

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT on

Database Management Systems (23CS3PCDBM)

Submitted by

Sahil Biswas (1BF24CS262)

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-2025 to Jan-2026

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 **Sahil Biswas (1BF24CS262)**, who is bonafide student of **B. M. S. College of Engineering**. It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2025. 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.

Kanchana Dixit Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE
---	--

Index

Sl. No .	Date	Experiment Title	Page No.
1	25-9-2025	Insurance Database	4 - 10
3	9-10-2025	Bank Database	11 - 17
4	16-10-202 5	More Queries on Bank Database	18- 21
5	30-10-202 5	More Queries on Insurance Database	22-23
6	6-11-2025	Employee Database	22-30
7	13-11-202 5	More Queries on Employee Database	31-32
5	20-11-202 5	Supplier Database	33-37
6	27-11-202 5	More Queries on Supplier Database	38-40
7	11-12-202 5	NO SQL- Student Database	41-42
8	11-12-202 5	NO SQL- Customer Database	43-44

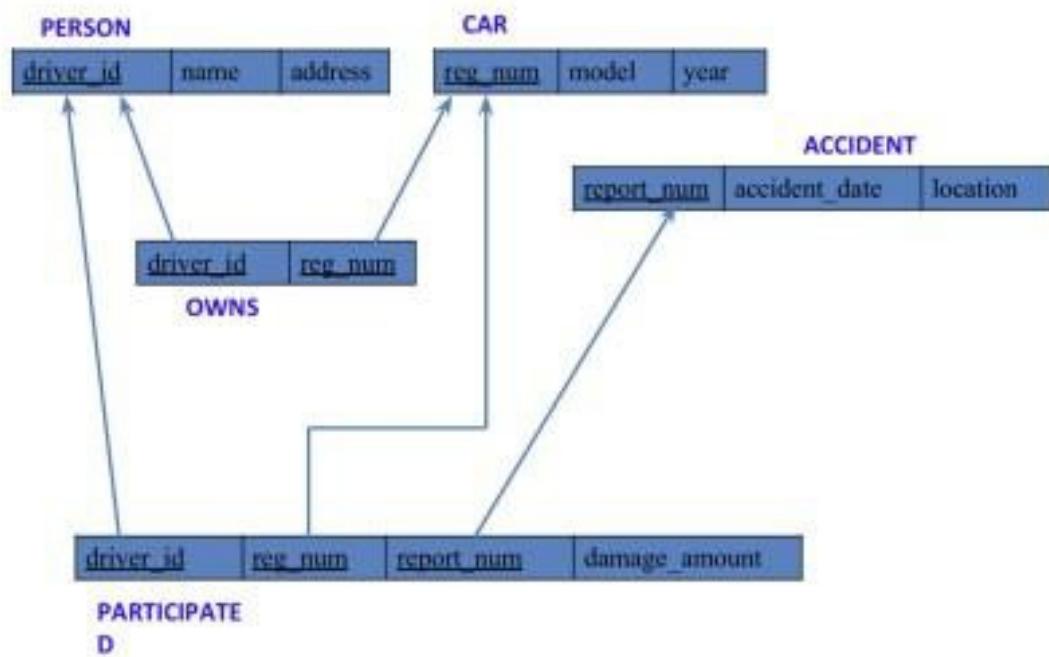
Insurance Database

Question

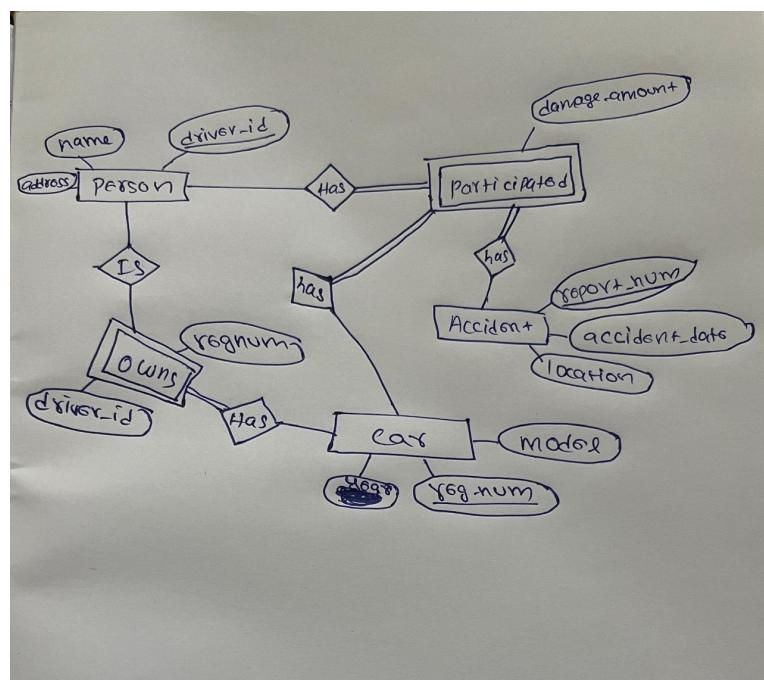
(Week 1)

- 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



ER diagram



Create database

```
create database Insurance_Database;  
use Insurance_Database;
```

Create table

```
create table Person (driver_id varchar(10), name varchar(20), address varchar(30), primary key(driver_id));  
create table Car (reg_num varchar(10), model varchar(10), year int, primary key(reg_num));  
create table Accident (report_num int, accident_date date, location varchar(20), primary key(report_num));  
create table Owns (driver_id varchar(10), reg_num varchar(10), primary key(driver_id, reg_num),  
foreign key(driver_id) references Person(driver_id), foreign key(reg_num) references Car(reg_num));  
create table Participated (driver_id varchar(10), reg_num varchar(10), report_num int, damage_amount int,  
primary key(driver_id, reg_num, report_num), foreign key(driver_id) references Person(driver_id),  
foreign key(reg_num) references Car(reg_num), foreign key(report_num) references Accident(report_num));
```

Structure of the table

```
desc Person;
```

	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(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', 'Monica', 'NR Colony'),  
('A02', 'Richard', 'Srinivas Nagar'), ('A03', 'Smith', 'Ashok  
Nagar'), ('A04', 'John', 'Hanumanth Nagar'), ('A05', 'Venu',  
'Rajaji Marg');
```

```
select * from Person;
```

	driver_id	name	address
▶	A01	Monica	NR Colony
◀	A02	Richard	Srinivas Nagar
◀	A03	Smith	Ashok Nagar
◀	A04	John	Hanumanth Nagar
◀	A05	Venu	Rajaji Marg
	NULL	NULL	NULL

```
insert into Car values ('KA052250', 'Indica', 1990),  
('KA031181', 'Lancer', 1957), ('KA095477', 'Toyota',  
1998), ('KA053408', 'Honda', 2008), ('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'), ('A02', 'KA031181'), ('A03', 'KA095477'), ('A04', 'KA053408'),
('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'), (12,
'2004-02-01', 'Southend Circle'), (13, '2003-01-21', 'Bull Temple
Road'), (14, '2008-02-17', 'Mysore Road'), (15, '2005-03-04',
'Kanapura Road');

```

