "**employee**"));

select * from project;

	pno	ploc	pname
▶	1	bengaluru	abcd
	2	hyderabad	bcda
	3	bengaluru	abab
	4	bengaluru	baba
	5	hyderabad	cdcd
	6	mysuru	efef
*	NULL	NULL	NULL

select * from dept;

deptno	dname	dloc
1	cse	bengaluru
2	ise	hyderabad
3	ece	bengaluru
4	ete	hyderabad
5	ime	bengaluru
6	mech	mysuru
NULL	NULL	NULL

select * from employee;

empno	ename	mgr_no	hiredate	sal	deptno
1	a	NULL	2023-11-09	70000	1
2	b	2	2023-08-09	70000	1
3	c	3	2023-06-08	70000	2
4	d	NULL	2023-08-06	70000	2
5	e	NULL	2023-05-04	70000	3
6	f	NULL	2023-06-01	90000	6
NULL	NULL	NULL	NULL	NULL	NULL

select * from incentivest;

empno	incentive_date	incentive_amount
4	2023-05-04	10000
3	2023-06-08	10000
2	2023-08-09	10000
5	2023-12-08	10000
1	2023-12-09	10000
NULL	NULL	NULL

select * from project;

empno	pno	job_role
1	1	employee
2	1	manager
3	2	manager
4	3	employee
5	4	employee
6	6	employee
NULL	NULL	NULL

Queries

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

select assigned_to.empno **from** assigned_to, project
where assigned_to.pno = project.pno **and** project.ploc **in** ("**bengaluru**", "**mysuru**", "**hyderabad**");

empno
101
501

● Get Employee ID's of those employees who didn't receive incentives

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

	empno
▶	601

● **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.empno, ename, dname, job_role, dloc, ploc
from employee, assigned_to, project, dept
where ploc = dloc **and** assigned_to.empno = employee.empno
and employee.deptno = dept.deptno **and** project.pno = assigned_to.pno;

	empno	ename	dname	job_role	dloc	ploc
▶	101	aaa	cse	devops	banglore	banglore
	201	bbb	ise	sde	delhi	delhi
	301	ccc	csds	manager	bombay	bombay
	401	ddd	ece	jpa	chennai	chennai
	501	eee	aiml	pa	mysuru	mysuru

