

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

“JnanaSangama”, Belgaum -590014, Karnataka.



## **LAB REPORT on**

## **Database Management Systems (23CS3PCDBM)**

*Submitted by*

**MAHENDRAGOWDA 1BM24CS156**

*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 **MAHENDRA GOWDA (1BM24CS156)**, 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. 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. Rashmi H Professor & HOD Department of CSE, BMSCE
--	---

## Index

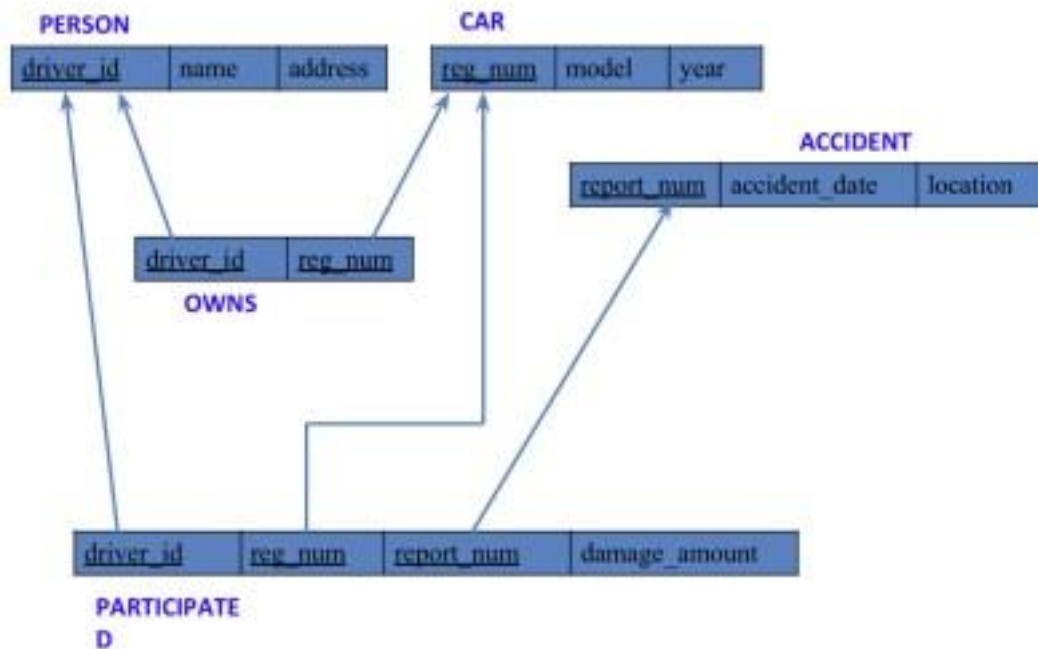
Sl. No.	Date	Experiment Title	Page No.
1	4-10-2024	Insurance Database	4-11
2	9-10-2024	More Queries on Insurance Database	12-15
3	16-10-2024	Bank Database	16-22
4	23-10-2024	More Queries on Bank Database	23-26
5	30-10-2024	Employee Database	27-33
6	13-11-2024	More Queries on Employee Database	34-37
7	20-11-2024	Supplier Database	38-46
8	27-11-2024	NO SQL - Student Database	47-49
9	4-12-2024	NO SQL - Customer Database	50-52
10	4-12-2024	NO SQL – Restaurant Database	53-57

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

## Schema Diagram



## Create database

```
create database insurance_dhiksha; use
insurance_dhiksha;
```

## Create table

```
create table
insurance_dhiksha.person( driver_id
varchar(20), name varchar(30),
address varchar(50),
PRIMARY KEY(driver_id)
);
create table insurance_dhiksha.car(
reg_num varchar(15),
model varchar(10),
year int,
PRIMARY KEY(reg_num)
);
create table insurance_dhiksha.owns(
```

```

driver_id varchar(20),
reg_num varchar(10),
PRIMARY KEY(driver_id, reg_num),
FOREIGN KEY(driver_id) REFERENCES person(driver_id),
FOREIGN KEY(reg_num) REFERENCES car(reg_num)
);

create table
insurance_dhiksha.accident(
report_num int, accident_date date,
location varchar(50),
PRIMARY KEY(report_num)
);

create table
insurance_dhiksha.participated( driver_id
varchar(20), reg_num varchar(10),
report_num int, damage_amount int,
PRIMARY KEY(driver_id,reg_num,report_num),
FOREIGN KEY(driver_id) REFERENCES person(driver_id),
FOREIGN KEY(reg_num) REFERENCES car(reg_num),
FOREIGN KEY(report_num) REFERENCES accident(report_num)
);

```

## Structure of the table

```
desc person;
```

Result Grid						
		Filter Rows:			Export:	Wrap Cell Content:
	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;

Result Grid						
		Filter Rows:			Export:	Wrap Cell Content:
	Field	Type	Null	Key	Default	Extra
▶	report_num	int	NO	PRI	NULL	
	accident_date	date	YES		NULL	
	location	varchar(20)	YES		NULL	

desc participated;

Result Grid						
		Filter Rows:			Export:	Wrap Cell Content:
	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;

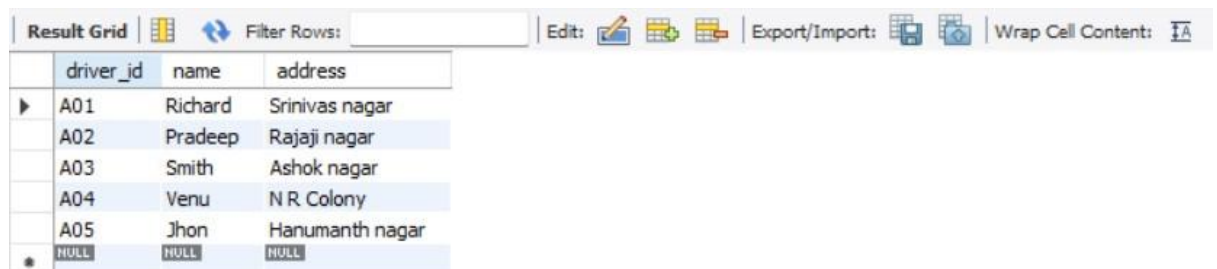
Result Grid						
		Filter Rows:			Export:	Wrap Cell Content:
	Field	Type	Null	Key	Default	Extra
▶	reg_num	varchar(10)	NO	PRI	NULL	
	model	varchar(10)	YES		NULL	
	year	int	YES		NULL	

desc owns;

Result Grid						
		Filter Rows:			Export:	Wrap Cell Content:
	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(10)	NO	PRI	NULL	
	reg_num	varchar(10)	NO	PRI	NULL	

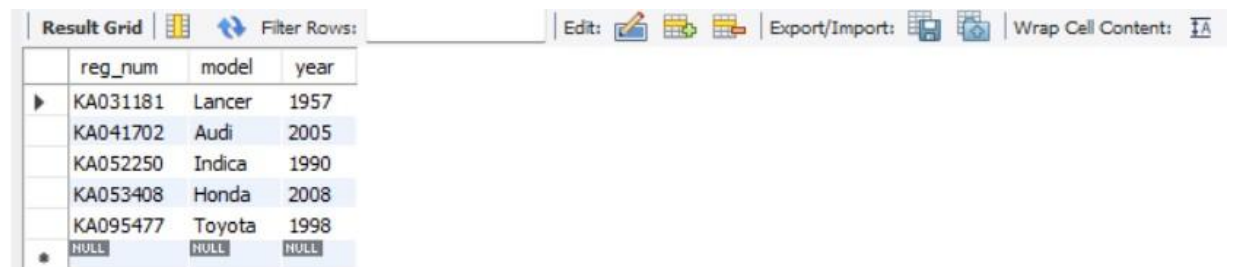
## Inserting Values to the table

```
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","Jhon", "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	Jhon	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;
```



Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	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,'2005-06-07',"Bull temple Road");

insert into accident values(14,'2006-09-01',"Kanakpura Road");

insert into accident values(15,'2007-10-11',"My Road"); select

\* from accident;

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	report_num	accident_date	location		
▶	11	2003-01-01	Mysore Road		
	12	2004-01-02	South End Cide		
	13	2005-06-07	Bull Temple Road		
	14	2006-09-01	Kanakpura Road		
	15	2007-10-11	My 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;

Result Grid		Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	driver_id	reg_num	report_num	damage_amount	
▶	A01	KA052250	11	10000	
	A02	KA053408	12	50000	
	A03	KA095477	13	25000	
	A04	KA031181	14	3000	
	A05	KA041702	15	5000	
✱	NULL	NULL	NULL	NULL	

