VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

Monisha S (1BM24CS174)

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

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-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 **Monisha S (1BM24CS174),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

Divyashree S Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

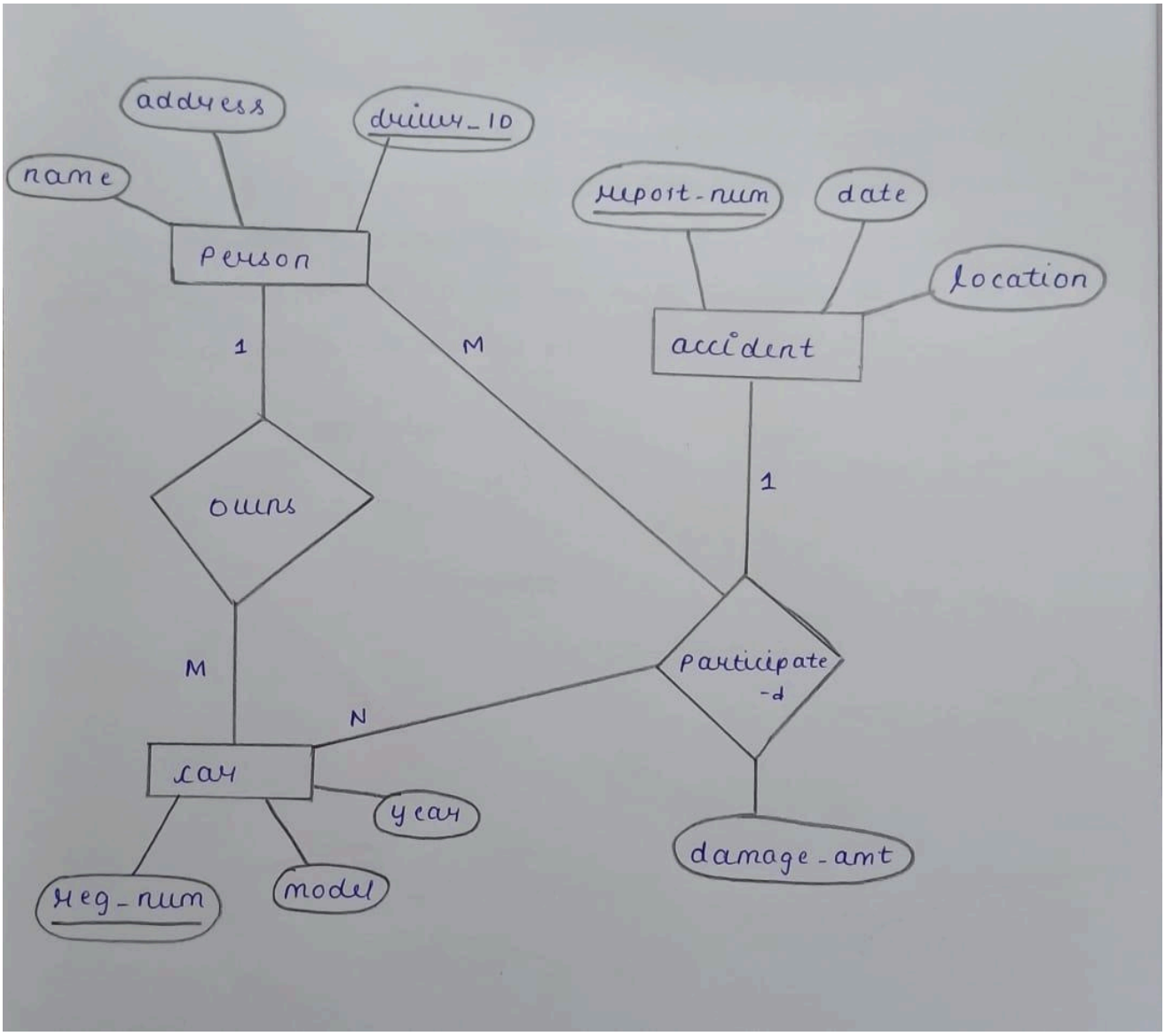
Index

Week 1 : Insurance Database

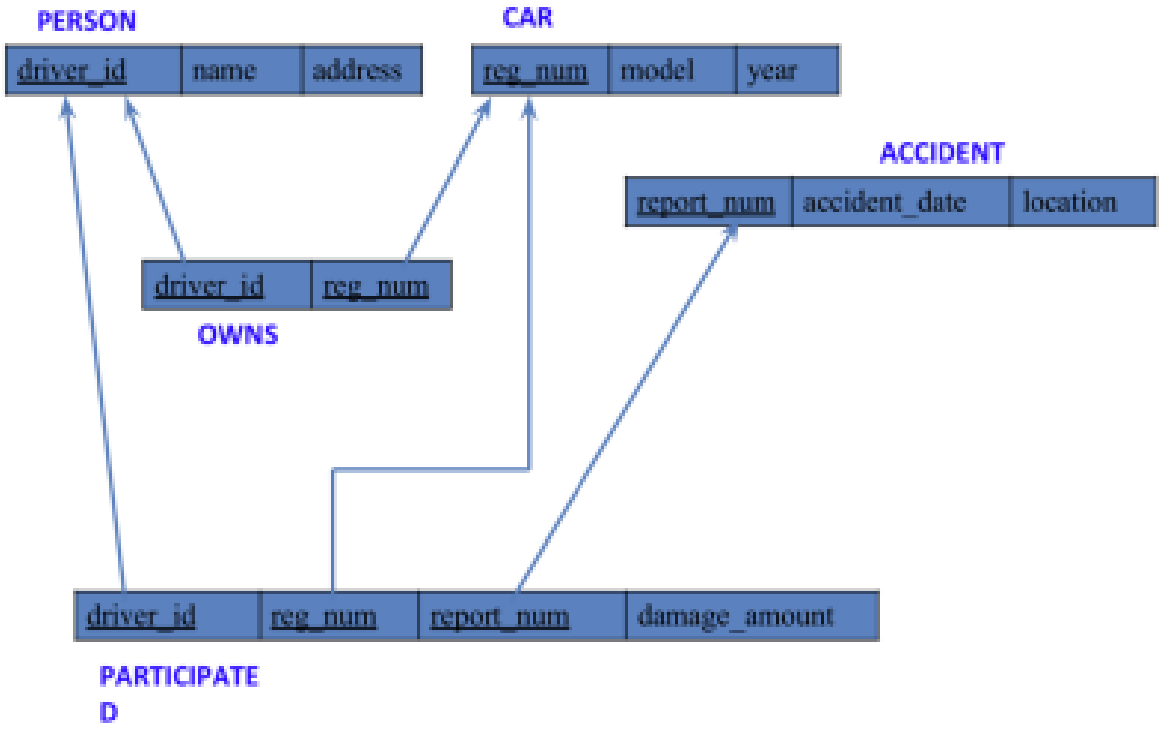
Question:

- 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

Entity Relationship Diagram



Schema Diagram



Create database

```
1 •  CREATE DATABASE NEWDATABASE;
2 •  USE NEWDATABASE;
3
```

Create table

```
4  CREATE TABLE IF NOT EXISTS PERSON(
5  driver_id varchar(50),
6  name varchar(50),
7  address varchar(100),
8  PRIMARY KEY(driver_id));
9
10 DESC PERSON;
11
```

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

Field	Type	Null	Key	Default	Extra
driver_id	varchar(50)	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
address	varchar(100)	YES		NULL	

```
11 CREATE TABLE IF NOT EXISTS CAR(
12 reg_num varchar(50),
13 model varchar(50),
14 year int,
15 PRIMARY KEY(reg_num));
16 DESC CAR;
```

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

Field	Type	Null	Key	Default	Extra
reg_num	varchar(50)	NO	PRI	NULL	
model	varchar(50)	YES		NULL	
year	int	YES		NULL	

```
18 • CREATE TABLE IF NOT EXISTS ACCIDENT(
19     report_num int,
20     accident_date date,
21     location varchar(50),
22     PRIMARY KEY(report_num));
23 •  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(50)	YES		NULL	

Inserting Values to the table

```
32 • CREATE TABLE IF NOT EXISTS PARTICIPATED(
33     driver_id varchar(50),
34     reg_num varchar(50),
35     report_num int,
36     damage_amount int,
37     foreign key(driver_id) references person(driver_id),
38     foreign key(reg_num) references car(reg_num),
39     foreign key(report_num) references accident(report_num));
40 •  DESC PARTICIPATED;
```

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

Field	Type	Null	Key	Default	Extra
driver_id	varchar(50)	YES	MUL	NULL	
reg_num	varchar(50)	YES	MUL	NULL	
report_num	int	YES	MUL	NULL	
damage_amount	int	YES		NULL	