```
select * from Accident;
```

report_num	accident_date	location
11	2003-01-01	Mysore Road
12	2004-02-01	Southend Circle
13	2003-01-21	Bull Temple Road
14	2008-02-17	Mysore Road
15	2005-03-04	Kanapura Road
16	2008-03-08	Dolmur
NULL	NULL	NULL

```

insert into Participated values ('A01', 'KA052250', 11, 10000),
('A02', 'KA031181', 12, 50000), ('A03', 'KA095477', 13, 25000),
('A04', 'KA053408', 14, 8000), ('A05', 'KA041702', 15, 5000);

```

```
select * from Participated;
```

driver_id	reg_num	report_num	damage_amount
A02	KA031181	12	50000
A03	KA095477	13	25000
A04	KA053408	14	25000
NULL	NULL	NULL	NULL

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

- 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
	13	2003-01-21	Bull temple road
	14	2008-02-17	Mysore road
	15	2004-03-15	kanakapura road
	16	2008-03-08	Domlur
*	NULL	NULL	NULL

TO DO

- **Display Accident date and location.**

```
select accident_date,location  
from Accident;
```

	accident_date	location
▶	2003-01-01	Mysore road
	2004-02-02	south end
	2003-01-21	Bull temple road
	2008-02-17	Mysore road
	2004-03-15	kanakapura road

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

```
SELECT driver_id  
FROM Participated  
WHERE damage_amount >= 25000;
```

	driver_id
▶	A02
	A03

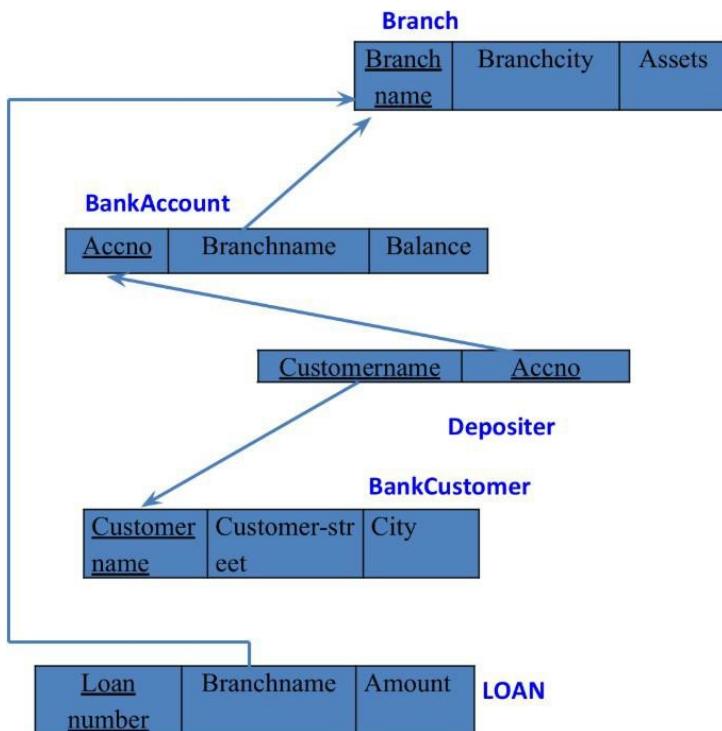
Bank Database

Question

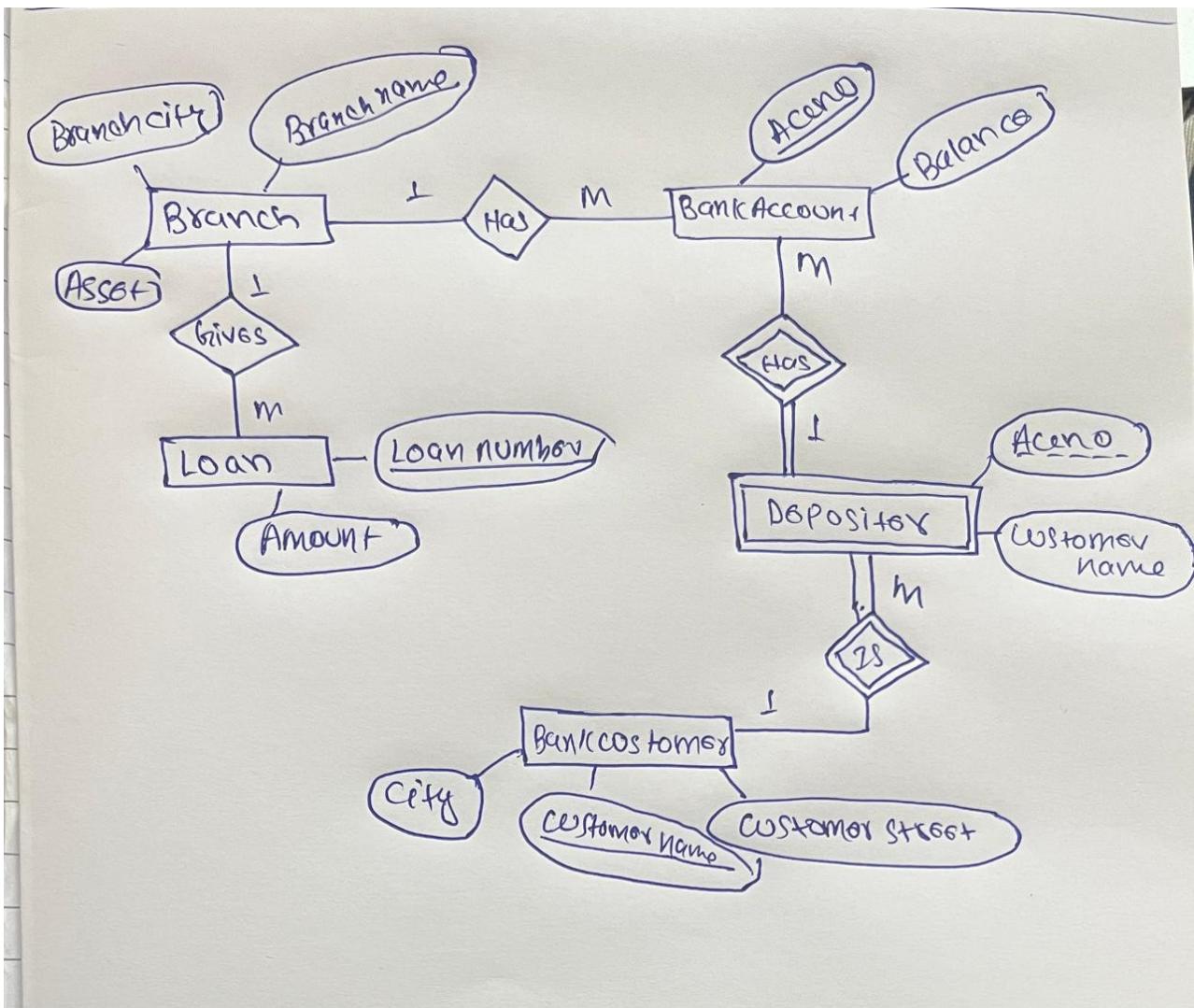
(Week 2)

- 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



ER diagram



Create database