## Queries

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

update participated set

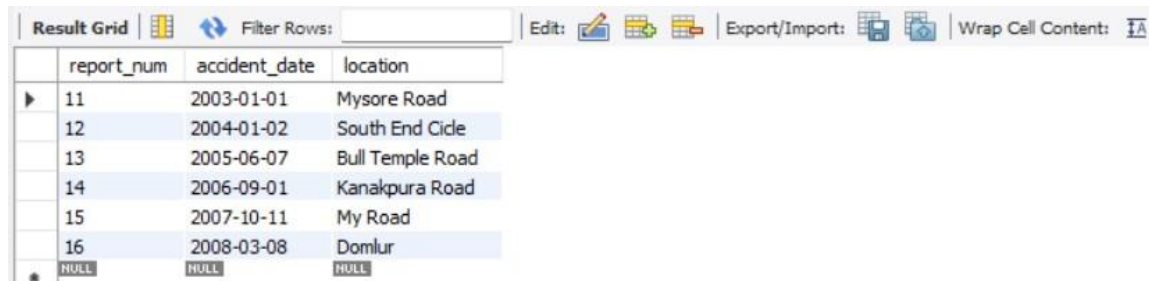
damage\_amount=25000

where reg\_num='KA053408' and report\_num=12;

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

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

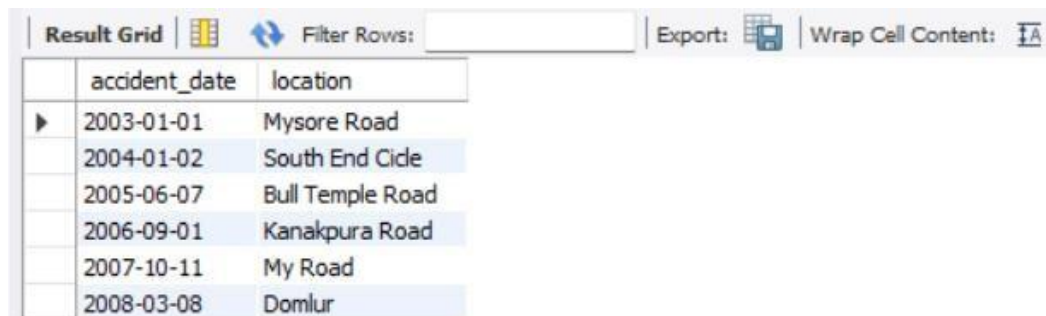
select \* from accident;



	report_num	accident_date	location
▶	11	2003-01-01	Mysore Road
	12	2004-01-02	South End Cicle
	13	2005-06-07	Bull Temple Road
	14	2006-09-01	Kanakpura Road
	15	2007-10-11	My Road
	16	2008-03-08	Domlur
•	NULL	NULL	NULL

- **Display accident date and location.**

select accident\_date, location from accident;



	accident_date	location
▶	2003-01-01	Mysore Road
	2004-01-02	South End Cicle
	2005-06-07	Bull Temple Road
	2006-09-01	Kanakpura Road
	2007-10-11	My Road
	2008-03-08	Domlur

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

select driver\_id from participated where damage\_amount>=25000;



	driver_id
▶	A02
	A03

## **More Queries on INSURANCE DATABASE**

### **QUESTION(WEEK 2)**

PERSON (driver\_id: String, name: String, address: String)

CAR (reg\_num: String, model: String, year: int)

ACCIDENT (report\_num: int, accident\_date: date, location: String)

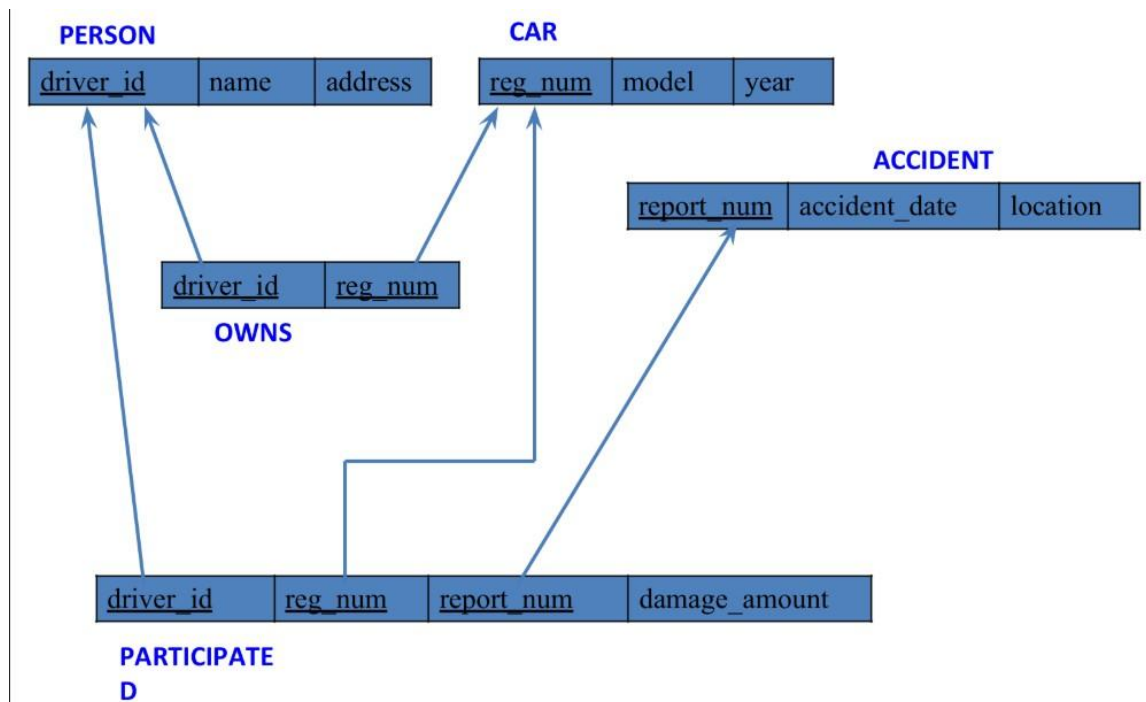
OWNS (driver\_id: String, reg\_num: String)

PARTICIPATED (driver\_id: String, reg\_num: String, report\_num: int, damage\_amount: int)

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

- Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
- Find the total number of people who owned cars that involved in accidents in 2008.
- LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER OF DAMAGE AMOUNT.
- FIND THE AVERAGE DAMAGE AMOUNT
- DELETE THE TUPLE WHOSE DAMAGE AMOUNT IS BELOW 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

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

select \* from car order by year asc;

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
reg_num	model	year		
KA031181	Lancer	1957		
KA052250	Indica	1990		
KA095477	Toyota	1998		
KA041702	Audi	2005		
KA053408	Honda	2008		
NULL	NULL	NULL		

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

select count(report\_num) from car c, participated p where p.reg\_num and c.model='Lancer';

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
count(report_num)			
0			

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

select count(distinct driver\_id) CNT from participated a, accident b where  
a.report\_num=b.report\_num and b.accident\_date like '\_\_08%';

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
CNT			
0			

**- LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER BY DAMAGE AMOUNT.**

select \* from participated order by damage\_amount desc;

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
driver_id	reg_num	report_num	damage_amount	
A02	KA053408	12	25000	
A03	KA095477	13	25000	
A01	KA052250	11	10000	
A05	KA041702	15	5000	
A04	KA031181	14	3000	
NULL	NULL	NULL	NULL	

**- FIND THE AVERAGE DAMAGE AMOUNT**

select avg(damage\_amount) from participated;

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
avg(damage_amount)			
13600.0000			

**- DELETE THE TUPLE WHOSE DAMAGE AMOUNT IS BELOW THE AVERAGE DAMAGE AMOUNT**

set sql\_safe\_updates=0;

delete from participated where damage\_amount<(select avg\_damage from(select  
avg(damage\_amount) as avg\_damage from participated) as temp);

**- 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>13600;

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	name			
▶	Pradeep			
	Smith			

**- FIND MAXIMUM DAMAGE AMOUNT.**

select max(damage\_amount) from participated;

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	max(damage_amount)			
▶	25000			

## **Bank Database**

### **QUESTION(WEEK 3)**

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

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

BankCustomer (customer-name: String, customer-street: String, customer-city: String)

