VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems

(23CS3PCDBM)

Submitted by

Nalluri Sindhuja (1BM24CS179)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

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 **Nalluri Sindhuja(1BM24CS179),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. 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 Assistant Professor Department of CSE, BMSCE	Dr. Joythi S Nayak 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)

1. Create the above tables by properly specifying the primary keys and the foreign keys.

2.Enter at least five tuples for each relation

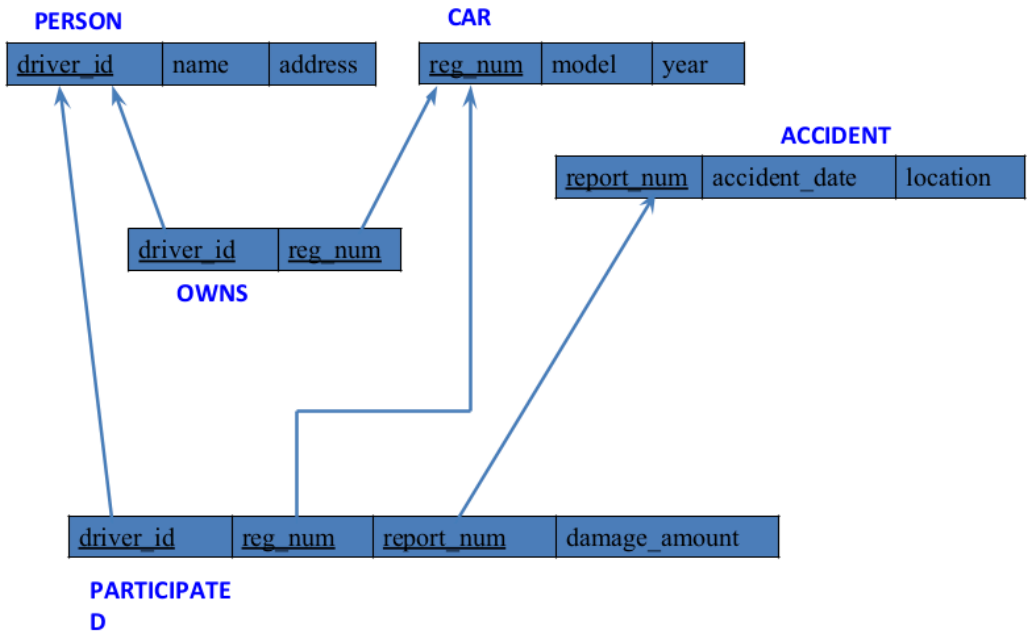
3.Display Accident date and location

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

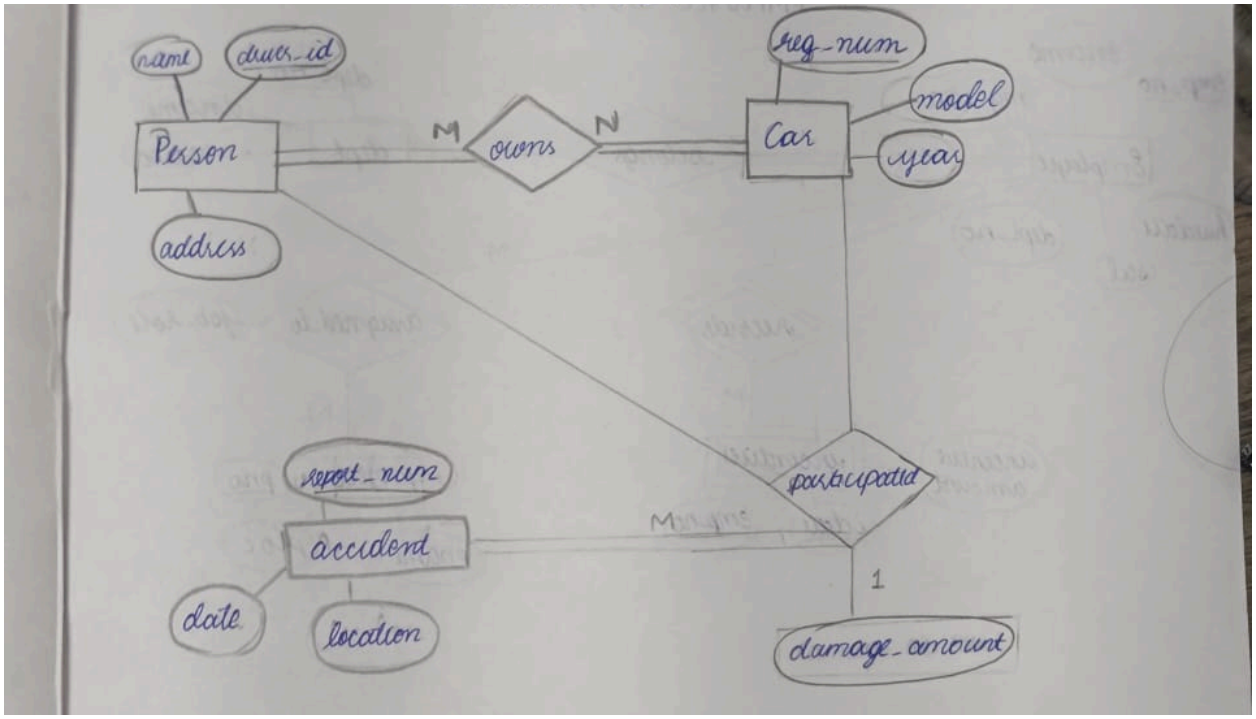
5.Add a new accident to the database.

6.Display driver id who did accident with damage amount greater than or equal to Rs.25000

Schema Diagram



ER Diagram:



Create Database:

```
Create database insurance;
Use insurance;
```

Create Tables:

```
create table person(
driver_id varchar(10) PRIMARY KEY,
name varchar(10),
address varchar(10)
);


create table car(
 reg_num varchar(10) PRIMARY KEY,
model varchar(10),
year INT);
```

```
create table accident(
report_num int PRIMARY KEY,
accident_date date ,
location varchar(10));

create table owns(
driver_id varchar(10) ,
reg_num varchar(10) ,
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,
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(10)	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", "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
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 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
*	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,'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
*	NULL	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

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
	2004-03-05	Kanakpura Road

2. **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;

   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
*	NULL	NULL	NULL	NULL

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

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

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
	16	2008-03-08	Domlur
*	NULL	NULL	NULL

4. **Display driver id who did accident with damage amount greater than or equal to Rs.25000**

select distinct driver_id

from participated

where damage_amount >= 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)

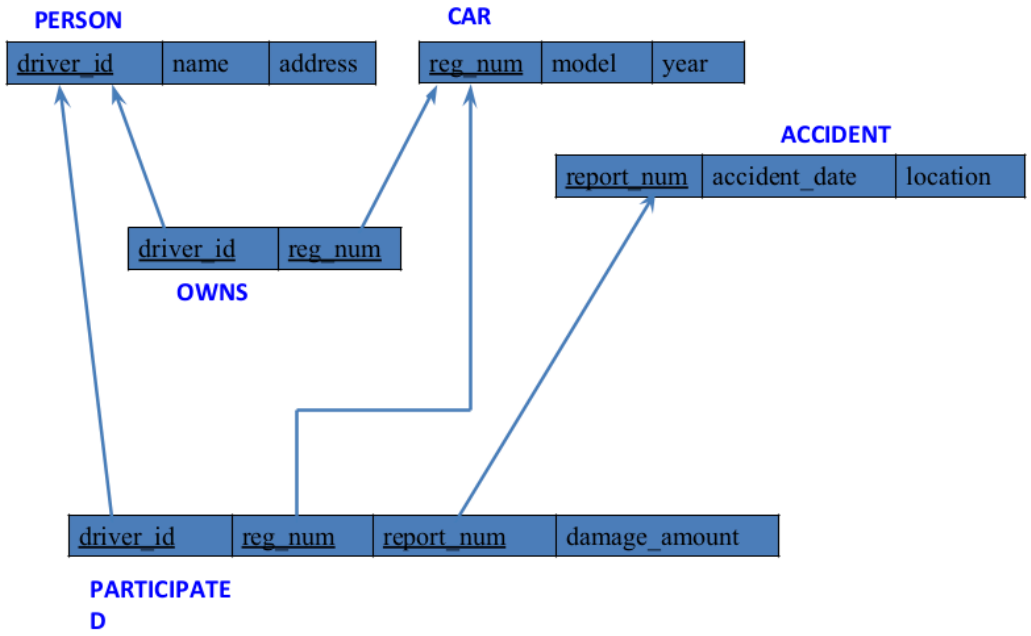
Create the above tables by properly specifying the primary keys and the foreign keys as done in “Program-1”week’s lab and Enter at least five tuples for each relation.

1. Display the entire CAR relation in the ascending order of manufacturing year.
2. Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
3. Find the total number of people who owned cars that were involved in accidents in 2008.
4. List the entire participated relation in the Descending Order of Damage Amount.
5. Find the Average Damage Amount.
6. Delete the tuple whose Damage Amount is below the Average Damage Amount

7 List the names of drivers whose Damage is Greater than the Average Damage Amount.

8. Find Maximum Damage Amount.

Schema Diagram:



Queries

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

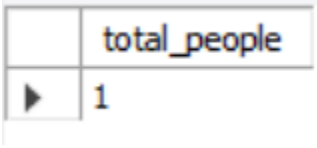
2. **Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.**

   select count(report_num) CNT from car c,participated p where c.reg_num=p.reg_num and model='Lancer';

	CNT
▶	1

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

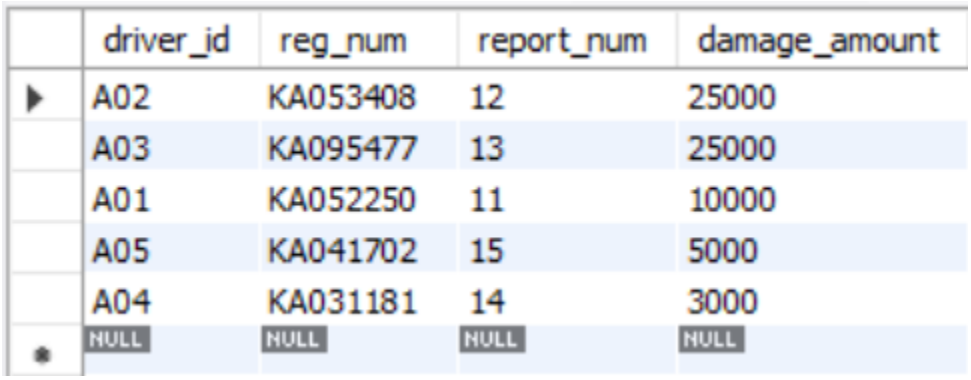
   select count(distinct driver_id) as total_people from participated a, accident b where a.report_num=b.report_num and b.accident_date like '2008-%';



	total_people
▶	1

4. **List the entire participated relation in the Descending Order of Damage Amount.**

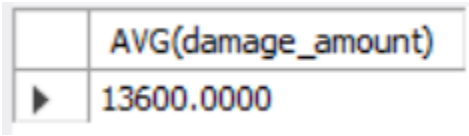
   SELECT * FROM participated ORDER BY damage_amount DESC;



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

5. **Find the Average Damage Amount.**

   SELECT AVG(damage_amount) FROM participated;



	AVG(damage_amount)
▶	13600.0000

6. **Delete the tuple whose Damage Amount is below the Average Damage Amount**

```
DELETE P FROM participated P
JOIN (
   SELECT AVG(damage_amount) AS avg_damage
   FROM PARTICIPATED
) A
ON P.damage_amount < A.avg_damage;
select * from participated;
```

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

7. **List the names 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

8. **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)

1.Create the above tables by properly specifying the primary keys and the foreign keys.

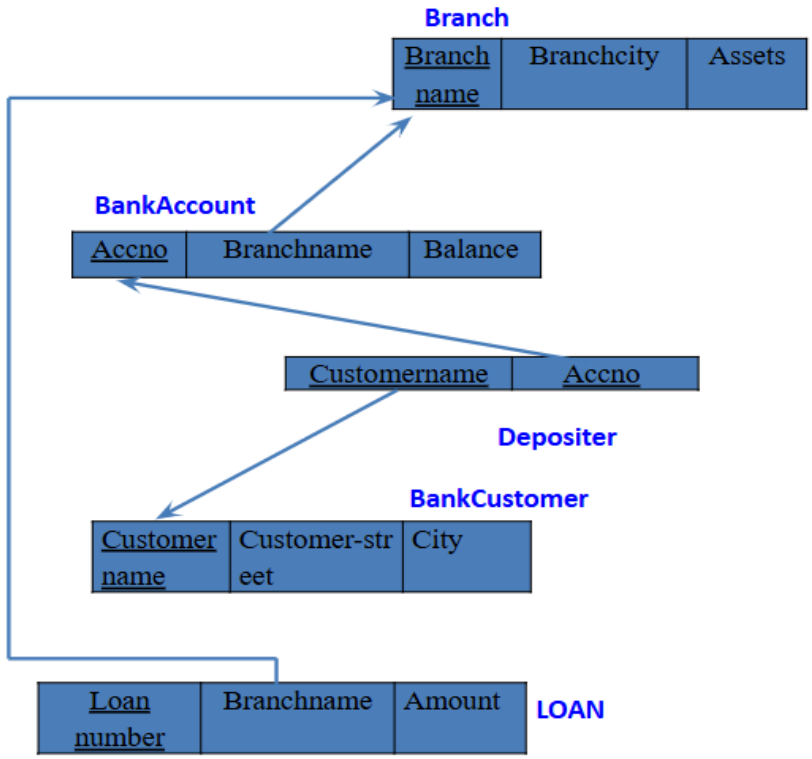
2.Enter at least five tuples for each relation.

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

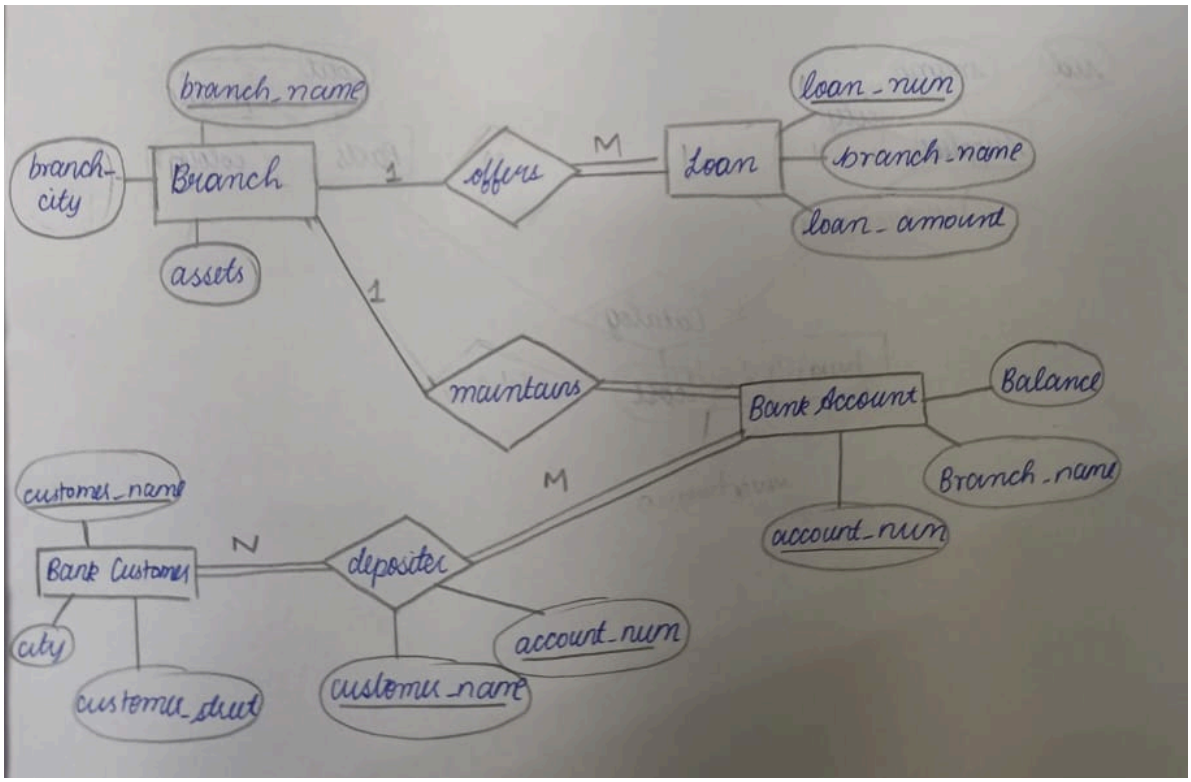
4.Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).

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

Schema Diagram:



ER Diagram



Create Database:

```
create database bank;
use bank;
```

Create table

```
create table branch (
branch_name varchar(30) ,
branch_city varchar(25),
assets int , primary key(branch_name));

create table bank_account(
accno int,
branch_name varchar(30),
balance int,
PRIMARY KEY(accno), foreign key (branch_name) references branch(branch_name));
create table bank_customer(
customer_name varchar(20),
customer_street varchar(30),
customer_city varchar (35),
PRIMARY KEY(customer_name));

create table depositer(
customer_name varchar(20),
accno int,
PRIMARY KEY(customer_name,accno),
foreign key (accno) references bank_account(accno),
foreign key (customer_name) references bank_customer(customer_name));

create table loan(
```

loan_number int,

branch_name varchar(30),

amount int,

PRIMARY KEY(loan_number),

foreign key (branch_name) references branch(branch_name));

Structure of the table

desc branch;

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

desc bank_account;

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

desc bank_customer;

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

desc depositor;

Field	Type	Null	Key	Default	Extra
customer_name	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	

Inserting Values to the table

insert into branch values('SBI_Chamrajpet','Bangalore',50000),

('SBI_ResidencyRoad','Bangalore',10000),

('SBI_ShivajiRoad','Bombay',20000),

('SBI_ParlimentRoad','Delhi',10000),

('SBI_Jantarmantar','Delhi',20000);

Select * from branch;

branch_name	branch_city	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

insert into bank_account values(1,'SBI_Chamrajpet',2000);

insert into bank_account values(2,'SBI_ResidencyRoad',5000);

insert into bank_account values(3,'SBI_ShivajiRoad',6000);

insert into bank_account values(4,'SBI_ParlimentRoad',9000);

insert into bank_account values(5,'SBI_Jantarmantar',8000);

insert into bank_account values(6,'SBI_ShivajiRoad',4000);

insert into bank_account values(8,'SBI_ResidencyRoad',4000);

insert into bank_account values(9,'SBI_ParlimentRoad',3000);

insert into bank_account values(10,'SBI_ResidencyRoad',5000);

insert into bank_account values(11,'SBI_Jantarmantar',2000);

select * from bank_account;

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

insert into bank_customer values('Avinash','Bull_Temple_Road','Bangalore');

insert into bank_customer values('Dinesh','Bannergatta_Road','Bangalore');

insert into bank_customer values('Mohan','NationalCollege_Road','Bangalore');

insert into bank_customer values('Nikil','Akbar_Road','Delhi');

insert into bank_customer values('Ravi','Prithviraj_Road','Delhi');

customer_name	customer_street	customer_city
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_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('Mohan',3);
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;
```

customer_name	accno
Avinash	1
Dinesh	2
Mohan	3
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11

```
insert into loan values(1,'SBI_Chamrajpet',1000);
insert into loan values(2,'SBI_ResidencyRoad',2000);
insert into loan values(3,'SBI_ShivajiRoad',3000);
insert into loan values(4,'SBI_ParlimentRoad',4000);
insert into loan values(5,'SBI_Jantarmantar',5000);
select * from loan;
```

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

Queries

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

   select branch_name,CONCAT(assets/100000,'Lakhs') as assets_in_lakha from branch ;

	branch_name	assets_in_lakha
▶	SBI_Chamrajpet	0.5000Lakhs
	SBI_Jantarmantar	0.2000Lakhs
	SBI_ParlimentRoad	0.1000Lakhs
	SBI_ResidencyRoad	0.1000Lakhs
	SBI_ShivajiRoad	0.2000Lakhs

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

   select d.customer_name from depositer d, bank_account b where

   b.branch_name='SBI_ResidencyRoad' and d.accno=b.accno group by

   d.customer_name having count(d.accno)>=2;

	customer_name
▶	Dinesh

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

```
create view sum_of_loan
as select branch_name, SUM(Balance)
from bank_account
group by branch_name;
select * from sum_of_loan;
```

branch_name	SUM(Balance)
SBI_Chamrajpet	2000
SBI_Jantarmantar	10000
SBI_ParlimentRoad	12000
SBI_ResidencyRoad	14000
SBI_ShivajiRoad	10000