```
42 •  INSERT INTO ACCIDENT VALUES(11, '2003-01-01','Mysore Road'),
43    (12,'2004-02-02','South end circle'),
44    (13,'2003-01-21','Bull temple Road'),
45    (14,'2008-02-17','Mysore Road'),
46    (15,'2004-03-05','Kanakpura Road');
47
48 •  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	2004-02-02	South end circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore Road
15	2004-03-05	Kanakpura Road
NULL	NULL	NULL

```
49 •  INSERT INTO PERSON VALUES('A01','Richard','Srinivas Nagar'),
50    ('A02','Pradeep','Rajajinagar'),
51    ('A03','Smith','Ashok Nagar'),
52    ('A04','Venu','N R Colony'),
53    ('A05','Jhon','Hanumanth nagar');
54 •  SELECT * FROM PERSON;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

driver_id	name	address
A01	Richard	Srinivas Nagar
A02	Pradeep	Rajajinagar
A03	Smith	Ashok Nagar
A04	Venu	N R Colony
A05	Jhon	Hanumanth nagar
NULL	NULL	NULL

```
56 •  INSERT INTO CAR VALUES('KA052250','Indica',1990),
57     ('KA031181','Lancer',1957),
58     ('KA095477','Toyota',1998),
59     ('KA053408','Honda',2008),
60     ('KA041702','Audi',2005);
61 •  SELECT * FROM CAR;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

reg_num	model	year
KA031181	Lancer	1957
KA041702	Audi	2005
KA052250	Indica	1990
KA053408	Honda	2008
KA095477	Toyota	1998
NULL	NULL	NULL

```
63 •  INSERT INTO OWNS VALUES('A01','KA052250'),
64     ('A02','KA053408'),
65     ('A03','KA031181'),
66     ('A04','KA095477'),
67     ('A05','KA041702');
68 •  SELECT * FROM OWNS;
```

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

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

```
70 •  INSERT INTO PARTICIPATED VALUES('A01','KA052250',11,10000),
71     ('A02','KA053408',12,50000),
72     ('A03','KA095477',13,25000),
73     ('A04','KA031181',14,3000),
74     ('A05','KA041702',15,5000);
75 •  SELECT * FROM PARTICIPATED;
```

Result Grid | Filter Rows: | Export: | 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

Queries

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

```
77 •  UPDATE PARTICIPATED
78    SET damage_amount=25000
79    WHERE reg_num='KA053408' and report_num=12;
80 •  SELECT * FROM PARTICIPATED;
```

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

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

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

```
13 •  select count(distinct driver_id) cnt from participated p, accident a
14    where p.report_num=a.report_num and
15    a.accident_date like '%08%';
```

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

cnt
1

3. Adding a new accident to the database.

```
5 •  insert into accident values(16,'2008-03-08','Domlur');
6 •  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	2004-02-02	South end circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore Road
15	2004-03-05	Kanakpura Road
16	2008-03-08	Domlur
NULL	NULL	NULL

4. Display Accident date and location

```
8 •  select accident_date as date,location from accident;
```

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

date	location
2003-01-01	Mysore Road
2004-02-02	South end circle
2003-01-21	Bull temple Road
2008-02-17	Mysore Road
2004-03-05	Kanakpura Road
2008-03-08	Domlur

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

```
17 •  select driver_id from participated
18    where damage_amount >= 25000;
```

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

driver_id
A02
A03

Week 2: More Queries on Insurance Database

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

```
13 •  select * from car order by year asc;
```

Result Grid | Filter Rows: | Edit:

reg_num	model	year
KA031181	Lancer	1957
KA052250	Indica	1990
KA095477	Toyota	1998
KA041702	Audi	2005
KA053408	Honda	2008
NULL	NULL	NULL

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

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

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

count(report_num)
1

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

```
13 •  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
1

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

```
18 •  SELECT * FROM PARTICIPATED ORDER BY DAMAGE_AMOUNT DESC;
```

Result Grid | Filter Rows: | Edit: | Export/Impo

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

5. Find the Average Damage Amount.

```
68 •  SELECT AVG(DAMAGE_AMOUNT) FROM PARTICIPATED;
```

Result Grid | Filter Rows: | Export: | Wra

AVG(DAMAGE_AMOUNT)
13600.0000

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

```
1 •  DELETE FROM PARTICIPATED WHERE DAMAGE_AMOUNT < (SELECT avg_damage FROM (SELECT AVG(DAMAGE_AMOUNT) AS avg_damage FROM PARTICIPATED) AS temp);
2 •  SET SQL_SAFE_UPDATES = 0;
3 •  SELECT * FROM PARTICIPATED;
4 •  SET SQL_SAFE_UPDATES = 1;
5 •  SELECT * FROM PARTICIPATED;
```