```
create database  
Bank_Database;  
  
use Bank_Database;
```

Create table

```
create table Branch (branchname varchar(25), branchcity varchar(15), assets real, primary key(branchname));  
  
create table BankAccount(accno integer, branchname varchar(25), balance real,  
primary key(accno), foreign key(branchname) references Branch(branchname));  
  
create table BankCustomer(customername varchar(10), customerstreet varchar(25),  
customercity varchar(15), primary key(customername));  
  
create table Depositer(customername varchar(10), accno integer,  
primary key(customername, accno), foreign key(customername) references  
BankCustomer(customername), foreign key(accno) references BankAccount(accno));  
  
create table Loan(loannumber int, branchname varchar(25), amount real,  
primary key (loannumber), foreign key(branchname) references Branch(branchname));
```

Structure of the table

```
desc Branch;
```

	Field	Type	Null	Key	Default	Extra
▶	branch_name	varchar(25)	NO	PRI	NULL	
	branch_city	varchar(15)	YES		NULL	
	assets	int	YES		NULL	

```
desc BankAccount;
```

	Field	Type	Null	Key	Default	Extra
▶	accno	int	NO	PRI	NULL	
	branch_name	varchar(25)	YES	MUL	NULL	
	balance	int	YES		NULL	

```
desc BankCustomer;
```

	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(10)	NO	PRI	NULL	
	customer_street	varchar(25)	YES		NULL	
	customer_city	varchar(15)	YES		NULL	

desc Depositer;

	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(10)	YES	MUL	NULL	
	accno	int	YES	MUL	NULL	

desc Loan;

	Field	Type	Null	Key	Default	Extra
▶	loan_number	int	NO	PRI	NULL	
	branch_name	varchar(25)	YES	MUL	NULL	
	amount	int	YES		NULL	

Inserting Values to the table

```
insert into Branch values('SBI_chamrajpet','Bangalore',5000); insert  
into Branch values('SBI_residencyRoad','Bangalore',10000);  
insert into Branch values('SBI_shivajiRoad','Bombay',20000);  
insert into Branch values('SBI_parlimentRoad','delhi',10000);  
insert into Branch values('SBI_jantarmantar','delhi',20000);  
select * from Branch;
```

	branch_name	branch_city	assets
▶	SBI_chamrajpet	Bangalore	5000
	SBI_jantarmantar	delhi	20000
	SBI_parlimentRoad	delhi	10000
	SBI_residencyRoad	Bangalore	10000
	SBI_shivajiRoad	Bombay	20000
*	NULL	NULL	NULL

```
insert into BankAccount values(1,'SBI_chamrajpet',2000);
```

```
insert into BankAccount values(2,'SBI_residencyRoad',5000);  
insert into BankAccount values(3,'SBI_shivajiRoad',6000);
```

```
insert into BankAccount values(4,'SBI_parlimentRoad',9000);  
insert into BankAccount values(5,'SBI_jantarmantar',8000);
```

```
insert into BankAccount values(6,'SBI_shivajiRoad',4000);
```

```
insert into BankAccount values(8,'SBI_residencyRoad',4000);  
insert into BankAccount values(9,'SBI_parlimentRoad',3000);
```

```
insert into BankAccount values(10,'SBI_residencyRoad',5000);
```

```
insert into BankAccount values(11,'SBI_jantarmantar',2000);
```

```
select * from BankAccount;
```

	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 BankCustomer values('avinash','BullTempleRoad','Bangalore');  
insert into BankCustomer values('dinesh','BannerGattaRoad','Bangalore');  
insert into BankCustomer values('mohan','nationalCollegeRoad','Bangalore');  
insert into BankCustomer values('nikil','AkbarRoad','Delhi');  
insert into BankCustomer values('ravi','pritvirajRoad','Delhi');  
select * from BankCustomer;
```

	customer_name	customer_street	customer_city
▶	avinash	BullTempleRoad	Bangalore
	dinesh	BannerGattaRoad	Bangalore
	mohan	nationalCollegeRoad	Bangalore
	nikil	AkbarRoad	Delhi
	ravi	pritvirajRoad	Delhi
*	NULL	NULL	NULL

```
insert into Depositer values('avinash',1);  
insert into Depositer values('dinesh',2);  
insert into Depositer values('nikil',4);  
insert into Depositer values('ravi',5);  
insert into Depositer values('avinash',8);  
insert into Depositer values('nikil',9);
```

```
insert into Depositer values('dinesh',10);
insert into Depositer values('nikil',11);
select * from Depositer;
```

	customer_name	accno
▶	avinash	1
	dinesh	2
	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

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

```
alter table Branch
```

```
rename column assets to assets_inlakhs;
```

```
select branch_name,assets_inlakhs from Branch;
```

	branch_name	assets_inlakhs
▶	SBI_chamrajpet	5000
	SBI_jantarmantar	20000
	SBI_parlimentRoad	10000
	SBI_residencyRoad	10000
	SBI_shivajiRoad	20000
*	NULL	NULL

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

```
select d.customer_name  
from Depositer d,BankAccount 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

- CREATE A VIEW WHICH GIVES EACH BRANCH THE SUM OF THE AMOUNT OF ALL THE LOANS AT THE BRANCH.

```
create view br  
as  
select branch_name,sum(amount)  
from Loan  
group by branch_name;  
select * from br;
```

	branch_name	sum(amount)
▶	SBI_chamrajpet	1000
	SBI_jantarmantar	5000
	SBI_parlimentRoad	4000
	SBI_residencyRoad	2000
	SBI_shivajiRoad	3000

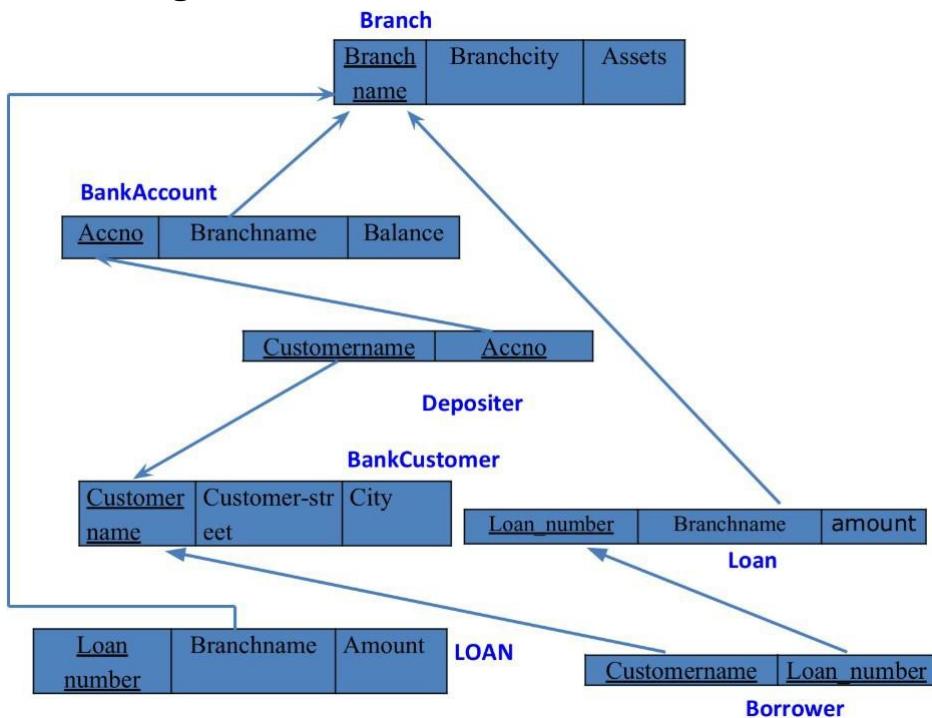
More Queries on Bank Database

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

Schema Diagram



Create table