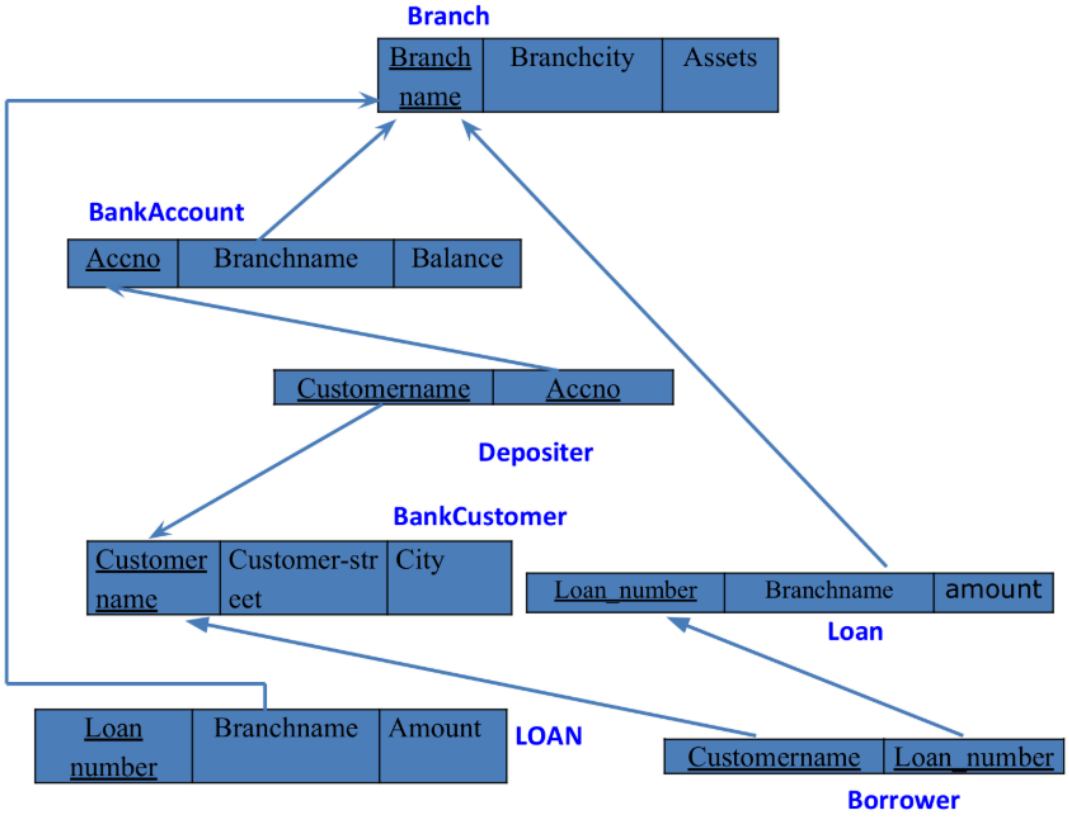
More Queries On BankDatabase

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)

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

Schema Diagram:



Create table

```
create table borrower (
        customer_name VARCHAR(50),
        loan_number INT,
        PRIMARY KEY (customer_name, loan_number),
        FOREIGN KEY (customer_name) REFERENCES
bank_customer(customer_name),
        FOREIGN KEY (loan_number) REFERENCES loan(loan_number)
);
```

Structure of table

```
desc borrower;
```

Field	Type	Null	Key	Default	Extra
customer_name	varchar(50)	NO	PRI	NULL	
loan_number	int	NO	PRI	NULL	

Insert values into table

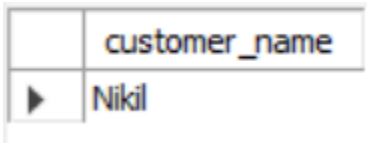
```
insert into borrower values ("Avinash", 1),
("Dinesh", 2),
("Mohan", 3),
("Nikil", 4),
("Ravi", 5);
select * from borrower;
```

customer_name	loan_number
Avinash	1
Dinesh	2
Mohan	3
Nikil	4
Ravi	5
NULL	NULL

Queries

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

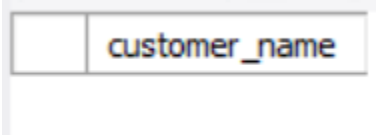
```
select d.customer_name
from depositer d
join bank_account ba
on d.accno=ba.accno
join branch b
on b.branch_name=ba.branch_name
WHERE b.branch_city = 'Delhi'
group by d.customer_name
HAVING COUNT(DISTINCT ba.branch_name) = (
   SELECT COUNT(*) FROM branch WHERE branch_city = 'Delhi'
);
```



customer_name
Nikil

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

```
select distinct customer_name from borrower
where customer_name not in (select customer_name from depositer);
```

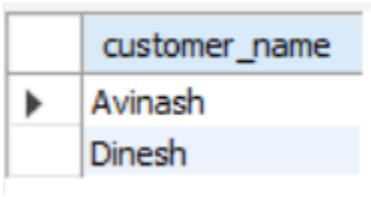


customer_name

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

```
SELECT DISTINCT bc.customer_name FROM bank_customer bc
JOIN depositer d ON bc.customer_name = d.customer_name
JOIN bank_account ba ON d.accno = ba.accno
JOIN borrower br ON bc.customer_name = br.customer_name
```

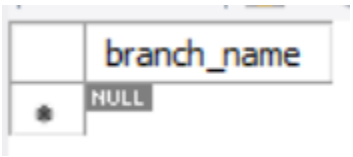
```
JOIN loan l ON br.loan_number = l.loan_number
JOIN branch b ON b.branch_name = ba.branch_name
WHERE b.branch_city = 'Bangalore'
AND l.branch_name = ba.branch_name;
```



customer_name
Avinash
Dinesh

4. **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

5. **Demonstrate how you delete all account tuples at every branch located in a specific city**

```
DELETE FROM depositer
WHERE accno IN (SELECT accno
FROM bank_account
WHERE branch_name IN (
SELECT branch_name
FROM branch
WHERE branch_city = 'Bombay'));
DELETE FROM bank_account
WHERE branch_name IN (
SELECT branch_name
FROM branch
WHERE branch_city = 'Bombay');
```

select * from bank_account;

	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
*	NULL	NULL	NULL

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

UPDATE BANK_ACCOUNT

SET balance = balance * 1.05;

	accno	branch_name	balance
▶	1	SBI_Chamrajpet	2100
	2	SBI_ResidencyRoad	5250
	4	SBI_ParlimentRoad	9450
	5	SBI_Jantarmantar	8400
	8	SBI_ResidencyRoad	4200
	9	SBI_ParlimentRoad	3150
	10	SBI_ResidencyRoad	5250
	11	SBI_Jantarmantar	2100
*	NULL	NULL	NULL

Employee Database

(WEEK 5)
Question

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

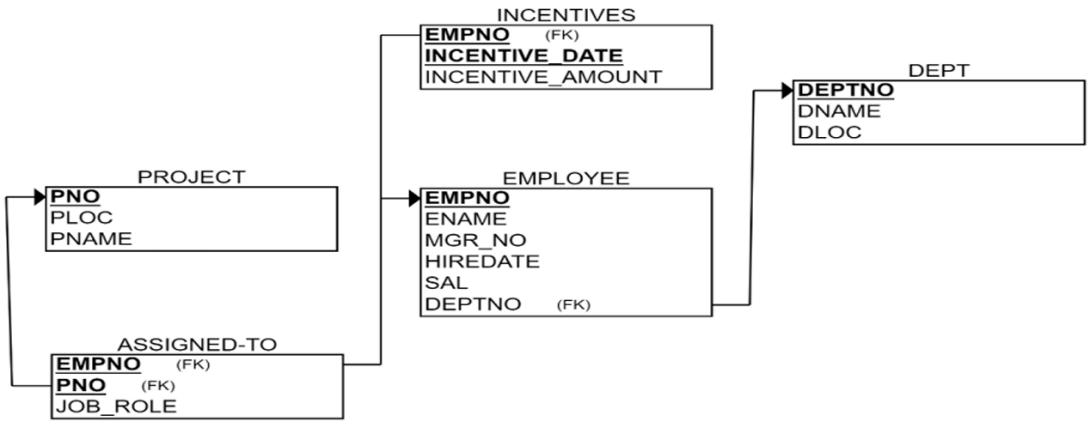
2.Enter greater than five tuples for each table.