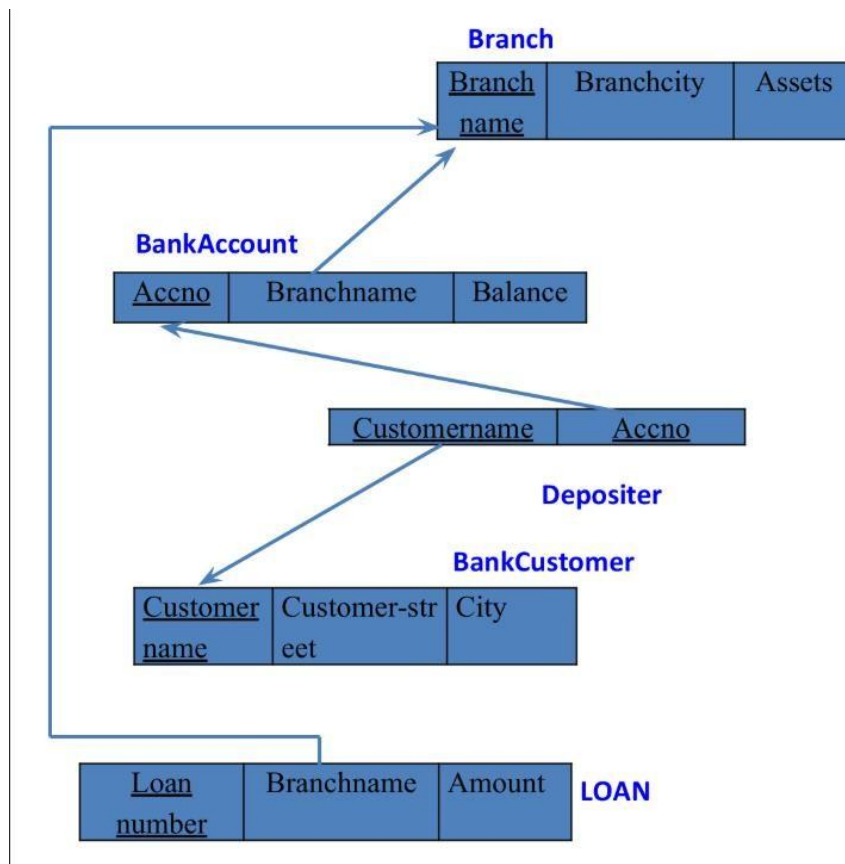
Depositer(customer-name: String, accno: int)

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

- Create the above tables by properly specifying the primary keys and the foreign keys.
- Enter at least five tuples for each relation.
- Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
- Find all the customers who have at least two accounts at the same branch (ex. SBI\_ResidencyRoad).
- CREATE A VIEW WHICH GIVES EACH BRANCH THE SUM OF THE AMOUNT OF ALL THE LOANS AT THE BRANCH.

### **SCHEMA DIAGRAM**





## CREATE DATABASE

```
create database d_bank;
```

```
use d_bank;
```

## CREATE TABLE

```
create table d_bank.branch(Branch_name varchar(30), Branch_city varchar(25), assets int, Primary key(Branch_name));
```

```
create table d_bank.BankAccount(Accno int, Branch_name varchar(30), Balance int, Primary key(Accno), Foreign key(Branch_name) references branch(Branch_name));
```

```
create table d_bank.BankCustomer(Customername varchar(20), Customer_street varchar(30), CustomerCity varchar(35), Primary key(Customername));
```

```
create table d_bank.Depositer(Customername varchar(20), Accno int, Primary key(Customername, Accno), foreign key(Accno) references BankAccount(Accno), foreign key(Customername) references BankCustomer(Customername));
```

```
create table d_bank.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 BankAccount;

Field	Type	Null	Key	Default	Extra
Accno	int	NO	PRI	NULL	
Branch_name	varchar(30)	YES	MUL	NULL	
Balance	int	YES		NULL	

desc BankCustomer;

Field	Type	Null	Key	Default	Extra
Customername	varchar(20)	NO	PRI	NULL	
Customer_street	varchar(30)	YES		NULL	
CustomerCity	varchar(35)	YES		NULL	

desc Depositer;

Field	Type	Null	Key	Default	Extra
Customername	varchar(20)	NO	PRI	NULL	
Accno	int	NO	PRI	NULL	

desc Loan;

Field	Type	Null	Key	Default	Extra
Loan_number	int	NO	PRI	NULL	
Branch_name	varchar(30)	YES	MUL	NULL	
Amount	int	YES		NULL	

## INSERTING VALUES TO THE TABLE

```

insert into branch values("HDFC_MGRoad", "Bangalore", 10000);
insert into branch values("HDFC_Jaynagar", "Bangalore", 50000);
insert into branch values("HDFC_Indiranagar", "Bangalore", 40000);
insert into branch values("HDFC_Andheri", "Mumbai", 30000); insert
into branch values("HDFC_Tnagar", "Chennai", 80000);
select * from branch;

```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
Branch_name	Branch_city	assets		
HDFC_Andheri	Mumbai	30000		
HDFC_Indiranagar	Bangalore	40000		
HDFC_Jaynagar	Bangalore	50000		
HDFC_MGRoad	Bangalore	10000		
HDFC_Tnagar	Chennai	80000		
NULL	NULL	NULL		

```

insert into BankAccount values(1, "HDFC_MGRoad", 1500);
insert into BankAccount values(2, "HDFC_Indiranagar", 10500);
insert into BankAccount values(3, "HDFC_Jaynagar", 2500);
insert into BankAccount values(4, "HDFC_MGRoad", 3500);
insert into BankAccount values(5, "HDFC_Andheri", 4500);
insert into BankAccount values(6, "HDFC_Tnagar", 5500); insert
into BankAccount values(7, "HDFC_Andheri", 6500); insert into
BankAccount values(8, "HDFC_Jaynagar", 7500); insert into
BankAccount values(9, "HDFC_Tnagar", 8500); insert into
BankAccount values(10, "HDFC_Indiranagar", 9500);
select * from BankAccount;

```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
Accno	Branch_name	Balance		
1	HDFC_MGRoad	1500		
2	HDFC_Indiranagar	10500		
3	HDFC_Jaynagar	2500		
4	HDFC_MGRoad	3500		
5	HDFC_Andheri	4500		
6	HDFC_Tnagar	5500		
7	HDFC_Andheri	6500		
8	HDFC_Jaynagar	7500		
9	HDFC_Tnagar	8500		
10	HDFC_Indiranagar	9500		
NULL	NULL	NULL		

```

insert into BankCustomer values("Ravi", "1st Cross", "Bangalore"); insert
into BankCustomer values("Priya", "Lake View Road", "Bangalore"); insert
into BankCustomer values("Kiran", "Andheri West", "Mumbai"); insert into

```

BankCustomer values("Meena", "Anna Salai", "Chennai"); insert into

BankCustomer values("Anita", "MG Road", "Bangalore");

select \* from BankCustomer;

Customername	Customer_street	CustomerCity
Anita	MG Road	Bangalore
Kiran	Andheri West	Mumbai
Meena	Anna Salai	Chennai
Priya	Lake View Road	Bangalore
Ravi	1st Cross	Bangalore
NULL	NULL	NULL

insert into Depositer values("Ravi", 1); insert

into Depositer values("Priya", 2); insert into

Depositer values("Kiran", 3); insert into

Depositer values("Ravi", 4); insert into

Depositer values("Meena", 5); insert into

Depositer values("Anita", 6); insert into

Depositer values("Kiran", 7); insert into

Depositer values("Meena", 8);

select \* from Depositer;

Customername	Accno
Ravi	1
Priya	2
Kiran	3
Ravi	4
Meena	5
Anita	6
Kiran	7
Meena	8
NULL	NULL

insert into Loan values(1, "HDFC\_MGRoad", 1000);

insert into Loan values(2, "HDFC\_Jaynagar", 2000);

insert into Loan values(3, "HDFC\_Indiranagar", 3000);

insert into Loan values(4, "HDFC\_Andheri", 4000);

insert into Loan values(5, "HDFC\_Tnagar", 5000); select

\* from Loan;

Result Grid			
Filter Rows:			
	Loan_number	Branch_name	Amount
▶	1	HDFC_MGRoad	1000
	2	HDFC_Jaynagar	2000
	3	HDFC_Indiranagar	3000
	4	HDFC_Andheri	4000
	5	HDFC_Tnagar	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'.