```
create table Borrower (
    customer_name varchar(10),
    loan_number int,
    foreign key(customer_name) references BankCustomer(customer_name),
    foreign key(loan_number) references Loan (loan_number)
);
```

Structure of table

```
desc Borrower;
```

	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(10)	YES	MUL	NULL	
	loan_number	int	YES	MUL	NULL	

Insert values to the table

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

Queries

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

```
select 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.branch_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_number not in(select d.accno from Depositer d where
b.loan_number=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_number in (select d.accno from Depositer d, BankAccount ba, Branch b where b.loan_number=d.accno and d.accno=ba.accno and
ba.branch_name=b.branch_name and b.branch_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
branch_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
branch_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

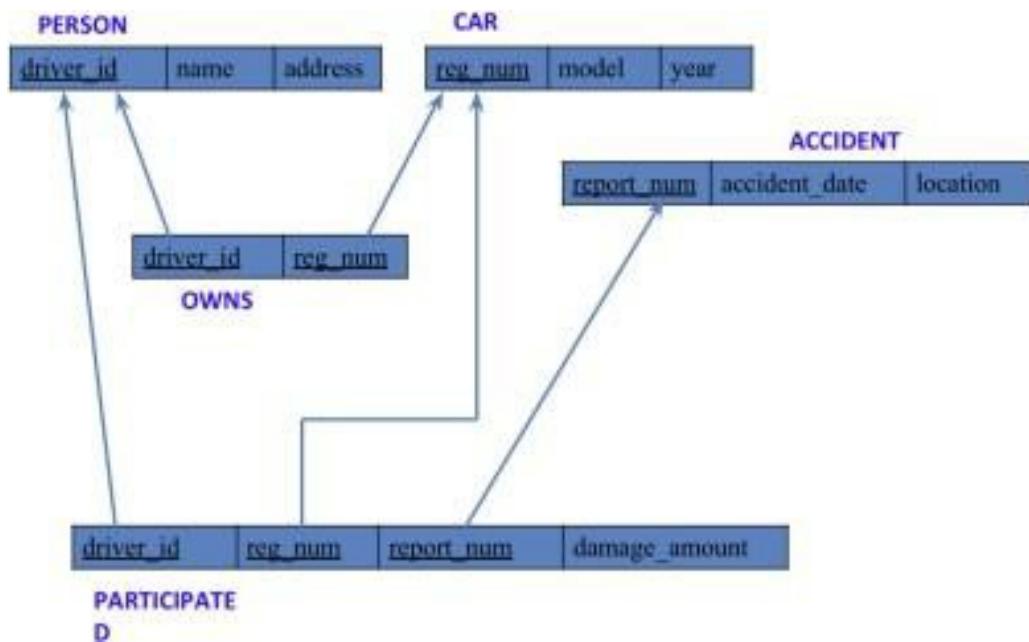
More Queries on Insurance Database

Question

(Week 4)

- 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)
- LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER OF DAMAGE AMOUNT.
- FIND THE AVERAGE DAMAGE AMOUNT.
- LIST THE NAME OF DRIVERS WHOSE DAMAGE IS GREATER THAN THE AVERAGE DAMAGE AMOUNT.
- FIND MAXIMUM DAMAGE AMOUNT.

Schema Diagram



Queries

- 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	KA031181	13	25000
	A01	KA052250	11	10000
	A05	KA041702	15	5000
	A04	KA095477	14	3000

- FIND THE AVERAGE DAMAGE AMOUNT.

```
SELECT AVG(damage_amount) 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

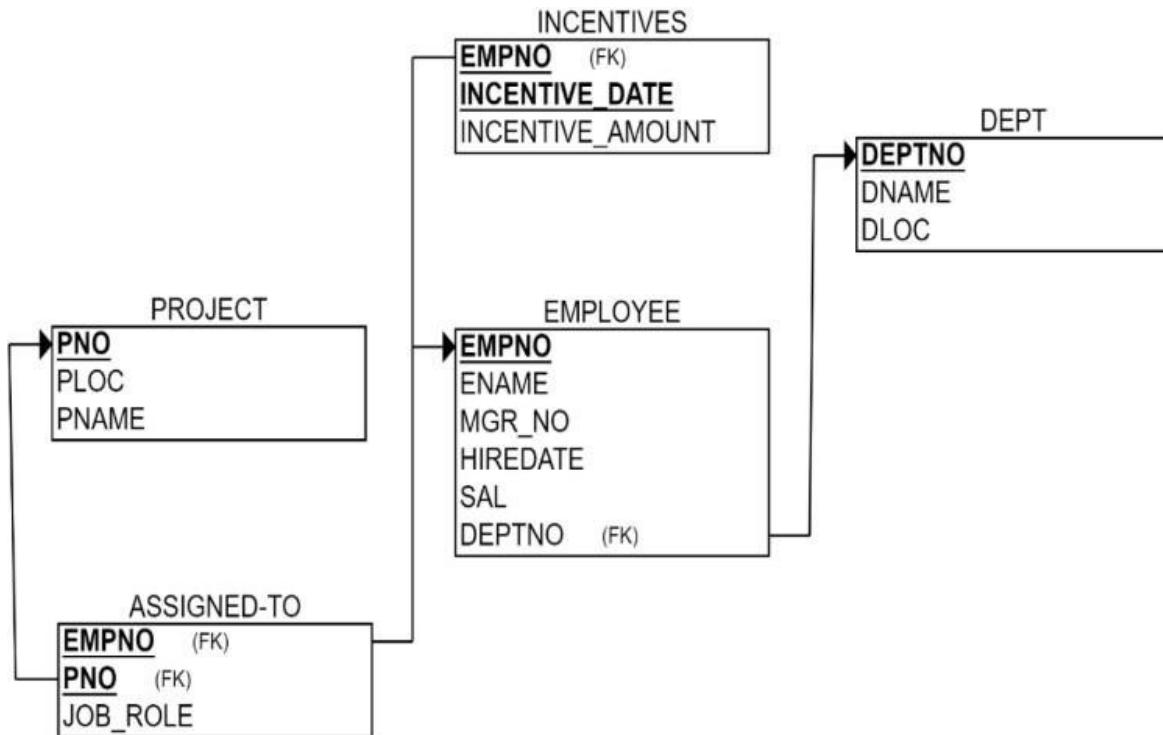
Employee Database

Question

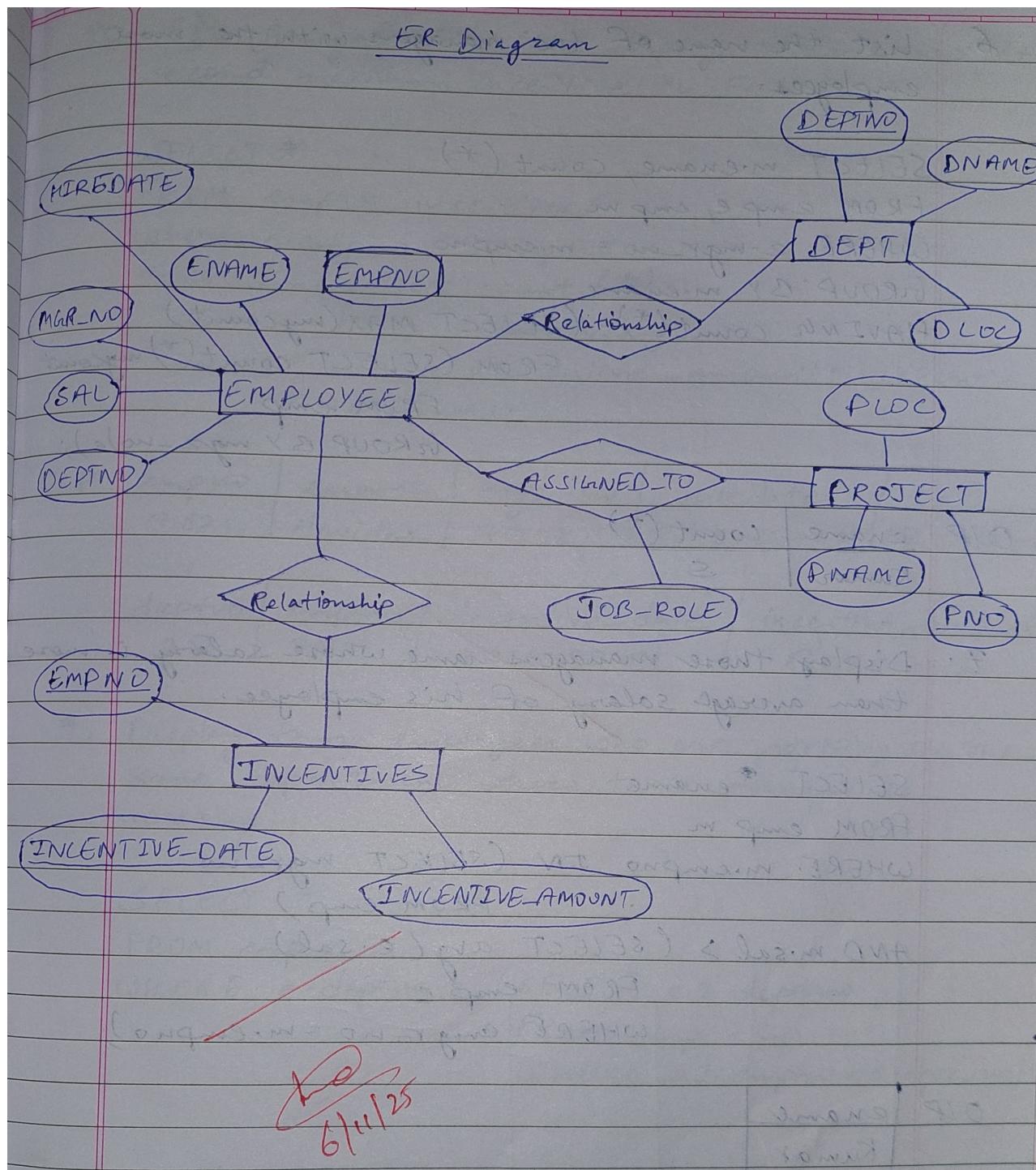
(Week 5)

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

Schema Diagram



ER diagram



Create database