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

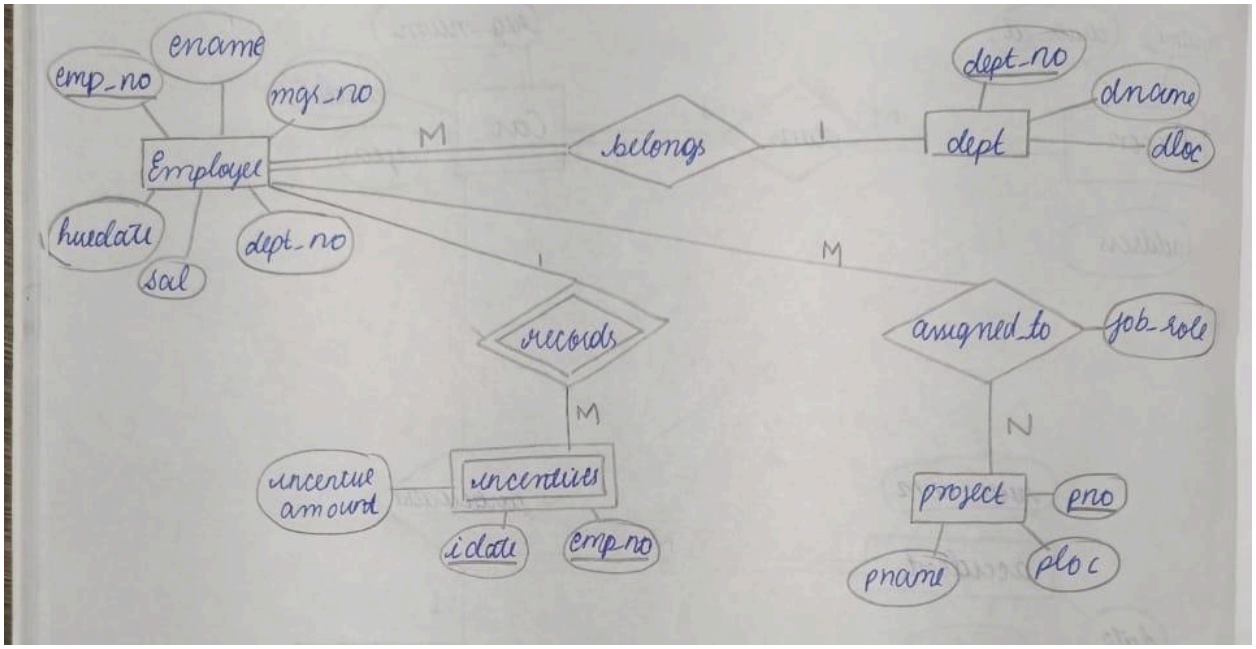
4.Get Employee ID's of those employees who didn't receive incentives

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



ER Diagram



Create database

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

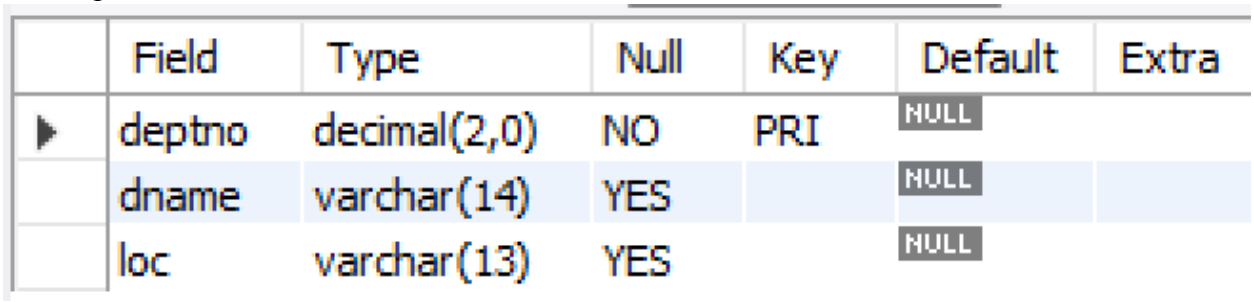
Create tables

```
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),
foreign key (deptno) 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)
);
```

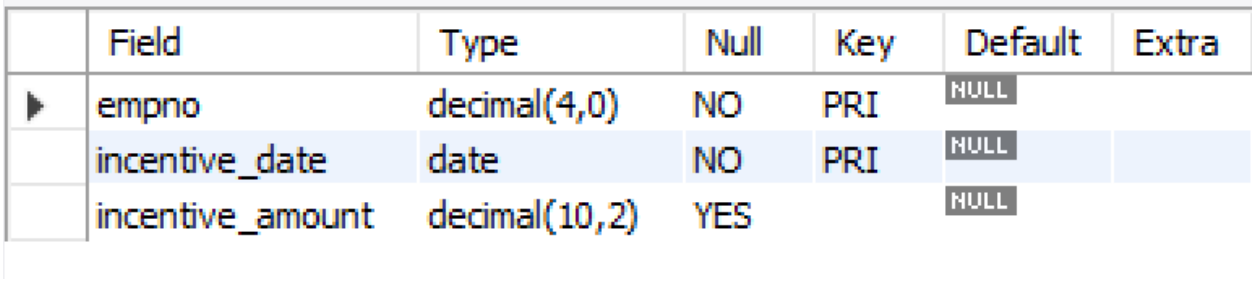
Structure of the table

desc dept;



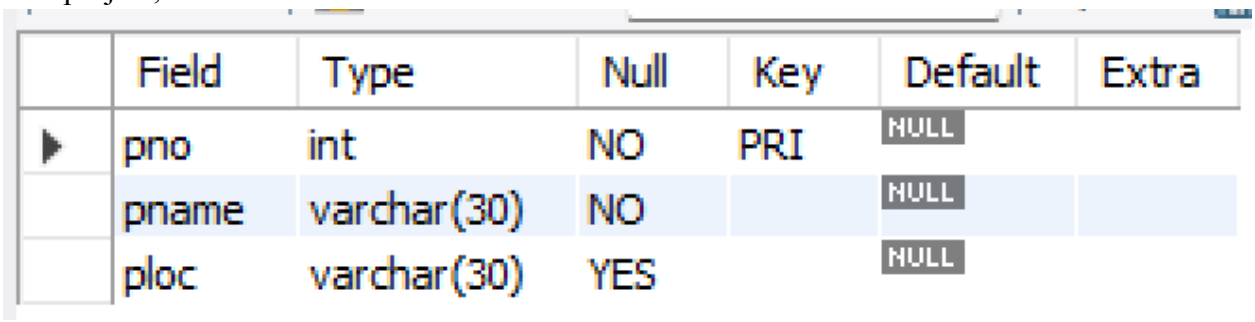
Field	Type	Null	Key	Default	Extra
deptno	decimal(2,0)	NO	PRI	NULL	
dname	varchar(14)	YES		NULL	
loc	varchar(13)	YES		NULL	

desc incentives;



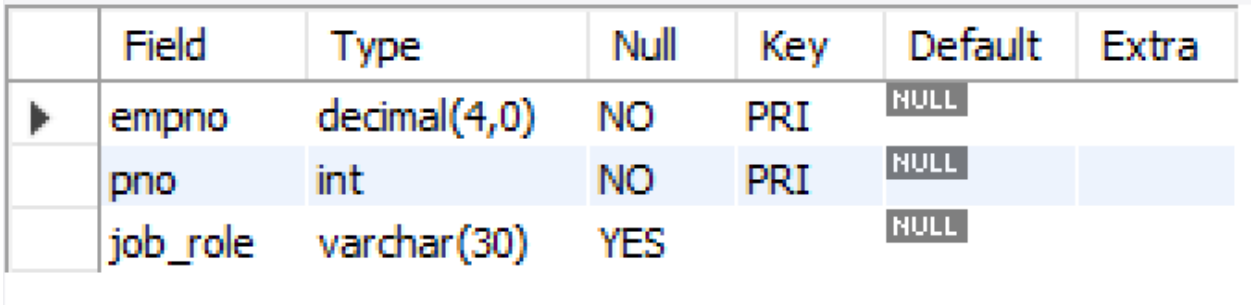
Field	Type	Null	Key	Default	Extra
empno	decimal(4,0)	NO	PRI	NULL	
incentive_date	date	NO	PRI	NULL	
incentive_amount	decimal(10,2)	YES		NULL	