```
select Branch_name, CONCAT(assets/100000,"lakhs")assets_in_lakhs from branch;
```

Result Grid		
Filter Rows:		
	Branch_name	assets_in_lakhs
▶	HDFC_Andheri	0.3000lakhs
	HDFC_Indiranagar	0.4000lakhs
	HDFC_Jaynagar	0.5000lakhs
	HDFC_MGRoad	0.1000lakhs
	HDFC_Tnagar	0.8000lakhs

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

```
select d.Customername from Depositer d, BankAccount b where b.Branch_name="HDFC_MGRoad"
and d.Accno=b.Accno group by d.Customername having count(d.Accno)>=2;
```

Result Grid	
Filter Rows:	
	Customername
▶	Ravi

- 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 BankAccount group by Branch\_name;

select \* from sum\_of\_loan;

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	Branch_name	SUM(Balance)			
▶	HDFC_Andheri	11000			
	HDFC_Indiranagar	20000			
	HDFC_Jaynagar	10000			
	HDFC_MGRoad	5000			
	HDFC_Tnagar	14000			

## **More Queries On Bank Database**

### **QUESTION(WEEK 4)**

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

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

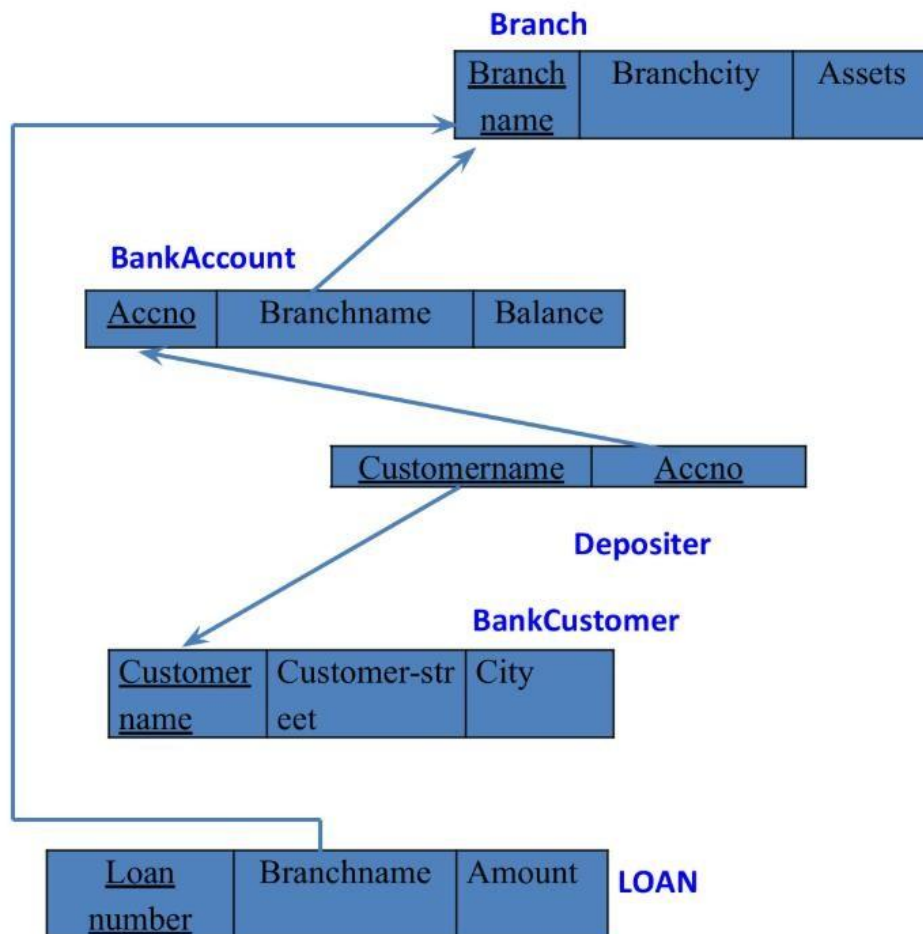
BankCustomer (customer-name: String, customer-street: String, customer-city: String)

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

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

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

## SCHEMA DIAGRAM



## QUERIES

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

```
select c.Customername from BankCustomer c join Depositer d on
c.Customername=d.Customername join BankAccount b on d.Accno=b.Accno join branch br on
b.Branch_name=br.Branch_name where br.Branch_city='Bangalore' group by d.Customername
having count(Distinct br.Branch_name);
```



Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Customername			
Kiran			
Meena			
Priya			
Ravi			

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

select distinct l.Loan\_number, l.Branch\_name from Loan l left join BankAccount b on  
l.Branch\_name = b.Branch\_name where b.Accno is NULL;

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Loan_number	Branch_name		

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

distinct c.Customername from BankCustomer c join Depositer d on  
c.Customername=d.Customername join BankAccount b on d.Accno=b.Accno join branch br on  
b.Branch\_name=br.Branch\_name join Loan l on l.Branch\_name=br.Branch\_name where  
br.Branch\_city='Bangalore';

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
Customername			
Priya			
Kiran			
Meena			
Ravi			

- 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');

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
Branch_name				
HDFC_Tnagar				
NULL				

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

```
ALTER TABLE Depositer DROP FOREIGN KEY depositer_ibfk_1; alter table depositer ADD
CONSTRAINT depositer_ibfk_1 FOREIGN KEY (Accno) REFERENCES
BankAccount(Accno) ON DELETE CASCADE;
```

```
delete from BankAccount where Branch_name in (Select Branch_name from branch where
Branch_city='Mumbai');
```

```
select * from BankAccount;
```

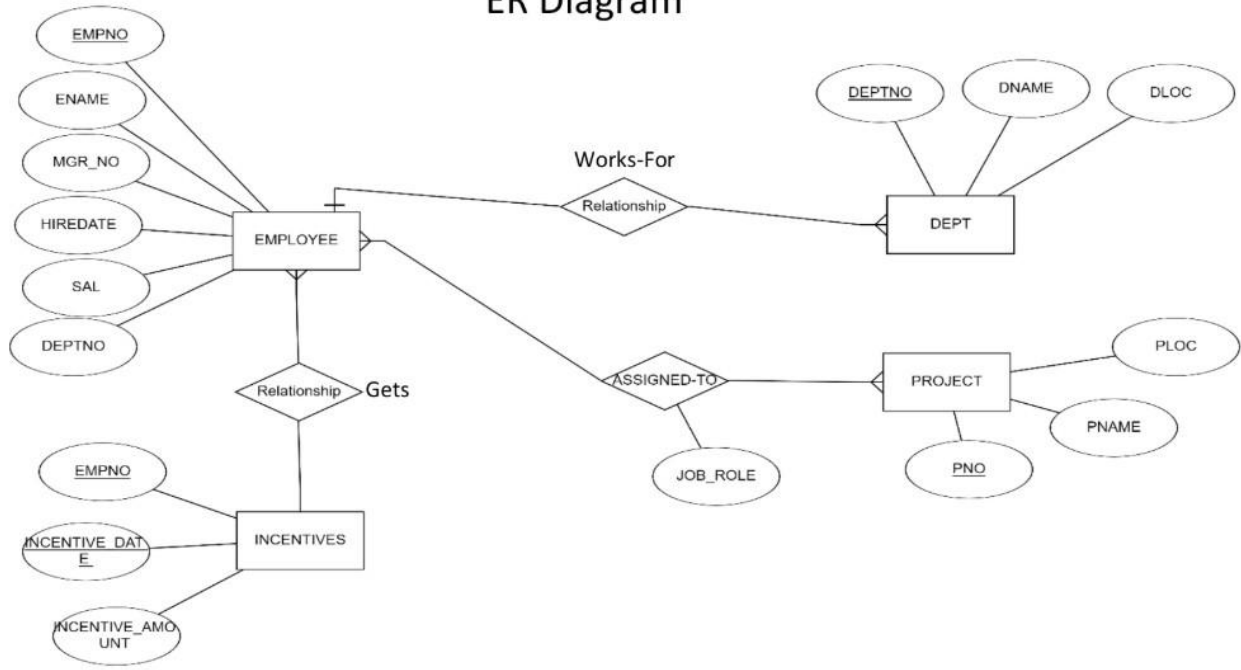
Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
Accno	Branch_name	Balance		
1	HDFC_MGRoad	1500		
2	HDFC_Indiranagar	10500		
3	HDFC_Jaynagar	2500		
4	HDFC_MGRoad	3500		
6	HDFC_Tnagar	5500		
8	HDFC_Jaynagar	7500		
9	HDFC_Tnagar	8500		
10	HDFC_Indiranagar	9500		
NULL	NULL	NULL		

- **Update the Balance of all accounts by 5% update BankAccount set Balance = Balance\*1.05 where Accno>0; select \* from BankAccount;**

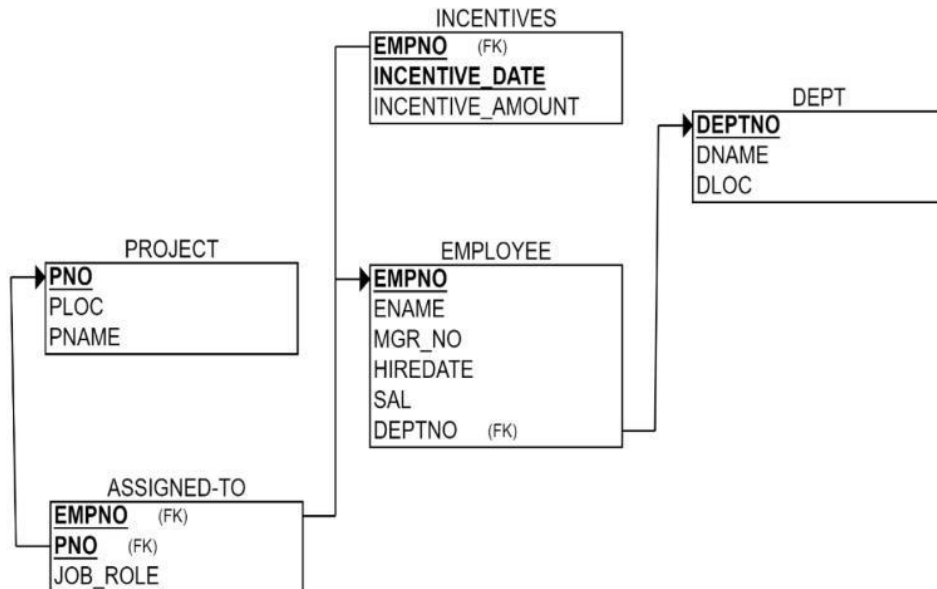
Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
Accno	Branch_name	Balance		
1	HDFC_MGRoad	1575		
2	HDFC_Indiranagar	11025		
3	HDFC_Jaynagar	2625		
4	HDFC_MGRoad	3675		
6	HDFC_Tnagar	5775		
8	HDFC_Jaynagar	7875		
9	HDFC_Tnagar	8925		
10	HDFC_Indiranagar	9975		
NULL	NULL	NULL		

## EMPLOYEE DATABASE QUESTION(WEEK 5)

ER Diagram



Schema Diagram



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. 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.

### **CREATE DATABASE**