```
create database EMP;
```

```
use EMP;
```

Create table

```
create table dept (
    deptno int,
    dname varchar(20),
    dloc varchar(20),
    primary key(deptno)
);
create table emp (
    empno int,
    ename varchar(20),
    mgr_no int,
    hiredate varchar(20),
    sal int,
    deptno int,
    primary key(empno,deptno),
    foreign key(deptno) references dept_202(deptno)
);
create table incentives (
    empno int,
    incentives_date varchar(20),
    amount int,
    primary key(empno,incentives_date),
    foreign key(empno) references employee_202(empno)
);
create table project (
    pno int,
    ploc varchar(20),
    pname varchar(20),
    primary key(pno)
);
create table assigned_to (
    empno int,
    pno int,
    job_role varchar(20),
    primary key(empno,pno),
    foreign key(empno) references employee_202(empno),
    foreign key(pno) references project_202(pno));
```

Structure of the table

desc dept;

	Field	Type	Null	Key	Default	Extra
▶	deptno	int	NO	PRI	NULL	
	dname	varchar(20)	YES		NULL	
	dloc	varchar(20)	YES		NULL	

desc emp;

	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	ename	varchar(20)	YES		NULL	
	mgr_no	int	YES		NULL	
	hiredate	varchar(20)	YES		NULL	
	sal	int	YES		NULL	
	deptno	int	NO	PRI	NULL	

desc incentives;

	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	incentives_date	varchar(20)	NO	PRI	NULL	
	amount	int	YES		NULL	

desc project;

	Field	Type	Null	Key	Default	Extra
▶	pno	int	NO	PRI	NULL	
	ploc	varchar(20)	YES		NULL	
	pname	varchar(20)	YES		NULL	

desc assigned_to;

	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	pno	int	NO	PRI	NULL	
	job_role	varchar(20)	YES		NULL	

Insert values to the tables

```
insert into dept values(1,'HR','new_delhi');
insert into dept values(2,'IT','bangalore');
insert into dept values(3,'Finance','mysuru');
insert into dept values(4,'development','hyderabad');
insert into dept values(5,'marketing','new_delhi');
select * from dept;
```

	deptno	dname	dloc
▶	1	HR	new_delhi
	2	IT	bangalore
	3	Finance	mysuru
	4	development	hyderabad
	5	marketing	new_delhi
*	NULL	NULL	NULL

```
insert into emp values(101,'raj',100,"12/01/1999",100000,1);
insert into emp values(201,'adhi',200,"17/01/2020",50000,2);
insert into emp values(301,'priyam',100,"01/09/2004",30000,3);
insert into emp values(401,'asha',101,"03/08/2000",10000,4);
insert into emp values(501,'shailesh',101,"29/2/2008",90000,5);
insert into emp values(601,'likith',102,"29/2/2008",90000,1);
```

```
select * from emp;
```

	empno	ename	mgr_no	hiredate	sal	deptno
▶	101	raj	100	12/01/1999	100000	1
	201	adhi	200	17/01/2020	50000	2
	301	priyam	100	01/09/2004	30000	3
	401	asha	101	03/08/2000	10000	4
	501	shailesh	101	29/2/2008	90000	5
	601	likith	102	29/2/2008	90000	1
*	NULL	NULL	NULL	NULL	NULL	NULL

```

insert into incentives values(101,"12/03/2004",50000);
insert into incentives values(201,"17/03/2024",25000);
insert into incentives values(301,"01/12/2019",15000);
insert into incentives values(401,"03/11/2019",5000);
insert into incentives values(501,"29/4/2019",45000);
select * from incentives;

```

	empno	incentives_date	amount
▶	101	12/03/2004	50000
	201	17/03/2024	25000
	301	01/12/2019	15000
	401	03/11/2019	5000
*	501	29/4/2019	45000
	NULL	NULL	NULL

```

insert into project values(10,"new_delhi","chatbot");
insert into project values(40,"bangalore","ml model");
insert into project values(50,"mysuru","blockchain");
insert into project values(30,"hyderabad","stocks");
insert into project values(80,"new_delhi","android app");
select * from project;

```

	pno	ploc	pname
▶	10	new_delhi	chatbot
	30	hyderabad	stocks
	40	bangalore	ml model
	50	mysuru	blockchain
	80	new_delhi	android app
*	NULL	NULL	NULL