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 to the table

```
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','99999.99','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	99999.99	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');

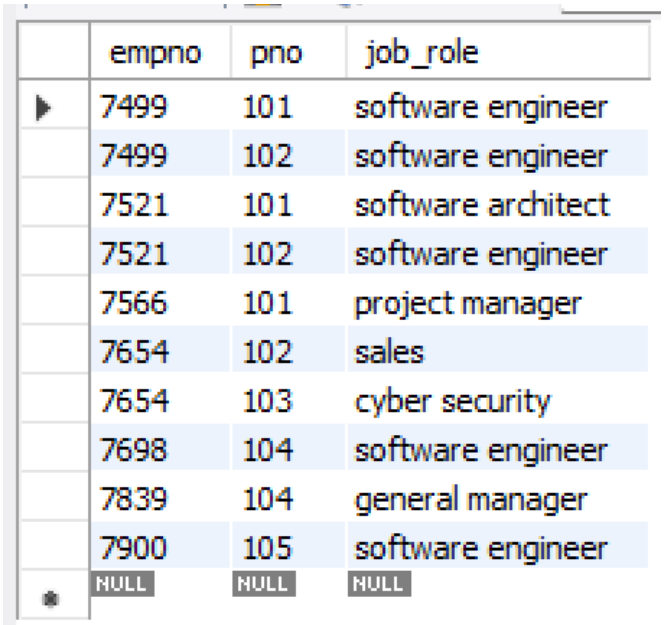
insert into project values (104,'data science','mysuru');

insert into project values (105,'autonomus systems','pune');

select * from project;

	pno	pname	ploc
▶	101	ai project	bengaluru
	102	iot	hyderabad
	103	blockchain	bengaluru
	104	data science	mysuru
	105	autonomus 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

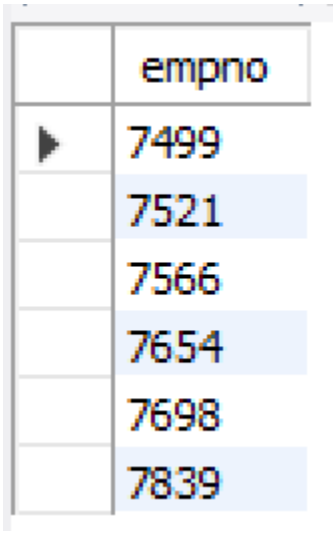
Queries

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

   select distinct e.empno from emp e join assigned_to a on e.empno=a.empno

   join project p on p.pno=a.pno

   where ploc in('bengaluru','hyderabad','mysuru');

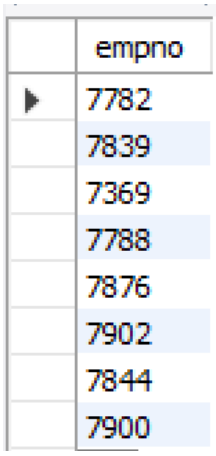


	empno
▶	7499
	7521
	7566
	7654
	7698
	7839

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

   select e.empno from emp e where e.empno not in

   (select empno from incentives);



	empno
▶	7782
	7839
	7369
	7788
	7876
	7902
	7844
	7900

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

```
select e.ename,e.empno,d.dname,a.job_role,d.loc as department_location,
p.ploc as project_location from emp e
join assigned_to a on e.empno=a.empno
join project p on a.pno=p.pno
join dept d on e.deptno=d.deptno
where p.ploc=d.loc;
```

	ename	empno	dname	job_role	department_location	project_location
▶	tanvir	7566	research	project manager	bengaluru	bengaluru

More Queries On Employee Database

(WEEK 6)
Question

1. Using Scheme diagram (under Program-5), 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 manager is working.

Queries

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

```
select m.ename,count(*)
from emp e,emp m
where e.mgr_no=m.empno
group by m.ename
having count(*) = (select max(mycount)
from (Select count(*)mycount
from emp
group by mgr_no)a);
```

	ename	count(*)
▶	kumar	5

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

```
select * from emp m

where m.empno in

(select mgr_no from emp)and

m.sal>(select avg(e.sal) from emp e

where e.mgr_no=m.empno);
```

empno	ename	mgr_no	hiredate	sal	deptno
7698	kumar	7839	2015-05-01	28500.00	30
7839	king	NULL	2009-11-17	99999.99	10
7788	scott	7566	2010-12-09	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

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

```
SELECT e.deptno, e.ename

FROM emp e WHERE e.sal = (SELECT MAX(sal)

FROM emp WHERE deptno = e.deptno

AND sal < (SELECT MAX(sal)  FROM emp WHERE deptno = e.deptno      ));
```

deptno	ename
30	shruthi
10	clark
20	scott
20	ford

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

```
SELECT e.*

FROM emp e

JOIN incentives i ON e.empno = i.empno

WHERE i.incentive_date BETWEEN '2019-01-01' AND '2019-01-31'

ORDER BY i.incentive_amount DESC

LIMIT 1 OFFSET 1;
```

	empno	ename	mgr_no	hiredate	sal	deptno

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

```
select * from emp e where e.deptno=

(select m.deptno from emp m

where m.empno=e.mgr_no);
```

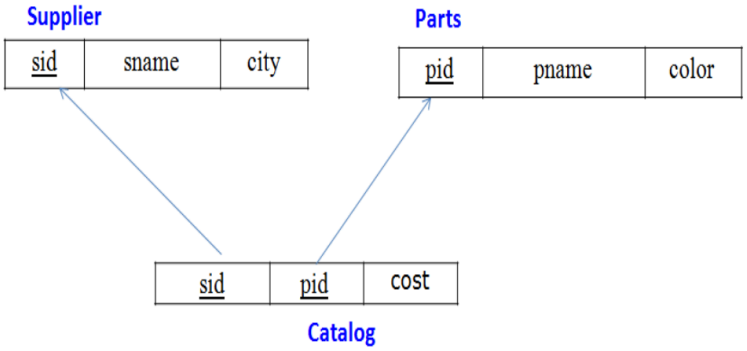
	empno	ename	mgr_no	hiredate	sal	deptno
▶	7369	adarsh	7902	2012-12-17	80000.00	20
	7499	shruthi	7698	2013-02-20	16000.00	30
	7521	anvitha	7698	2015-02-22	12500.00	30
	7654	ramesh	7698	2014-09-28	12500.00	30
	7782	clark	7839	2017-06-09	24500.00	10
	7788	scott	7566	2010-12-09	30000.00	20
	7844	turner	7698	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

Supplier Database

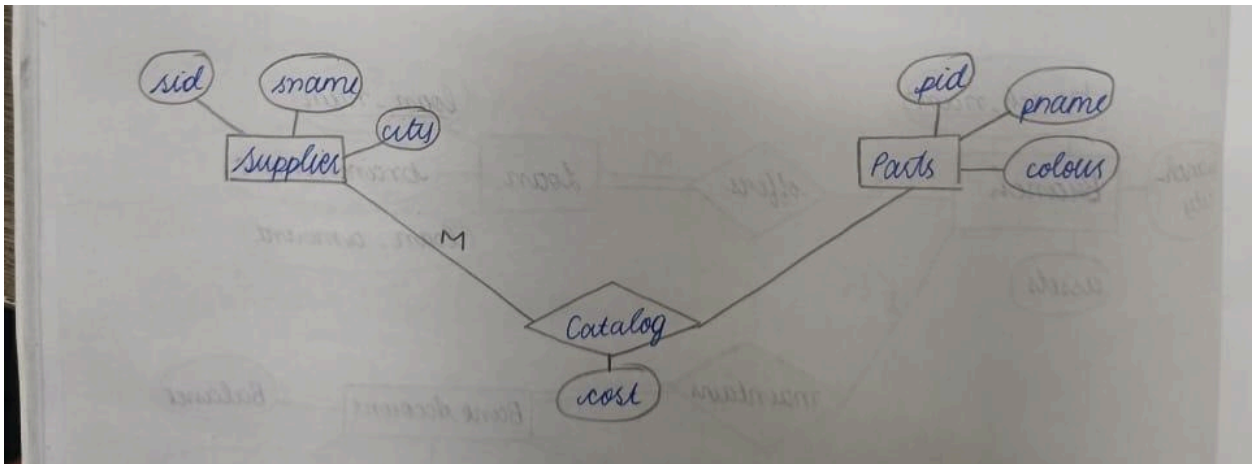
Question
(Week 7)

1.Using a Schema 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



ER Diagram



Create database

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

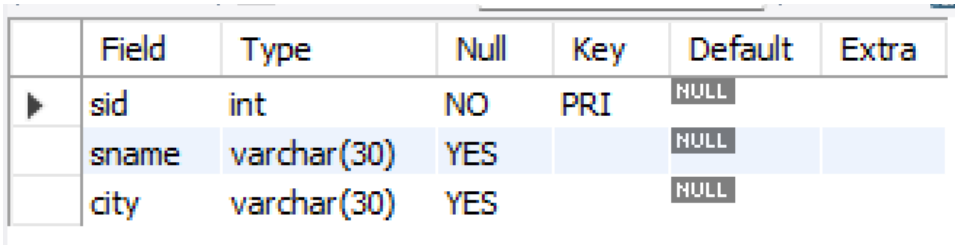
Create table