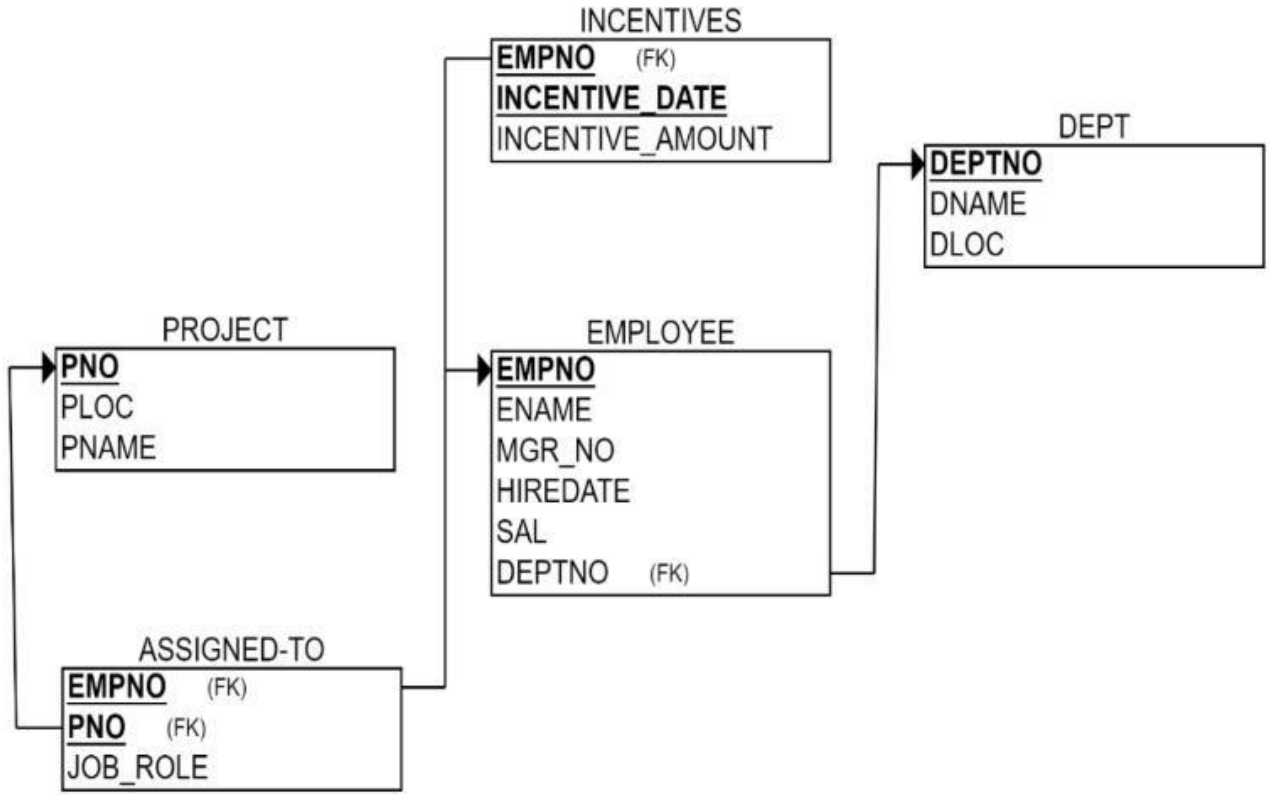
More Queries on Employee Database

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

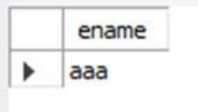


Queries

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

select e.ename from employee e where e.empno in(select mgr_no from employee group by mgr_no having count(*)=
(select max(emp_count) from (select count(*) as emp_count from employee group by mgr_no)AS

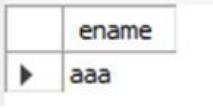
SUBQUERY));



	ename
▶	aaa

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

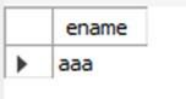
```
select e.ename from employee e
where e.sal>(select avg(sub.sal) from employee sub
where sub.mgr_no=e.empno);
```



	ename
▶	aaa

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

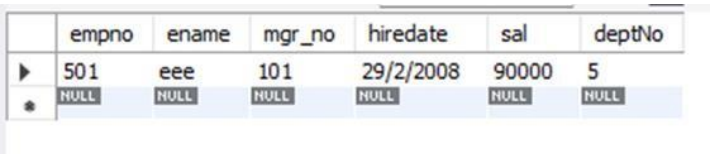
```
select ename from employee where sal=(select max(sal) from employee where
sal<(select max(Sal) from employee));
```



	ename
▶	aaa

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

```
select * from employee where empno=(select empno from incentives where amount=
(select max(amount) from incentives where amount<
(select max(amount) from incentives)));
```



	empno	ename	mgr_no	hiredate	sal	deptNo
▶	501	eee	101	29/2/2008	90000	5
*	NULL	NULL	NULL	NULL	NULL	NULL

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

```
select E.ename from employee e , employee m WHERE e.mgr_no=m.empno aND
e.deptno=m.deptno;
```

	ename
▶	GGG

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.

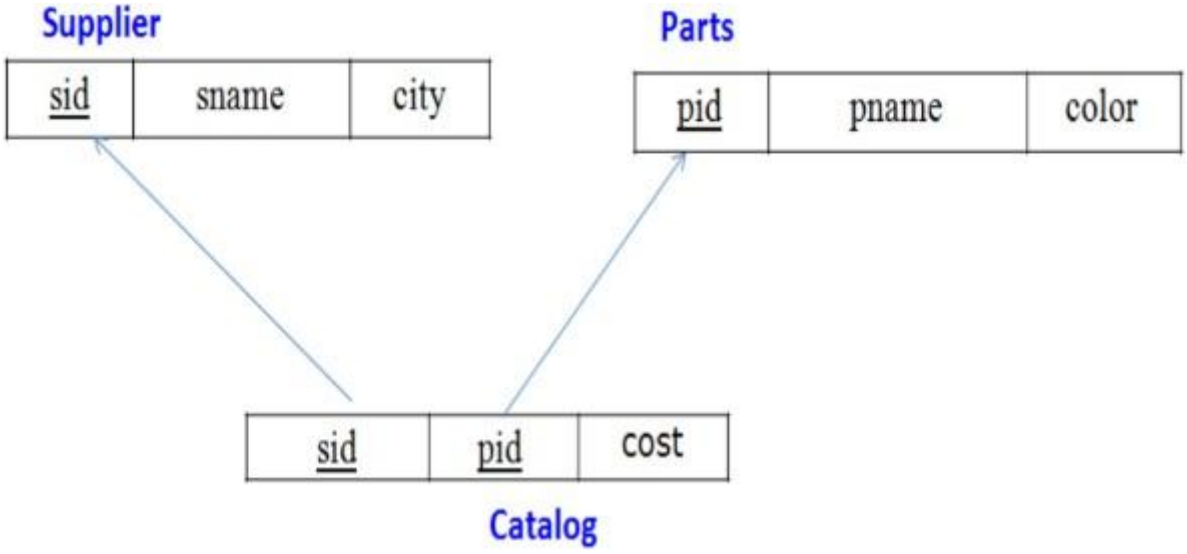
5. Find the snames of suppliers who supply every red part.