```
create database emp; show
databases;
use emp;
```

### **CREATE TABLE**

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

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

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

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

create table assigned\_to(empno decimal(4,0) references emp(empno) on delete cascade on update cascade, pno int references project(pno) on delete cascade on update cascade, job\_role varchar(30), primary key(empno, pno));

## STRUCTURE OF THE TABLE

desc dept;

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

desc emp;

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

desc incentives;

Result Grid						
	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;

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

desc assigned\_to;

Result Grid


Filter Rows:

Export:



Wrap Cell Content:



	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,'CSE', 'MUMBAI');

INSERT INTO dept VALUES (20, 'ISE', 'BENGALURU');

INSERT INTO dept VALUES (30, 'AI ML', 'DELHI');

INSERT INTO dept VALUES (40,'CIVIL', 'CHENNAI');

INSERT INTO dept VALUES (50, 'RESEARCH', 'ASSAM');

SELECT \* FROM dept;

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	deptno	dname	loc
▶	10	CSE	MUMBAI
	20	ISE	BENGALURU
	30	AI ML	DELHI
	40	CIVIL	CHENNAI
	50	RESEARCH	ASSAM
•	NULL	NULL	NULL

INSERT INTO emp VALUES (1000,'ADI', 1009,'2012-12-17',50000.00,20);

INSERT INTO emp VALUES (1001,'ARUN',1005,'2013-02-20',15000.00,30);

INSERT INTO emp VALUES (1002,'BHOOMI',1005,'2015-02-22',12500.00,30);

INSERT INTO emp VALUES (1003,'GOVIND',1008,'2008-04-02',22500.00,20);

INSERT INTO emp VALUES (1004,'RAM',1005,'2014-09-28',12500.00,30);

INSERT INTO emp VALUES (1005,'HEMANTH',1008,'2015-05-01',28500.00,30);

INSERT INTO emp VALUES (1006,'ADAM',1008,'2017-06-09',24500.00,10);

INSERT INTO emp VALUES (1007,'EVE',1003,'2010-12-09',30000.00,20);

INSERT INTO emp VALUES (1008,'GAGAN',NULL,'2009-11-17',50000.00,10);

INSERT INTO emp VALUES (1009,'HARSHA',1005,'2017-12-03',9500.00,30);

SELECT \* FROM EMP;

empno	ename	mgr_no	hiredate	sal	deptno
1000	ADI	1009	2012-12-17	50000.00	20
1001	ARUN	1005	2013-02-20	15000.00	30
1002	BHOOMI	1005	2015-02-22	12500.00	30
1003	GOVIND	1008	2008-04-02	22500.00	20
1004	RAM	1005	2014-09-28	12500.00	30
1005	HEMANTH	1008	2015-05-01	28500.00	30
1006	ADAM	1008	2017-06-09	24500.00	10
1007	EVE	1003	2017-06-09	30000.00	20
1008	GAGAN	NULL	2009-11-17	50000.00	10
1009	HARSHA	1005	2017-12-03	9500.00	30
NULL	NULL	NULL	NULL	NULL	NULL

```

INSERT INTO incentives VALUES(1001,'2019-02-01',5000.00);
INSERT INTO incentives VALUES(1002,'2019-03-01',2500.00);
INSERT INTO incentives VALUES(1003,'2022-02-01',5070.00);
INSERT INTO incentives VALUES(1004,'2020-02-01',2000.00); INSERT
INTO incentives VALUES(1004,'2022-04-01',879.00);
INSERT INTO incentives VALUES(1002,'2019-02-01',8000.00); INSERT
INTO incentives VALUES(1005,'2019-03-01',500.00);
INSERT INTO incentives VALUES(1005,'2020-03-01',9000.00); INSERT
INTO incentives VALUES(1005,'2022-04-01',4500.00);
select * from incentives;

```

empno	incentive_date	incentive_amount
1001	2019-02-01	5000.00
1002	2019-02-01	8000.00
1002	2019-03-01	2500.00
1003	2022-02-01	5070.00
1004	2020-02-01	2000.00
1004	2022-04-01	879.00
1005	2019-03-01	500.00
1005	2020-03-01	9000.00
1005	2022-04-01	4500.00
NULL	NULL	NULL