```
create table suppliers (
    sid int primary key,
    sname varchar(30),
    city varchar(30)
);

create table parts (
    pid int primary key,
    pname varchar(30),
    color varchar(20)
);
create table catalog (
    sid int,
    pid int,
```

```
    cost int,
    primary key (sid, pid),
    foreign key (sid) references suppliers(sid),
    foreign key (pid) references parts(pid)
);
```

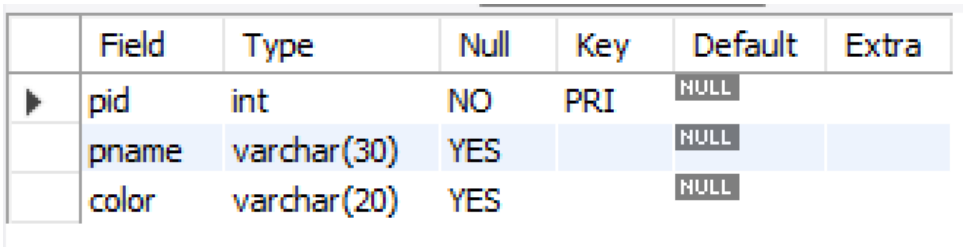
Structure of the table

desc suppliers;



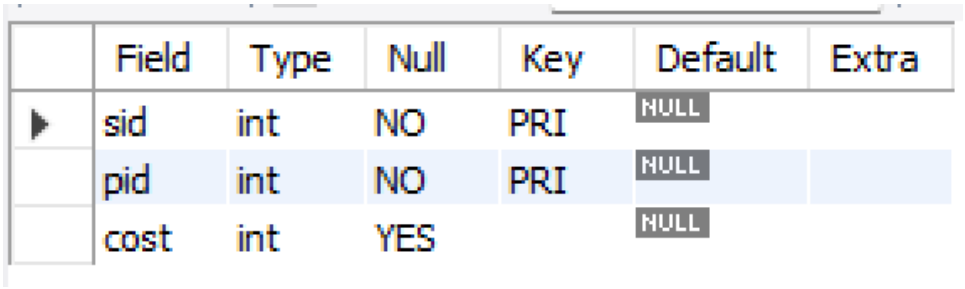
	Field	Type	Null	Key	Default	Extra
▶	sid	int	NO	PRI	NULL	
	sname	varchar(30)	YES		NULL	
	city	varchar(30)	YES		NULL	

desc parts;



	Field	Type	Null	Key	Default	Extra
▶	pid	int	NO	PRI	NULL	
	pname	varchar(30)	YES		NULL	
	color	varchar(20)	YES		NULL	

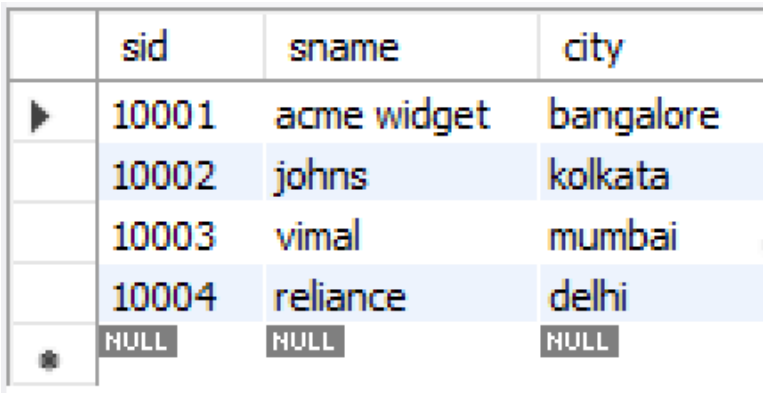
desc catalog;



	Field	Type	Null	Key	Default	Extra
▶	sid	int	NO	PRI	NULL	
	pid	int	NO	PRI	NULL	
	cost	int	YES		NULL	

Inserting Values to the table

```
insert into suppliers values
(10001, 'acme widget', 'bangalore'),
(10002, 'johns', 'kolkata'),
(10003, 'vimal', 'mumbai'),
(10004, 'reliance', 'delhi');
select * from suppliers;
```



sid	sname	city
10001	acme widget	bangalore
10002	johns	kolkata
10003	vimal	mumbai
10004	reliance	delhi
NULL	NULL	NULL

```
insert into parts values
(20001, 'book', 'red'),
(20002, 'pen', 'red'),
(20003, 'pencil', 'green'),
(20004, 'mobile', 'green'),
(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),

(10001, 20002, 10),

(10001, 20003, 30),

(10001, 20004, 10),

(10001, 20005, 10),

(10002, 20001, 10),

(10002, 20002, 20),

(10003, 20003, 30),

(10004, 20003, 40);

select* from catalog;