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 DB:

```
create database sup_170;
use  sup_170;
```

Create Table:

```
create table supplier(
sid int,
sname varchar(20),
```

```
city varchar(20),
primary key(sid));

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

create table catalog(
sid int,pid int, cost int,
foreign key(sid) references supplier(sid),
foreign key(pid) references parts(pid));
```

Inserting values:

```
 insert into Supplier values (10001, 'Acme Widget','Bangalore');
insert into Supplier values (10002, 'Johns','Kolkata');
insert into Supplier values (10003, 'Vimal','Mumbai');
insert into Supplier values (10004, 'Reliance','Delhi');

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

insert into Catalog values (10001, 20001 , 10);
insert into Catalog values (10001, 20002 , 10);
insert into Catalog values (10001, 20003 , 30);
insert into Catalog values (10001, 20004 , 10);
insert into Catalog values (10001, 20005 , 10);
insert into Catalog values (10002, 20001 , 10);
insert into Catalog values (10002, 20002 , 20);
insert into Catalog values (10003, 20003 , 30);
insert into Catalog values (10004, 20003 , 40);


select * from supplier;
```

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

```
select * from supplier;
```

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

select * from supplier;

	sid	pid	cost
▶	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

Queries

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

select distinct p.pname from parts p , catalog c where p.pid=c.pid;

	pname
▶	Book
	Pen
	Pencil
	Mobile
	Charger

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

select s.sname from supplier s
where NOT exists(select p.pid from
parts p where not exists (select c.sid from catalog c where c.sid=s.sid and c.pid=p.pid));

	sname
▶	Acme Widget

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

SELECT S.SNAME FROM SUPPLIER S
WHERE NOT EXISTS(SELECT P.PID FROM PARTS P
WHERE P.COLOR="RED" AND NOT EXISTS (
SELECT C.SID FROM CATALOG C WHERE C.SID=S.SID AND C.PID=P.PID));

	SNAME
▶	Acme Widget
	Johns

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

```
select p.pname from parts p,catalog c,supplier s where
p.pid=c.Pid and c.sid=s.sid and s.sname="acme widget" and not exists (
select * from catalog c1,supplier s1 where p.pid=c1.Pid AND C1.SID=s1.sid and s1.sname!="acme widget");
```

	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 where c.cost>(select avg(c1.cost) from catalog c1 where c1.pid=c.pid);
```

	sid
▶	10002
	10004

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

```
SELECT P.PID , S.SNAME FROM PARTS P,SUPPLIER S,CATALOG c
WHERE C.PID=P.PID AND C.SID=S.SID AND C.COST=
(SELECT MAX(C1.COST) from CATALOG C1 WHERE C1.PID=P.PID);
```

	PID	SNAME
▶	20001	Acme Widget
	20004	Acme Widget
	20005	Acme Widget
	20001	Johns
	20002	Johns
	20003	Reliance

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"**},{**rollno:**2,**a ge:**22,**cont:**9976,**email:"sohan@gmail.com"**},
{**rollno:**3,**age:**21,**cont:**5576,**email:"farhan@gmail.com"**},
{**rollno:**4,**age:**20,**cont:**4476,**email:"sakshi@gmail.com"**},{**rollno:**5,**age:**23,**cont:**2276,**email:"sa nika@gmail.com"**}]);

db.student.find()

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.find()
[
  {
    _id: ObjectId("6746b87366152224f4779211"),
    RollNo: 1,
    Age: 21,
    Const: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8ac66152224f4779212"),
    RollNo: 2,
    Age: 22,
    Const: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8d266152224f4779213"),
    RollNo: 3,
    Age: 21,
    Const: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8f166152224f4779214"),
    RollNo: 10,
    Age: 20,
    Const: 2276,
    email: 'rekha.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"}})

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.update({RollNo:10},{$set:{email:"Abhinav@gmail.com"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.find()
[
  {
    _id: ObjectId("6746b87366152224f4779211"),
    RollNo: 1,
    Age: 21,
    Const: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8ac66152224f4779212"),
    RollNo: 2,
    Age: 22,
    Const: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8d266152224f4779213"),
    RollNo: 3,
    Age: 21,
    Const: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8f166152224f4779214"),
    RollNo: 10,
    Age: 20,
    Const: 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"}})

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.find()
[
  {
    _id: ObjectId("6746b87366152224f4779211"),
    RollNo: 1,
    Age: 21,
    Const: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8ac66152224f4779212"),
    RollNo: 2,
    Age: 22,
    Const: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8d266152224f4779213"),
    RollNo: 3,
    Age: 21,
    Const: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8f166152224f4779214"),
    RollNo: 10,
    Age: 20,
    Const: 2276,
    email: 'Abhinav@gmail.com'
  },
  {
    _id: ObjectId("6746ba1266152224f4779215"),
    RollNo: 11,
    Age: 22,
    Const: 2276,
    email: 'rea.de9@gmail.com',
    Name: 'FEM'
  }
]
```