```

INSERT INTO project VALUES(101,'CSE Project','BENGALURU');
INSERT INTO project VALUES(102,'ISE PROJECT','HYDERABAD');
INSERT INTO project VALUES(103,'AI ML PROJECT','BENGALURU');
INSERT INTO project VALUES(104,'CIVIL PROJECT','MYSURU');
INSERT INTO project VALUES(105,'RESEARCH PROJECT','PUNE');

```



SELECT \* FROM PROJECT;

Result Grid			
Filter Rows:			
	pno	pname	ploc
▶	101	CSE Project	BENGALURU
	102	ISE PROJECT	HYDERABAD
	103	AI ML PROJECT	BENGALURU
	104	CIVIL PROJECT	MYSURU
	105	RESEARCH PROJECT	PUNE
•	NULL	NULL	NULL

INSERT INTO assigned\_to VALUES(1001,101,'oftware Engineer');

INSERT INTO assigned\_to VALUES(1002,101,'Software Architect');

INSERT INTO assigned\_to VALUES(1003,101,'Project Manager');

INSERT INTO assigned\_to VALUES(1004,102,'Sales');

INSERT INTO assigned\_to VALUES(1002,102,'Software Engineer');

INSERT INTO assigned\_to VALUES(1004,103,'Cyber Security');

INSERT INTO assigned\_to VALUES(1005,104,'Software Engineer');

INSERT INTO assigned\_to VALUES(1009,105,'Software Engineer');

INSERT INTO assigned\_to VALUES(1008,104,'General Manager');

SELECT \* FROM ASSIGNED\_TO;

Result Grid			
Filter Rows:			
	empno	pno	job_role
▶	1001	101	oftware Engineer
	1002	101	Software Architect
	1002	102	Software Engineer
	1003	101	Project Manager
	1004	102	Sales
	1004	103	Cyber Security
	1005	104	Software Engineer
	1008	104	General Manager
	1009	105	Software Engineer
•	NULL	NULL	NULL

## QUERIES

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

select e.empno from emp e, assigned\_to a, project p where e.empno=a.empno and a.pno=p.pno and p.ploc in ('Bengaluru','Hyderabad','Mysuru');



Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	empno			
▶	1001			
	1002			
	1002			
	1003			
	1004			
	1004			
	1005			
	1008			

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

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

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	EMPNO			
▶	1000			
	1006			
	1007			
	1008			
	1009			

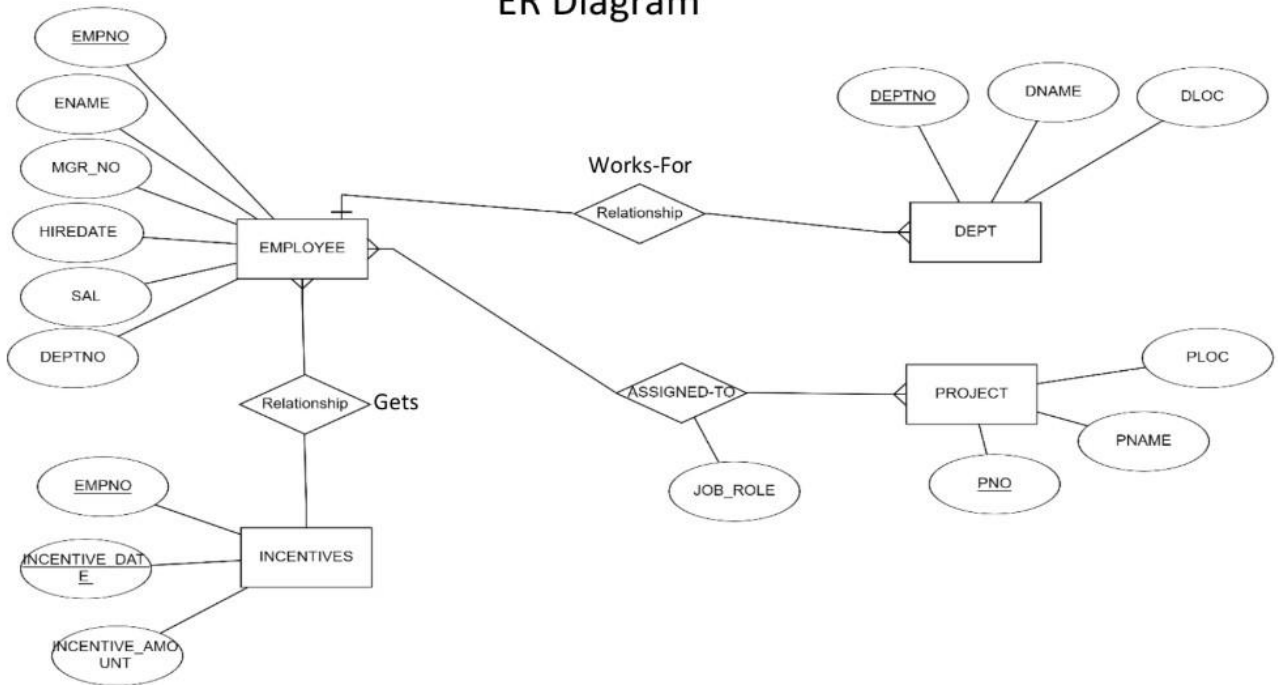
**- Write a SQL query to find the employees name, number, dept, job\_role, department location and project location who are working for a project location same as his/her department location.**

select e.empno, e.ename, d.dname, d.loc, a.job\_role, p.ploc from emp e, dept d, assigned\_to a, project p where e.deptno=d.deptno and e.empno=a.empno and a.pno=p.pno and d.loc=p.ploc;

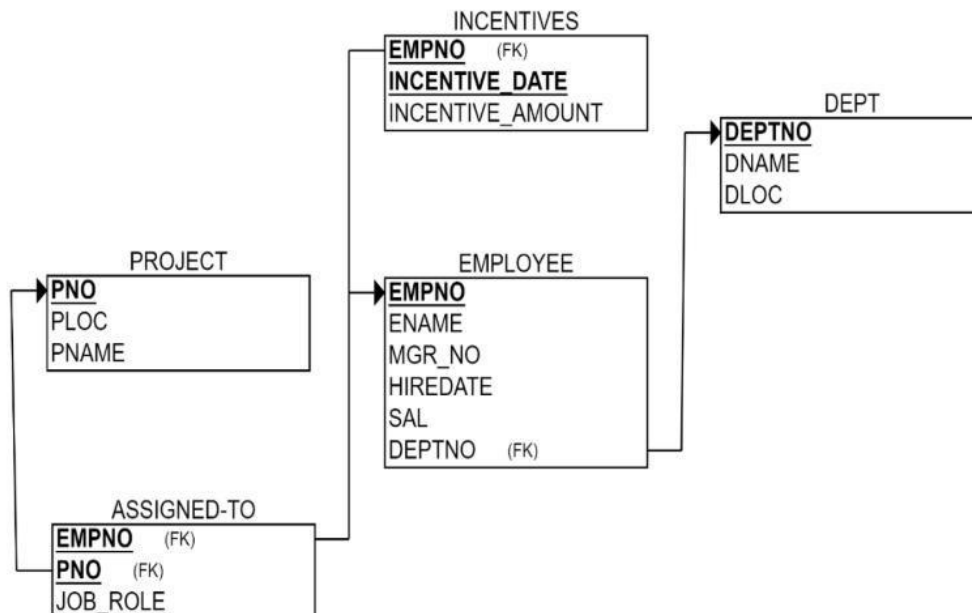
# MORE QUERIES ON EMPLOYEE DATABASE

## QUESTION(WEEK 6)

ER Diagram



Schema Diagram



1. List the name of the managers with the most employees

2. Display those managers name whose salary is more than average salary of his employee?
3. SQL Query to find the name of the top level manager of each department.
4. SQL Query to find the employee details who got second maximum incentive in February 2019.
5. Display those employees who are working in the same dept where his manager is work. ?
6. Write a SQL query to find those employees whose net pay are higher than or equal to the salary of any other employee in the company

## QUERIES

### **- List the name of the managers with the most 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);

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
ename	count(*)		
▶ HEMANTH	4		

### **- Display those managers name whose salary is more than 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);

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	empno	ename	mgr_no	hiredate	sal	deptno
▶	1005	HEMANTH	1008	2015-05-01	28500.00	30
	1008	GAGAN	NULL	2009-11-17	50000.00	10
*	NULL	NULL	NULL	NULL	NULL	NULL

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

select distinct m.mgr\_no from emp e, emp m where e.mgr\_no=m.mgr\_no and e.deptno=m.deptno and e.empno in (select distinct m.mgr\_no from emp e, emp m where e.mgr\_no=m.mgr\_no and e.deptno=m.deptno);

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	mgr_no			
▶	1005			
	1008			

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

select \* from emp e, incentives i where e.empno=i.empno and 2=(select count(\*) from incentives j where i.incentive\_amount<=j.incentive\_amount);

Result Grid

Filter Rows:

Export:

Wrap Cell Content:

	empno	ename	mgr_no	hiredate	sal	deptno	empno	incentive_date	incentive_amount
▶	1002	BHOOMI	1005	2015-02-22	12500.00	30	1002	2019-02-01	8000.00

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

select \* from emp e where e.deptno=(select e1.deptno from emp e1 where e1.empno=e.mgr\_no);

Result Grid

Filter Rows:

Edit:

Export/Import:

Wrap Cell Content:

	empno	ename	mgr_no	hiredate	sal	deptno
▶	1001	ARUN	1005	2013-02-20	15000.00	30
	1002	BHOOMI	1005	2015-02-22	12500.00	30
	1004	RAM	1005	2014-09-28	12500.00	30
	1006	ADAM	1008	2017-06-09	24500.00	10
	1007	EVE	1003	2010-12-09	30000.00	20
	1009	HARSHA	1005	2017-12-03	9500.00	30
•	NULL	NULL	NULL	NULL	NULL	NULL

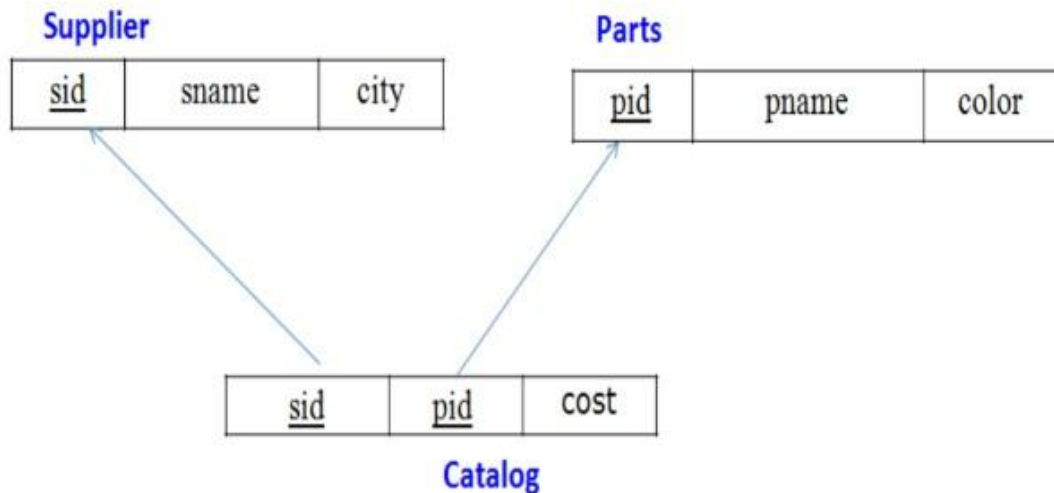
- Write a SQL query to find those employees whose net pay are higher than or equal to the salary of any other employee in the company

```
select distinct e.ename from emp e, incentives i where (select max(sal+incentive_amount) from emp,incentives)>=any(select sal from emp e1 where e.deptno=e1.deptno);
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	ename			
▶	ADI			
	ARUN			
	BHOOMI			
	GOVIND			
	RAM			
	HEMANTH			
	ADAM			
	EVE			
	GAGAN			
	HARSHA			

## SUPPLIER DATABASE QUESTION(WEEK 7)

### Schema Diagram



1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2. Insert appropriate records in each table.
3. Find the pnames of parts for which there is some supplier.
4. Find the snames of suppliers who supply every part.
5. Find the snames of suppliers who supply every red part.
6. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

7. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).
8. For each part, find the sname of the supplier who charges the most for that part.