```

insert into assigned_to values(101,10,"devops");
insert into assigned_to values(201,40,"sde");

```

```

insert into assigned_to values(301,50,"manager");
insert into assigned_to values(401,30,"jpa");

```

```

insert into assigned_to values(501,80,"pa");

```

```

select * from assigned_to;

```

	empno	pno	job_role
▶	101	10	devops
	201	40	sde
	301	50	manager
	401	30	jpa
	501	80	pa
*	NULL	NULL	NULL

Queries

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

select a.empno from assigned_to a,project p

where a.pno=p.pno and (ploc='bangalore' or ploc='hyderabad' or ploc='mysuru');

	empno
▶	401
	201
	301

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

select e.empno from emp e

where e.empno!=all(select i.empno from incentives i);

	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 e.empno, e.ename, d.dname, a.job_role, d.dloc, p.ploc

from emp e, dept d, assigned_to a, project p

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

	empno	ename	dname	job_role	dloc	ploc
▶	101	raj	HR	devops	new_delhi	new_delhi
	201	adhi	IT	sde	bangalore	bangalore
	301	priyam	Finance	manager	mysuru	mysuru
	401	asha	development	jpa	hyderabad	hyderabad
	501	shailesh	marketing	pa	new_delhi	new_delhi

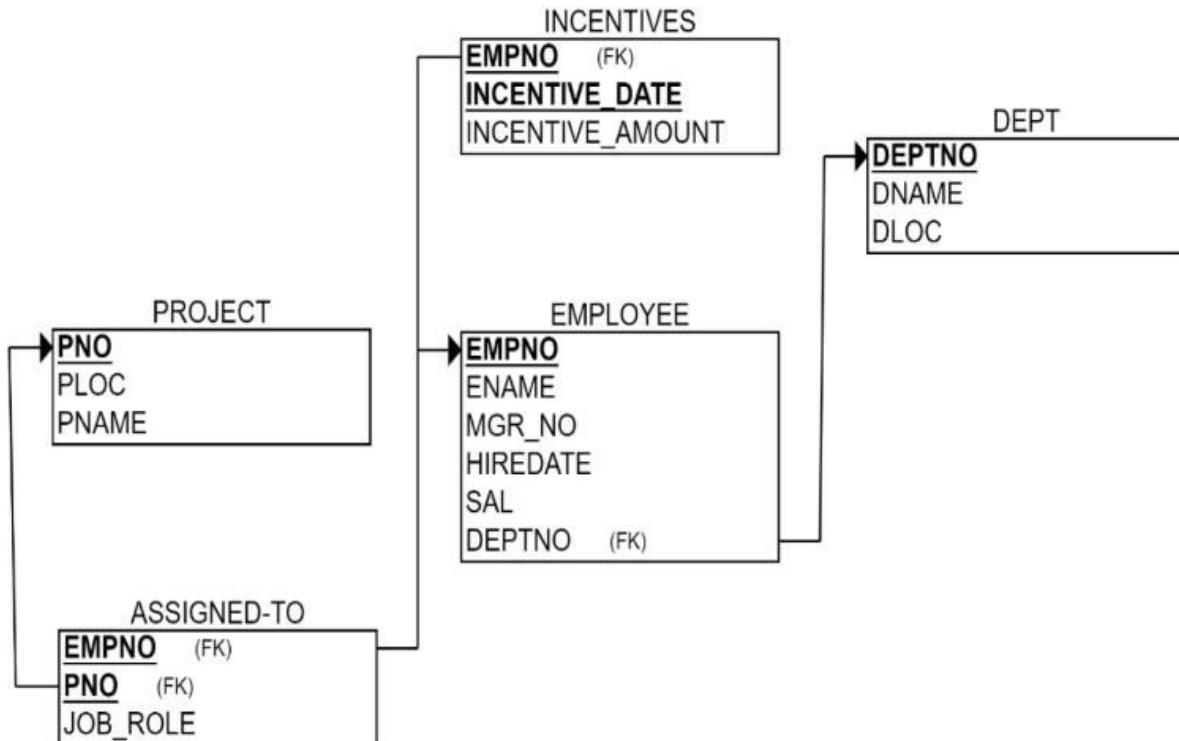
More Queries on Employee Database

Question

(Week 6)

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

Schema Diagram



Queries

- List the name of the managers with the maximum employees