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"}]);

db.student.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'
  }
]
```

Finding all checking accounts with balance greater than 12000

db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});

```
test> db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});
[
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4c'),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4d'),
    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-10jjz6-shard-0 [primary] test> db.Customer.aggregate([{$group:{_id:"$custid", minBal:{$min:"$acc_bal"}, maxBal:
... {$max:"$acc_bal"}}}]);
[
  { _id: 1, minBal: 10000, maxBal: 20000 },
  { _id: 4, minBal: 10000, maxBal: 10000 },
  { _id: 3, minBal: 50000, maxBal: 50000 },
  { _id: 5, minBal: 2000, maxBal: 2000 }
]
```

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

mongoexport mongodb+srv://Likhith:@cluster0.xbmgopf.mongodb.net/test --collection=Customer -- out D:\1BM23CS170\st.json

50

- **Dropping collection "Customer"**

db.Customer.drop();

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

mongoimport mongodb+srv://Likhith:@cluster0.xbmgopf.mongodb.net/test --collection=Customer --file D:\1BM23CS170\st.json

```
D:\1BM23CS170\DBMS\mongodb-database-tools-windows-x86_64-100.10.0\bin>mongoexport mongodb+srv://Likhith:@cluster0.r5p8c.mongodb.net/test --collection=student --out D:\1BM23CS170\st.json
Enter password for mongo user:

2024-12-04T12:48:54.089+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.r5p8c.mongodb.net/test
2024-12-04T12:48:54.267+0530    exported 5 records

D:\1BM23CS170\DBMS\mongodb-database-tools-windows-x86_64-100.10.0\bin>mongoimport mongodb+srv://Likhith:@cluster0.r5p8c.mongodb.net/test --collection=Customer --file D:\1BM23CS170\st.json
Enter password for mongo user:

2024-12-04T12:55:38.858+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.r5p8c.mongodb.net/test
2024-12-04T12:55:38.924+0530    5 document(s) imported successfully. 0 document(s) failed to import.
```

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'
  }
]
```

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.

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

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

Creating Table:

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

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-10jjz6-shard-0 [primary] test> db.restaurants.insertMany([
... {name:"Meghna Foods",town:"Jayanagar", cuisine: "Indian", score: 8, addraddress: { 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, addreaddress: { zipcode: "10300", street: "Majestic" } },
... { name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram" }}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6751f5566a59c75535ff9944"),
    '1': ObjectId("6751f5566a59c75535ff9945"),
    '2': ObjectId("6751f5566a59c75535ff9946"),
    '3': ObjectId("6751f5566a59c75535ff9947"),
    '4': ObjectId("6751f5566a59c75535ff9948")
  }
}
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({})
[
  {
    _id: ObjectId("6751f5566a59c75535ff9944"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9945"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9946"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
```

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

```
  {
    _id: ObjectId("6751f5566a59c75535ff9947"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9948"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({}).sort({ name: -1 })
[
  {
    _id: ObjectId("6751f5566a59c75535ff9948"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9944"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9947"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
```

3) db.Restraunt.find(
{ "grades.score": { $lte: 10 } },

{ _id: 1, name: 1, town: 1, cuisine: 1, restaurant_id: 1 }
);

```
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9946"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })
[
  {
    _id: ObjectId("6751f5566a59c75535ff9944"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9945"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9947"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9948"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
```

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

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }
... ])
[
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Empire', average_score: 7 }
]
Atlas atlas-10jjz6-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: 'MG Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momos', address: { street: 'Malleshwaram' } }
]
```

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


```
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }
... ])
[
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Empire', average_score: 7 }
]
Atlas atlas-10jjz6-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: 'MG Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momos', address: { street: 'Malleshwaram' } }
]
```