## CREATE DATABASE

```
create database sup;
```

```
use sup;
```

## CREATE TABLE

```
create table Supplier(sid decimal(3,0) primary key, sname varchar(50), city varchar(50));
```

```
create table Parts(pid decimal(3,0) primary key, pname varchar(50), color varchar(20));
```

```
create table Catalog(sid decimal(3,0), pid decimal(3,0), cost decimal(2,0), primary key(sid, pid),  
foreign key(sid) references Supplier(sid), foreign key(pid) references Parts(pid));
```

## STRUCTURE OF TABLE

```
desc Supplier;
```

Result Grid						
		Filter Rows:				
		Export:				
		Wrap Cell Content:				
	Field	Type	Null	Key	Default	Extra
▶	sid	decimal(3,0)	NO	PRI	NULL	
	sname	varchar(50)	YES		NULL	
	city	varchar(50)	YES		NULL	

```
desc Parts;
```

Result Grid						
		Filter Rows:				
		Export:				
		Wrap Cell Content:				
	Field	Type	Null	Key	Default	Extra
▶	pid	decimal(3,0)	NO	PRI	NULL	
	pname	varchar(50)	YES		NULL	
	color	varchar(20)	YES		NULL	

```
desc Catalog;
```

Result Grid						
	Field	Type	Null	Key	Default	Extra
▶	sid	decimal(3,0)	NO	PRI	NULL	
	pid	decimal(3,0)	NO	PRI	NULL	
	cost	decimal(2,0)	YES		NULL	

## INSERTING VALUES TO THE TABLE

INSERT INTO Supplier VALUES(101, 'Acme Widget', 'Bangalore');

INSERT INTO Supplier VALUES(102, 'Johns', 'Kolkata');

INSERT INTO Supplier VALUES(103, 'Vimal', 'Mumbai'); INSERT

INTO Supplier VALUES(104, 'Reliance', 'Delhi');

select \* from Supplier;

Result Grid			
	sid	sname	city
▶	101	Acme Widget	Bangalore
	102	Johns	Kolkata
	103	Vimal	Mumbai
	104	Reliance	Delhi
*	NULL	NULL	NULL

INSERT INTO Parts VALUES(201, 'Book', 'Red');

INSERT INTO Parts VALUES(202, 'Pen', 'Red');

INSERT INTO Parts VALUES(203, 'Pencil', 'Green');

INSERT INTO Parts VALUES(204, 'Mobile', 'Green'); INSERT

INTO Parts VALUES(205, 'Charger', 'Black');

select \* from Parts;

Result Grid			
	pid	pname	color
▶	201	Book	Red
	202	Pen	Red
	203	Pencil	Green
	204	Mobile	Green
	205	Charger	Black
*	NULL	NULL	NULL

INSERT INTO Catalog VALUES(101, 201, 10);

INSERT INTO Catalog VALUES(101, 202, 10);



```

INSERT INTO Catalog VALUES(101, 203, 30);
INSERT INTO Catalog VALUES(101, 204, 10);
INSERT INTO Catalog VALUES(102, 201, 10);
INSERT INTO Catalog VALUES(102, 202, 10);
INSERT INTO Catalog VALUES(103, 203, 30); INSERT
INTO Catalog VALUES(104, 203, 40);
select * from Catalog;

```

Result Grid	Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
sid	pid	cost		
101	201	10		
101	202	10		
101	203	30		
101	204	10		
102	201	10		
102	202	10		
103	203	30		
104	203	40		
NULL	NULL	NULL		

## QUERIES

**- 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;
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
pname			
Book			
Pen			
Pencil			
Mobile			

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

```
select s.sname from Supplier s join Catalog c on s.sid=c.sid group by s.sid, s.sname having
count(distinct c.pid)=(select count(*) from Parts);
```

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
sname			

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

select s.sname from Supplier s join Catalog c on s.sid=c.sid join Parts p on p.pid=c.pid where p.color='Red' group by s.sid, s.sname having count(distinct p.pid)=(select count(\*) from Parts where color='Red');

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
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 join Catalog c on p.pid=c.pid join Supplier s on s.sid=c.sid group by p.pid, p.pname having count(distinct s.sid)=1 and max(s.sname)='Acme Widget';

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
pname			
Mobile			

**- 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);

Result Grid	Filter Rows:	Export:	Wrap Cell Content:
sid			
104			

**- For each part, find the sname of the supplier who charges the most for that part.** select p.pid, s.sname from Parts p, Supplier s, Catalog c where p.pid=c.pid and s.sid=c.sid and c.cost=(select max(cost) from Catalog where pid=p.pid);

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	pid	sname			
▶	201	Acme Widget			
	202	Acme Widget			
	204	Acme Widget			
	201	Johns			
	202	Johns			
	203	Reliance			

## Extra queries on Supplier database

1. Find the most expensive part overall and the supplier who supplies it.
2. Find suppliers who do NOT supply any red parts.
3. Show each supplier and total value of all parts they supply.
4. Find suppliers who supply at least 2 parts cheaper than ₹20.
5. List suppliers who offer the cheapest cost for each part.
6. Create a view showing suppliers and the total number of parts they supply.
7. Create a view of the most expensive supplier for each part.
8. Create a Trigger to prevent inserting a Catalog cost below 1.
9. Create a trigger to set to default cost if not provided.

## QUERIES

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

```
SELECT s.sid, s.sname, p.pid, p.pname, c.cost FROM Supplier s JOIN Catalog c ON s.sid = c.sid
JOIN Parts p ON p.pid = c.pid WHERE c.cost = (SELECT MAX(cost) FROM Catalog);
```

Result Grid						Filter Rows:	Export:	Wrap Cell Content:
	sid	sname	pid	pname	cost			
▶	104	Reliance	203	Pencil	40			

### - Find suppliers who do NOT supply any red parts.

```
SELECT s.sid, s.sname FROM Supplier s WHERE s.sid NOT IN ( SELECT c.sid FROM Catalog c
JOIN Parts p ON c.pid = p.pid WHERE p.color = 'Red');
```

Result Grid			Filter Rows:	Edit:	Export/Import:	Wrap Cell Content:
	sid	sname				
▶	103	Vimal				
	104	Reliance				
*	NULL	NULL				

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

```
SELECT s.sid, s.sname, SUM(c.cost) AS total_value FROM Supplier s JOIN Catalog c ON
s.sid = c.sid GROUP BY s.sid, s.sname;
```

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
	sid	sname	total_value			
▶	101	Acme Widget	60			
	102	Johns	20			
	103	Vimal	30			
	104	Reliance	40			

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

```
SELECT s.sid, s.sname FROM Supplier s JOIN Catalog c ON s.sid = c.sid WHERE c.cost < 20
GROUP BY s.sid, s.sname HAVING COUNT(*) >= 2;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
	sid	sname			
▶	101	Acme Widget			
	102	Johns			

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

```
SELECT c.sid, s.sname, c.pid, p.pname, c.cost FROM Catalog c JOIN Parts p ON c.pid = p.pid
JOIN Supplier s ON c.sid = s.sid WHERE c.cost = (SELECT MIN(c2.cost) FROM Catalog c2
WHERE c2.pid = c.pid);
```

Result Grid						Filter Rows:	Export:	Wrap Cell Content:
	sid	sname	pid	pname	cost			
▶	101	Acme Widget	201	Book	10			
	101	Acme Widget	202	Pen	10			
	101	Acme Widget	203	Pencil	30			
	101	Acme Widget	204	Mobile	10			
	102	Johns	201	Book	10			
	102	Johns	202	Pen	10			
	103	Vimal	203	Pencil	30			

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