```
select e.ename from emp e where e.empno in (select m.mgr_no from emp m group by m.mgr_no having count(*) = ( select max(emp_count) from ( select count(*) as emp_count from emp em group by em.mgr_no ) as emp_count_subquery));
```

	ename
▶	raj

- Display those managers' names of the manager whose salary is more than the average salary of his employee.

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

	ename
▶	raj

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

select ename from emp where sal=(select max(sal) from emp where sal < (select max(sal) from emp));

	ename
▶	shailesh
	likith
	likith

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

select * from emp 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	shailesh	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 emp e ,emp m where e.mgr_no=m.empno and e.deptno=m.deptno;

	ename
▶	likith

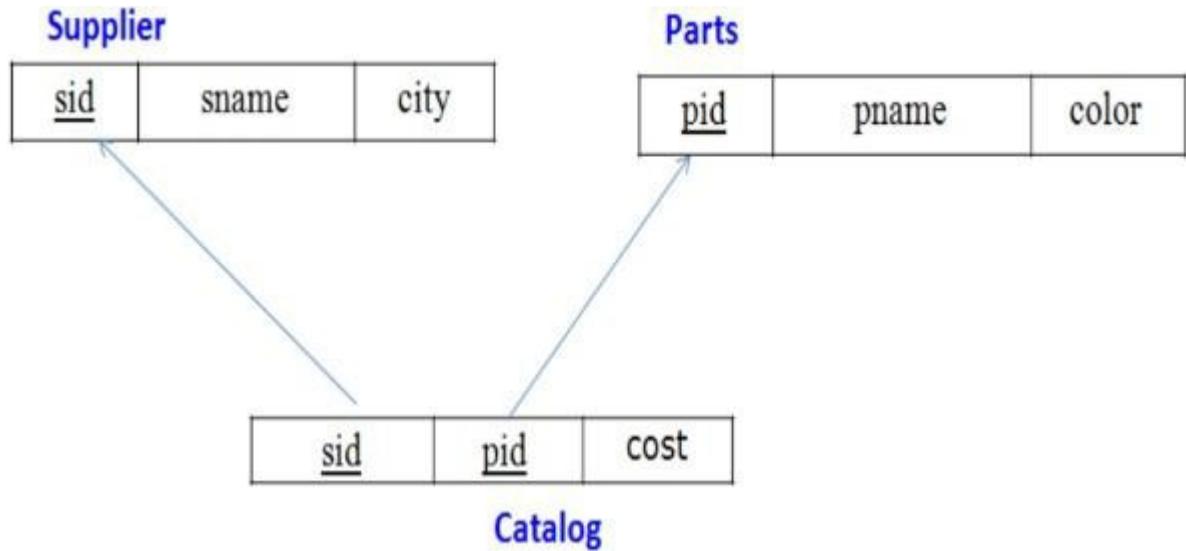
Supplier Database

Question

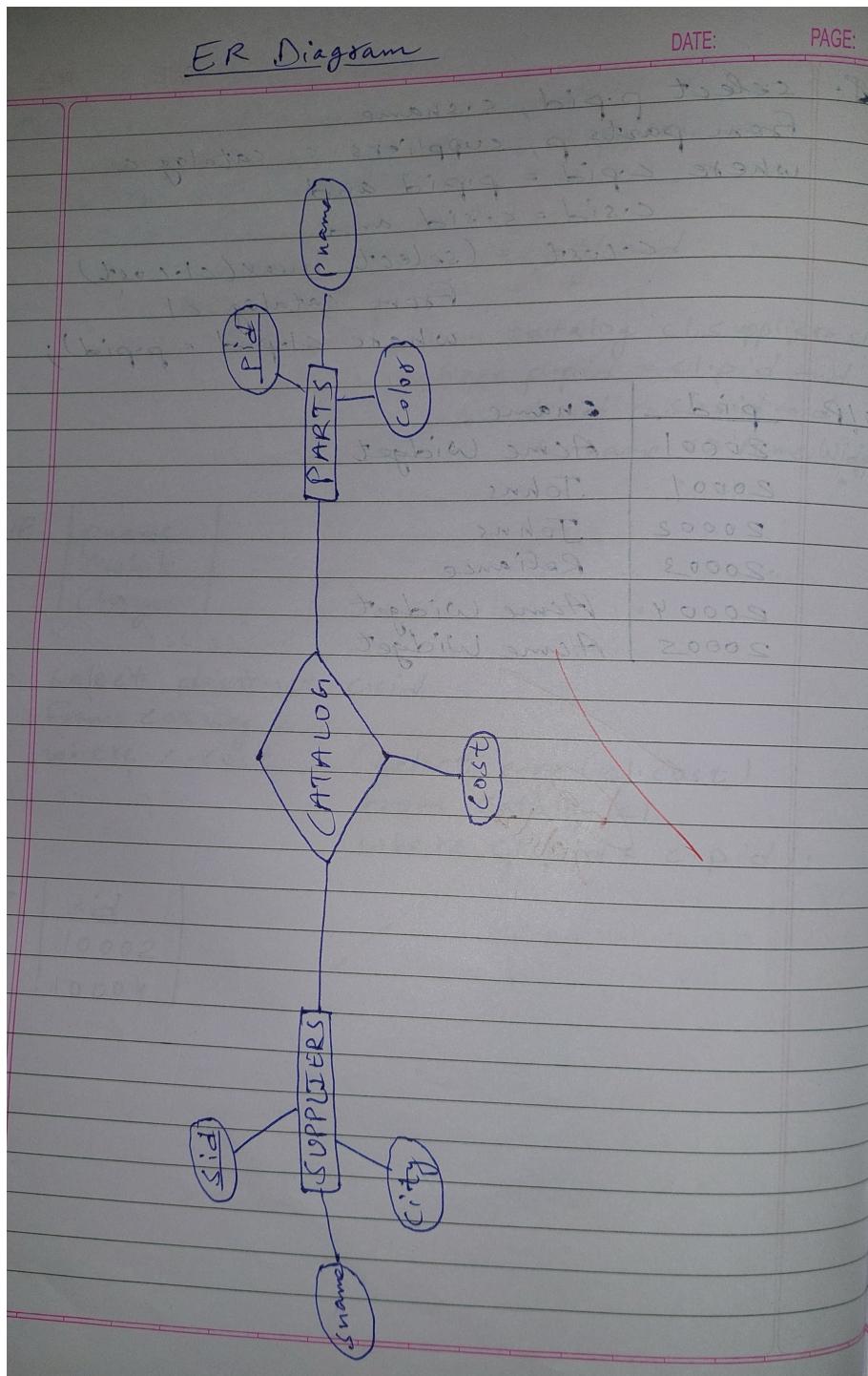
(Week 7)

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

Schema Diagram



ER diagram



Create database

```
create database supplier;
```

```
use supplier;
```

Create table

```
create table  
suppliers ( 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_202(sid),  
foreign key(pid) references parts_202(pid));
```

Structure of the table

```
desc suppliers;
```

	Field	Type	Null	Key	Default	Extra
▶	sid	int	NO	PRI	NULL	
	sname	varchar(20)	YES		NULL	
	city	varchar(20)	YES		NULL	

```
desc parts;
```

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

desc catalog;

	Field	Type	Null	Key	Default	Extra
▶	sid	int	YES	MUL	NULL	
	pid	int	YES	MUL	NULL	
	cost	int	YES		NULL	

Insert values to the tables

```
insert into suppliers values (10001, 'Acme Widget','Bangalore');
insert into suppliers values (10002, 'Johns','Kolkata');
insert into suppliers values (10003, 'Vimal','Mumbai');
insert into suppliers values (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');
insert into parts values (20002, 'Pen','Red');
insert into parts values (20003, 'Pencil','Green');
insert into parts values (20004, 'Mobile','Green');
insert into parts values (20005, 'Charger','Black');
select * from parts;
```

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

```
insert into catalog values (10001, 20001 , 10);
insert into catalog values (10001, 20002 , 10);
insert into catalog values (10001, 20003 , 30);
insert into catalog values (10001, 20004 , 10);
insert into catalog values (10001, 20005 , 10);
insert into catalog values (10002, 20001 , 10);
```

```

insert into catalog values (10002, 20002 , 20);
insert into catalog values (10003, 20003 , 30);
insert into catalog values (10004, 20003 , 40);
select * from catalog;

```

	sid	pid	cost
▶	10001	20001	10
	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 distinct s.sname from suppliers 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 p.pid=c.pid));
```

	sname
▶	Acme Widget

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

```
select s.sname from suppliers 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 p.pid=c.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_202 c,suppliers s
 where c.sid=s.sid and p.pid=c.pid and s.sname='Acme Widget' and not exists(
 select * from catalog c1,suppliers s1 where c1.sid=s1.sid and p.pid=c1.pid 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, catalog c, suppliers s
 where c.sid=s.sid and p.pid=c.pid 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

MORE QUERIES ON SUPPLIER DATABASE

- Find the most expensive part overall and the supplier who supplies it.

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

	Pname	Sname	Cost
▶	Pencil	Reliance	40

- Find suppliers who do NOT supply any red parts.

Select s.sname from suppliers s where s.sid not in (select c.sid from catalog c join parts p on c.pid=p.pid where p.color='Red');

	Sname
▶	Vimal
	Reliance

- Show each supplier and total value of all parts they supply.

select s.sname, sum(c.cost) as total_value from suppliers s join catalog c on s.sid=c.sid group by s.sname

	Sname	Total_Value
▶	Acme Widget	70
	Johns	30
	Vimal	30
	Reliance	40

- Find suppliers who supply at least 2 parts cheaper than ₹20.

select s.sname from suppliers s join catalog c on s.sid=c.sid where c.cost<20 group by s.sname having count(c.pid)>=2;

	Sname
▶	Acme Widget

- List suppliers who offer the cheapest cost for each part.

select p.pname, s.sname, c.cost from parts p join catalog c on p.pid = c.pid join suppliers s on c.sid = s.sid where c.cost = (select min(c1.cost) from catalog c1 where c1.pid = p.pid);

	Pname	Sname	Cost
▶	Book	Acme Widget	10
	Pen	Acme Widget	10
	Pencil	Acme Widget	30
	Mobile	Acme Widget	10
	Charger	Acme Widget	10
	Book	Johns	10
	Pencil	Vimal	30

- **Create a view showing suppliers and the total number of parts they supply.**