	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
*	NULL	NULL	NULL

Queries

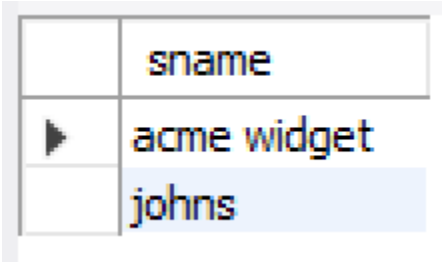
1. **Find the pnames of parts for which there is some supplier.**
   select distinct p.pname from parts p join catalog c on p.pid = c.pid;

	pname
▶	book
	pen
	pencil
	mobile
	charger

2. **Find the snames of suppliers who supply every part.**
   select s.sname from suppliers s where not exists (select * from parts p where not exists (select * from catalog c where c.sid = s.sid and c.pid = p.pid));

	sname
▶	acme widget

3. **Find the snames of suppliers who supply every red part.**
   select s.sname from suppliers s where not exists (select * from parts p where p.color ='red' and not exists (select * from catalog c where c.sid = s.sid and c.pid = p.pid));



	sname
▶	acme widget
	johns

4. **Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.**
   select p.pname from parts p

   join catalog c1 on p.pid = c1.pid

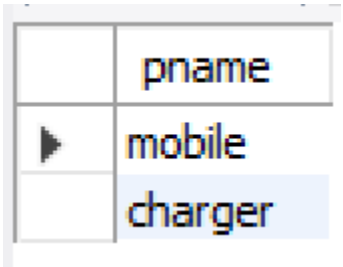
   join suppliers s on s.sid = c1.sid

   where s.sname = 'acme widget'

   and not exists (

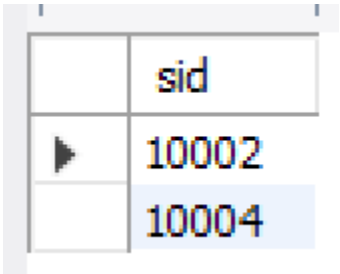
   select * from catalog c2 where c2.pid = p.pid and c2.sid != c1.sid

   );



	pname
▶	mobile
	charger

5. **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) a on c.pid = a.pid where c.cost > a.avg_cost;



	sid
▶	10002
	10004

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

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

pname	sname
book	acme widget
mobile	acme widget
charger	acme widget
book	johns
pen	johns
pencil	reliance

NoSQL Student Database

Question
(Week 8)

Perform the following DB operations using MongoDB.

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

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.Drop the table

Queries

1. **Create a database “Student” with the following attributes Rollno, Age, ContactNo, EmailId.**

   db.createCollection("Student");

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
```

2. **Insert appropriate values**

   db.Student.insertMany([

   {RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"},

   {RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"},

   {RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"},

   {RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"},

   {RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"}

   ])

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('694bbc07bb458fdad91e2621'),
    '1': ObjectId('694bbc07bb458fdad91e2622'),
    '2': ObjectId('694bbc07bb458fdad91e2623'),
    '3': ObjectId('694bbc07bb458fdad91e2624'),
    '4': ObjectId('694bbc07bb458fdad91e2625')
  }
}
```

db.Student.find()

```
[
  {
    _id: ObjectId('694bbc99bb458fdad91e2626'),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId('694bbc99bb458fdad91e2627'),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId('694bbc99bb458fdad91e2628'),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId('694bbc99bb458fdad91e2629'),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId('694bbc99bb458fdad91e262a'),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

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

db.Student.update({RollNo:10},{$set:{email:"Abhinav@gmail.com"}})

```
Atlas atlas-r8p6bd-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
}
```

```
{
  _id: ObjectId('694bbc99bb458fdad91e262a'),
  RollNo: 10,
  Age: 23,
  Cont: 2276,
  email: 'Abhinav@gmail.com'
}
```

db.Student.insert({RollNo:11,Age:22,Name:"ABC",Cont:2276,
email:"rea.de9@gmail.com"});

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.Student.insert({RollNo:11,Age:22,Name:
... "ABC",Cont:2276,email:"rea.de9@gmail.com"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('694bbd13bb458fdad91e262b') }
}
```

```
{
  _id: ObjectId('694bbd13bb458fdad91e262b'),
  RollNo: 11,
  Age: 22,
  Name: 'ABC',
  Cont: 2276,
  email: 'rea.de9@gmail.com'
}
```

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

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

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId('694bbd13bb458fdad91e262b'),
  RollNo: 11,
  Age: 22,
  Name: 'FEM',
  Cont: 2276,
  email: 'rea.de9@gmail.com'
}
```

5. **Drop the table**

db.Student.drop();

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.Student.drop();
true
```

NoSQL Customer Database

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 ‘Z’ for each customer_id.

4.Determine Minimum and Maximum account balance for each customer_id.

5.Drop the table

Queries

1. **Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type**

   db.createCollection("Customers")

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.createCollection("Customers")
{ ok: 1 }
```

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

   db.Customers.insertMany([

   { Cust_id: 101, Acc_Bal: 500, Acc_Type: "Z" },

   { Cust_id: 101, Acc_Bal: 800, Acc_Type: "Z" },

   { Cust_id: 102, Acc_Bal: 300, Acc_Type: "A" },

   { Cust_id: 102, Acc_Bal: 700, Acc_Type: "Z" },

   { Cust_id: 103, Acc_Bal: 1500, Acc_Type: "Z" }

   ])

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('694bd9b244dc217d681e2621'),
    '1': ObjectId('694bd9b244dc217d681e2622'),
    '2': ObjectId('694bd9b244dc217d681e2623'),
    '3': ObjectId('694bd9b244dc217d681e2624'),
    '4': ObjectId('694bd9b244dc217d681e2625')
  }
}
```

3. **Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.**

db.Customers.aggregate([{$match: { Acc_Type: "Z" } },
{$group: {_id:"$Cust_id",TotalBalance: { $sum: "$Acc_Bal"}}},
{$match:{TotalBalance:{$gt:1200 } } }])

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.Customers.aggregate([
...   { $match: { Acc_Type: "Z" } },
...   {
...     $group: {
...       _id: "$Cust_id",
...       TotalBalance: { $sum: "$Acc_Bal" }
...     }
...   },
...   { $match: { TotalBalance: { $gt: 1200 } } }
... ])
[ { _id: 103, TotalBalance: 1500 }, { _id: 101, TotalBalance: 1300 } ]
```

4. **Determine Minimum and Maximum account balance for each customer_id.**

db.Customers.aggregate([{ $group: { _id: "$Cust_id",MinBalance: { $min: "$Acc_Bal"},
MaxBalance:{ $max: "$Acc_Bal" }}}])

```
[
  { _id: 103, MinBalance: 1500, MaxBalance: 1500 },
  { _id: 101, MinBalance: 500, MaxBalance: 800 },
  { _id: 102, MinBalance: 300, MaxBalance: 700 }
]
```

5. **Drop the table**

db.Customers.drop()

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.Customers.drop()
true
```

NoSQL Restaurant Database

Question
(Week 9)

Perform the following DB operations using MongoDB.

1.Write NoSQL Queries on “Restaurant” collection.

2.Write a MongoDB query to display all the documents in the collection restaurants.

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

4.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.

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

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

Queries

1. **Write NoSQL Queries on “Restaurant” collection.**

db.createCollection("restaurants");

```
Atlas atlas-r8p6bd-shard-0 [secondary] test> db.createCollection("restaurants");
{ ok: 1 }
```

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"}}])

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693baae8ef7941ced11e2621'),
    '1': ObjectId('693baae8ef7941ced11e2622'),
    '2': ObjectId('693baae8ef7941ced11e2623'),
    '3': ObjectId('693baae8ef7941ced11e2624'),
    '4': ObjectId('693baae8ef7941ced11e2625')
  }
}
```

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

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.restaurants.find({})
[
  {
    _id: ObjectId('693baae8ef7941ced11e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

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

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

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.restaurants.find({}).sort({ name: -1 });
[
  {
    _id: ObjectId('693baae8ef7941ced11e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

4. **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.**

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

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cucuisine: 1 });
[
  {
    _id: ObjectId('693baae8ef7941ced11e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId('693baae8ef7941ced11e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
]
```

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

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

```
Atlas atlas-r8p6bd-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }]);
[
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Empire', average_score: 7 }
]
```

6. **6.Write a MongoDB 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-r8p6bd-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' } }
]
```