```
CREATE VIEW SupplierPartCount AS SELECT s.sid, s.sname, COUNT(c.pid) AS total_parts
FROM Supplier s LEFT JOIN Catalog c ON s.sid = c.sid GROUP BY s.sid, s.sname; Select
* from SupplierPartCount;
```

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
	sid	sname	total_parts			
▶	101	Acme Widget	4			
	102	Johns	2			
	103	Vimal	1			
	104	Reliance	1			

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

```
CREATE VIEW MaxCostSupplier AS SELECT c.sid, s.sname, c.pid, p.pname, c.cost
FROM Catalog c JOIN Supplier s ON c.sid = s.sid JOIN Parts p ON c.pid = p.pid
WHERE c.cost = ( SELECT MAX(c2.cost) FROM Catalog c2 WHERE c2.pid = c.pid );
Select * from MaxCostSupplier;
```

Result Grid						Filter Rows:	Export:	Wrap Cell Content:
	sid	sname	pid	pname	cost			
▶	101	Acme Widget	201	Book	10			
	101	Acme Widget	202	Pen	10			
	101	Acme Widget	204	Mobile	10			
	102	Johns	201	Book	10			
	102	Johns	202	Pen	10			
	104	Reliance	203	Pencil	40			

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

```

DELIMITER $$
CREATE TRIGGER CheckCostBeforeInsert
BEFORE INSERT ON Catalog
FOR EACH ROW
BEGIN
    IF NEW.cost < 1 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Cost cannot be less than 1';
    END IF;
END $$
DELIMITER ;

```

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

```

DELIMITER $$
CREATE TRIGGER DefaultCost
BEFORE INSERT ON Catalog
FOR EACH ROW
BEGIN
    IF NEW.cost IS NULL THEN
        SET NEW.cost = 10;
    END IF;
END $$
DELIMITER ;

```

## **NoSQL STUDENT DATABASE QUESTION(WEEK 8)**

Perform the following DB operations using MongoDB.

- i. Create a database “Student” with the following attributes Rollno, Age, ContactNo, EmailId.
- ii. Insert appropriate values
- iii. Write query to update Email-Id of a student with rollno 10.
- iv. Replace

the student name from “ABC” to “FEM” of rollno 11. v. Export the created table into local file system vi. Drop the table.  
vii. Import a given csv dataset from local file system into mongodb collection.

## QUERIES

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

use student

```
test> use student
switched to db student
```

**- Insert appropriate values**

```
db.students.insertMany([
  { Rollno: 1, Age: 20, ContactNo: "9876543211", EmailId: "s1@gmail.com", Name: "AAA" },
  { Rollno: 2, Age: 21, ContactNo: "9876543212", EmailId: "s2@gmail.com", Name: "BBB" },
  { Rollno: 3, Age: 22, ContactNo: "9876543213", EmailId: "s3@gmail.com", Name: "CCC" },
  { Rollno: 10, Age: 20, ContactNo: "9876543214", EmailId: "old10@gmail.com", Name: "DDD" },
  { Rollno: 11, Age: 21, ContactNo: "9876543215", EmailId: "abc@gmail.com", Name: "ABC" }
])
```

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693ee5a338276760d51e2621'),
    '1': ObjectId('693ee5a338276760d51e2622'),
    '2': ObjectId('693ee5a338276760d51e2623'),
    '3': ObjectId('693ee5a338276760d51e2624'),
    '4': ObjectId('693ee5a338276760d51e2625')
  }
}
```

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

```
db.students.updateOne({ Rollno: 10 }, { $set: { EmailId: "new10@gmail.com" } })
```

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

```
db.students.updateOne({ Rollno: 11 }, { $set: { Name: "FEM" } })
```

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

### - Export the created table into local file system

```
exit
```

```
mongoexport --db student --collection students --out students.json
```

```
2025-12-14T23:54:57.237+0530    connected to: mongodb://localhost/
2025-12-14T23:54:57.239+0530    exported 5 records
```

### - Drop the table

```
mongosh
```

```
use student
```

```
db.students.drop()
```

```
...
true
```



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

exit

```
mongoimport --db student --collection students --type csv --headerline --file students.csv
```

```
2025-12-15T00:01:41.016+0530    connected to: mongodb://localhost/
2025-12-15T00:01:41.032+0530    5 document(s) imported successfully. 0 document(s) failed to import.
```

## **NoSQL CUSTOMER DATABASE**

### **QUESTION(WEEK 9)**

Perform the following DB operations using MongoDB.

- i. Create a collection by name Customers with the following attributes. Cust\_id, Acc\_Bal, Acc\_Type
- ii. Insert at least 5 values into the table.
- iii. Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer\_id.
- iv. Determine Minimum and Maximum account balance for each customer\_id.
- v. Export the created collection into local file system.
- vi. Drop the table.

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

### QUERIES

- Create a collection by name Customers with the following attributes.  
Cust\_id, Acc\_Bal, Acc\_Type

use Bank

```
...  
switched to db Bank
```

- Insert at least 5 values into the table.

```
db.Customers.insertMany([  
  { Cust_id: 1, Acc_Bal: 500, Acc_Type: "Z" },  
  { Cust_id: 1, Acc_Bal: 800, Acc_Type: "Z" },  
  { Cust_id: 2, Acc_Bal: 1500, Acc_Type: "Z" },  
  { Cust_id: 3, Acc_Bal: 400, Acc_Type: "A" },  
  { Cust_id: 4, Acc_Bal: 2000, Acc_Type: "Z" }  
)
```

```
{  
  acknowledged: true,  
  insertedIds: {  
    '0': ObjectId('693f03b6dc08fb76311e2621'),  
    '1': ObjectId('693f03b6dc08fb76311e2622'),  
    '2': ObjectId('693f03b6dc08fb76311e2623'),  
    '3': ObjectId('693f03b6dc08fb76311e2624'),  
    '4': ObjectId('693f03b6dc08fb76311e2625')  
  }  
}
```

- 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 } } } ])
```

```
[
  { _id: 4, TotalBalance: 2000 },
  { _id: 1, TotalBalance: 1300 },
  { _id: 2, TotalBalance: 1500 }
]
```

### - 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: 4, MinBalance: 2000, MaxBalance: 2000 },
  { _id: 2, MinBalance: 1500, MaxBalance: 1500 },
  { _id: 1, MinBalance: 500, MaxBalance: 800 },
  { _id: 3, MinBalance: 400, MaxBalance: 400 }
]
```

### - Export the created collection into local file system.

exit

```
mongoexport --db Bank --collection Customers --out customers.json
```

```
2025-12-15T00:08:15.623+0530    connected to: mongodb://localhost/
2025-12-15T00:08:15.626+0530    exported 5 records
```

### - Drop the table.

```
mongosh use
```

```
Bank
```

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

```
...
true
```

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

exit

```
mongoimport --db Bank --collection Customers --type csv --headerline --file customers.csv
```

```
2025-12-15T00:13:38.817+0530    connected to: mongodb://localhost/
2025-12-15T00:13:38.833+0530    5 document(s) imported successfully. 0 document(s) failed to import.
```

## **NoSQL RESTAURANT DATABASE QUESTION(WEEK 10)**

Perform the following DB operations using MongoDB. i.

Write NoSQL Queries on “Restaurant” collection.

- ii. Write a MongoDB query to display all the documents in the collection restaurants.
- iii. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.
- iv. 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.
- v. Write a MongoDB query to find the average score for each restaurant.

## **QUERIES**

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

use restaurant

```
switched to db restaurant
```

```
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('6940128b35591ce5311e2621'),
    '1': ObjectId('6940128b35591ce5311e2622'),
    '2': ObjectId('6940128b35591ce5311e2623'),
    '3': ObjectId('6940128b35591ce5311e2624'),
    '4': ObjectId('6940128b35591ce5311e2625')
  }
}
```

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

```
db.restaurants.find()
```

```

[
  {
    _id: ObjectId('6940128b35591ce5311e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]

```

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

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

```

...
[
  {
    _id: ObjectId('6940128b35591ce5311e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('6940128b35591ce5311e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]

```

**- 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 }},{ restaurantId: 1, name: 1, town: 1, cuisine: 1, _id: 0 })
```



```
[
  { name: 'Meghna Foods', town: 'Jayanagar', cuisine: 'Indian' },
  { name: 'Empire', town: 'MG Road', cuisine: 'Indian' },
  { name: 'Kyotos', town: 'Majestic', cuisine: 'Japanese' },
  { name: 'WOW Momos', town: 'Malleshwaram', cuisine: 'Indian' }
]
```

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

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

```
[
  { _id: 'Kyotos', averageScore: 9 },
  { _id: 'Chinese WOK', averageScore: 12 },
  { _id: 'Meghna Foods', averageScore: 8 },
  { _id: 'Empire', averageScore: 7 },
  { _id: 'WOW Momos', averageScore: 5 }
]
```