```
create view supplierpartcount_view as select s.sname, count(c.pid) as part_count from suppliers s
join catalog c on s.sid = c.sid group by s.sname;
```

```
select * from SupplierPartCount_View;
```

	Sname	Part_Count
▶	Acme Widget	5
	Johns	2
	Vimal	1
	Reliance	1

- **Create a view of the most expensive supplier for each part.**

```
create view mostexpensivesupplier_view as select p.pname, s.sname, c.cost from parts p join catalog
c on p.pid = c.pid join suppliers s on c.sid = s.sid where c.cost = ( select max(c1.cost) from catalog
c1 where c1.pid = p.pid );
```

```
select * from MostExpensiveSupplier_View;
```

	Pname	Sname	Cost
▶	Book	Acme ...	10
	Mobile	Acme ...	10
	Charger	Acme ...	10
	Book	Johns	10
	Pen	Johns	20
	Pencil	Reliance	40

- **Create a Trigger to prevent inserting a Catalog cost below 1.**

```
delimiter //
```

```
create trigger min_cost_check_new
before insert on catalog
for each row
```

```

begin
    if new.cost < 1 then
        signal sqlstate '45000' set message_text = 'catalog cost cannot be less than 1.';
    end if;
end//  

delimiter ;

```

insert into catalog values (101, 201, 0);

Error Code: 1644. Catalog cost cannot be less than 1.

- Create a trigger to set to default cost if not provided.

```

delimiter //  

create trigger set_default_cost_new
before insert on catalog
for each row
begin
    if new.cost is null then
        set new.cost = 10;
    end if;
end//  

delimiter ;

```

insert into catalog values (10002, 20005, null);
select * from catalog where sid=10002 and pid= 20005;

	sid	pid	cost
▶	10002	20005	10
*	NULL	NULL	NULL

NO SQL- STUDENT DATABASE

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

```
Create database db.createCollection("Student");
```

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

```
Atlas atlas-117olx-shard-0 [primary] test> db.createCollection("Student")  
{ ok: 1 }  
Atlas atlas-117olx-shard-0 [primary] test> db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});  
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.  
{  
  acknowledged: true,  
  insertedIds: { '0': ObjectId('693b8910af646a8d4f1e2621') }  
}  
Atlas atlas-117olx-shard-0 [primary] test> db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});  
...  
... db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});  
...  
... db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});  
...  
... db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});  
{  
  acknowledged: true,  
  insertedIds: { '0': ObjectId('693b8921af646a8d4f1e2625') }  
}
```

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

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

```
Atlas atlas-117olx-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  
}
```

```
db.Student.insert({RollNo:11,Age:22,Name:"ABC",Cont:2276,email:"rea.de9@gmail.com"});  
Atlas atlas-117olx-shard-0 [primary] test> db.Student.insert({RollNo:11,Age:22,Name:  
... "ABC",Cont:2276,email:"rea.de9@gmail.com"});  
{  
  acknowledged: true,  
  insertedIds: { '0': ObjectId('693b8a99af646a8d4f1e2626') }  
}
```

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

```
Atlas atlas-117olx-shard-0 [primary] test> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
db.Student.drop()
db.New_Student.find()
```

```
Atlas atlas-117olx-shard-0 [primary] test> db.New_Student.find()
[
  {
    _id: ObjectId('693b8921af646a8d4f1e2622'),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId('693b8921af646a8d4f1e2623'),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId('693b8921af646a8d4f1e2625'),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  },
  {
    _id: ObjectId('693b8a99af646a8d4f1e2626'),
    RollNo: 11,
    Age: 22,
    Name: 'FEM',
    Cont: 2276,
    email: 'rea.de9@gmail.com'
  },
  {
    _id: ObjectId('693b8910af646a8d4f1e2621'),
    RollNo: 1,
    Age: 21,
  }
]
```

```
}
```

```
Atlas atlas-117olx-shard-0 [primary] test> db.Student.drop();
true
```

NO SQL- Customer DATABASE

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

```
Cust_id, Acc_Bal, Acc_Type use myDB db.createCollection("Customers")
Atlas atlas-117olx-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: 1000, Acc_Type: "A" },
{ Cust_id: 102, Acc_Bal: 1500, Acc_Type: "Z" },
{ Cust_id: 102, Acc_Bal: 200, Acc_Type: "Z" },
{ Cust_id: 103, Acc_Bal: 3000, Acc_Type: "A" },
{ Cust_id: 101, Acc_Bal: 500, Acc_Type: "Z" },
{ Cust_id: 104, Acc_Bal: 1300, Acc_Type: "Z" }
```

```
])
```

```
Atlas atlas-117olx-shard-0 [primary] test> db.Customers.insertMany([
...   { Cust_id: 101, Acc_Bal: 1000, Acc_Type: "A" },
...   { Cust_id: 102, Acc_Bal: 1500, Acc_Type: "Z" },
...   { Cust_id: 102, Acc_Bal: 200, Acc_Type: "Z" },
...   { Cust_id: 103, Acc_Bal: 3000, Acc_Type: "A" },
...   { Cust_id: 101, Acc_Bal: 500, Acc_Type: "Z" },
...   { Cust_id: 104, Acc_Bal: 1300, Acc_Type: "Z" }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693b915caf646a8d4f1e2627'),
    '1': ObjectId('693b915caf646a8d4f1e2628'),
    '2': ObjectId('693b915caf646a8d4f1e2629'),
    '3': ObjectId('693b915caf646a8d4f1e262a'),
    '4': ObjectId('693b915caf646a8d4f1e262b'),
    '5': ObjectId('693b915caf646a8d4f1e262c')
  }
}
```

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-117olx-shard-0 [primary] test> db.Customers.aggregate([
...   // Step 1: Filter for Account Type 'Z'
...   { $match: { Acc_Type: "Z" } },
...
...   // Step 2: Group by Customer ID and sum the Account Balance
...   { $group: { _id: "$Cust_id", totalBalance: { $sum: "$Acc_Bal" } } },
...
...   // Step 3: Filter results where totalBalance is greater than 1200
...   { $match: { totalBalance: { $gt: 1200 } } }
... ])
[ { _id: 102, totalBalance: 1700 }, { _id: 104, 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" } } }])
```

```
Atlas atlas-117olx-shard-0 [primary] test> db.Customers.aggregate([
...   {
...     $group: {
...       _id: "$Cust_id",
...       minBalance: { $min: "$Acc_Bal" },
...       maxBalance: { $max: "$Acc_Bal" }
...     }
...   }
... ])
[
  { _id: 103, minBalance: 3000, maxBalance: 3000 },
  { _id: 102, minBalance: 200, maxBalance: 1500 },
  { _id: 101, minBalance: 500, maxBalance: 1000 },
  { _id: 104, minBalance: 1300, maxBalance: 1300 }
]
```

5. Drop the table

```
db.Customers.drop()
```

```
Atlas atlas-117olx-shard-0 [primary] test> db.Student.drop();
true
```