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
NULL	NULL	NULL	NULL

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

```
70 •  SELECT NAME FROM PERSON A, PARTICIPATED B WHERE A.DRIVER_ID = B.DRIVER_ID
71    AND DAMAGE_AMOUNT>(SELECT AVG(DAMAGE_AMOUNT) FROM PARTICIPATED);
```

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

NAME
Pradeep
Smith

8. Find Maximum Damage Amount.

```
73 •  SELECT MAX(DAMAGE_AMOUNT) FROM PARTICIPATED;
```

Result Grid | Filter Rows: | Export: | Wr

MAX(DAMAGE_AMOUNT)
25000

Week 3: Bank Database

Question:

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

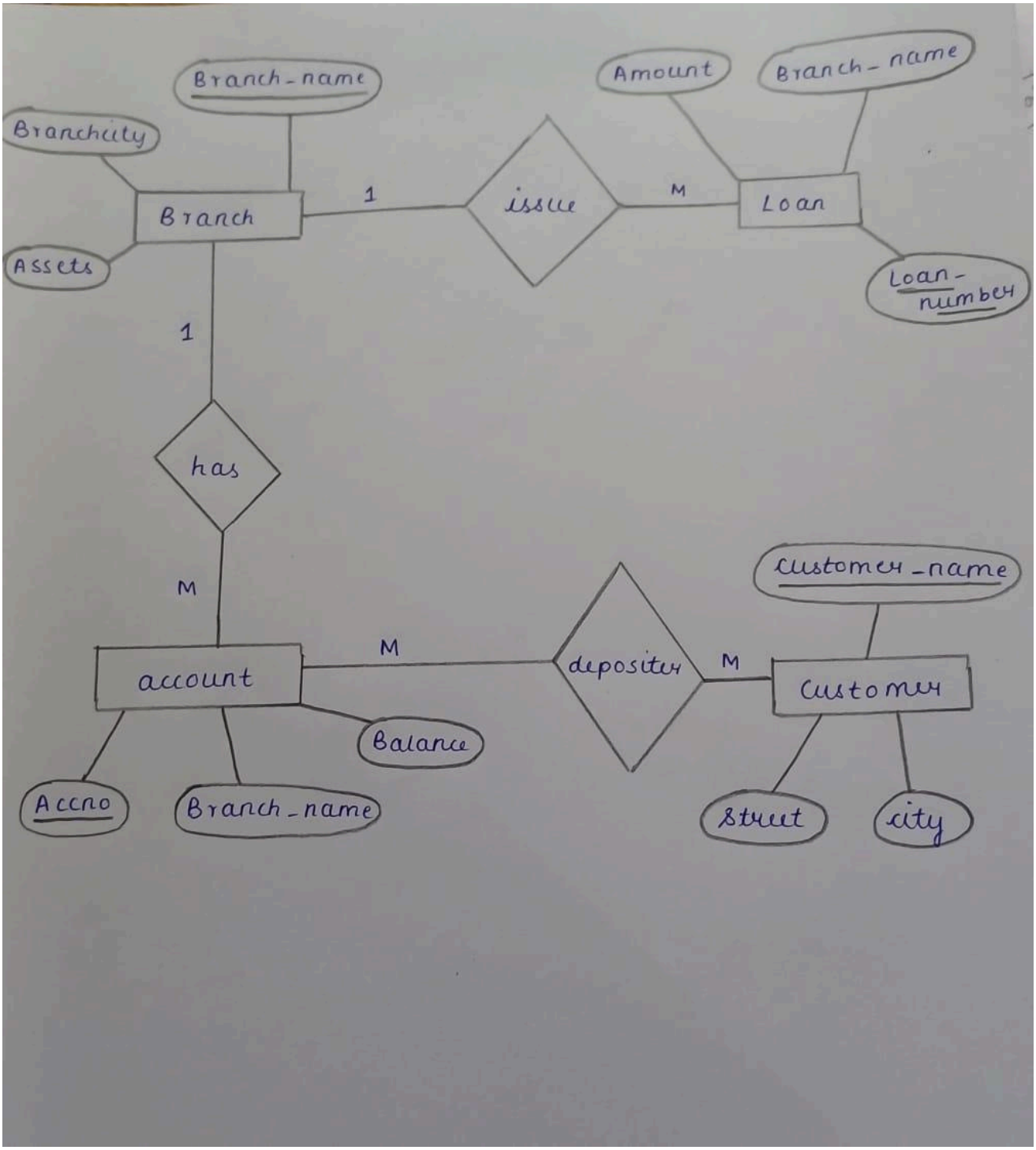
BankAccount(accno: int, branch-name: String, balance: real)

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

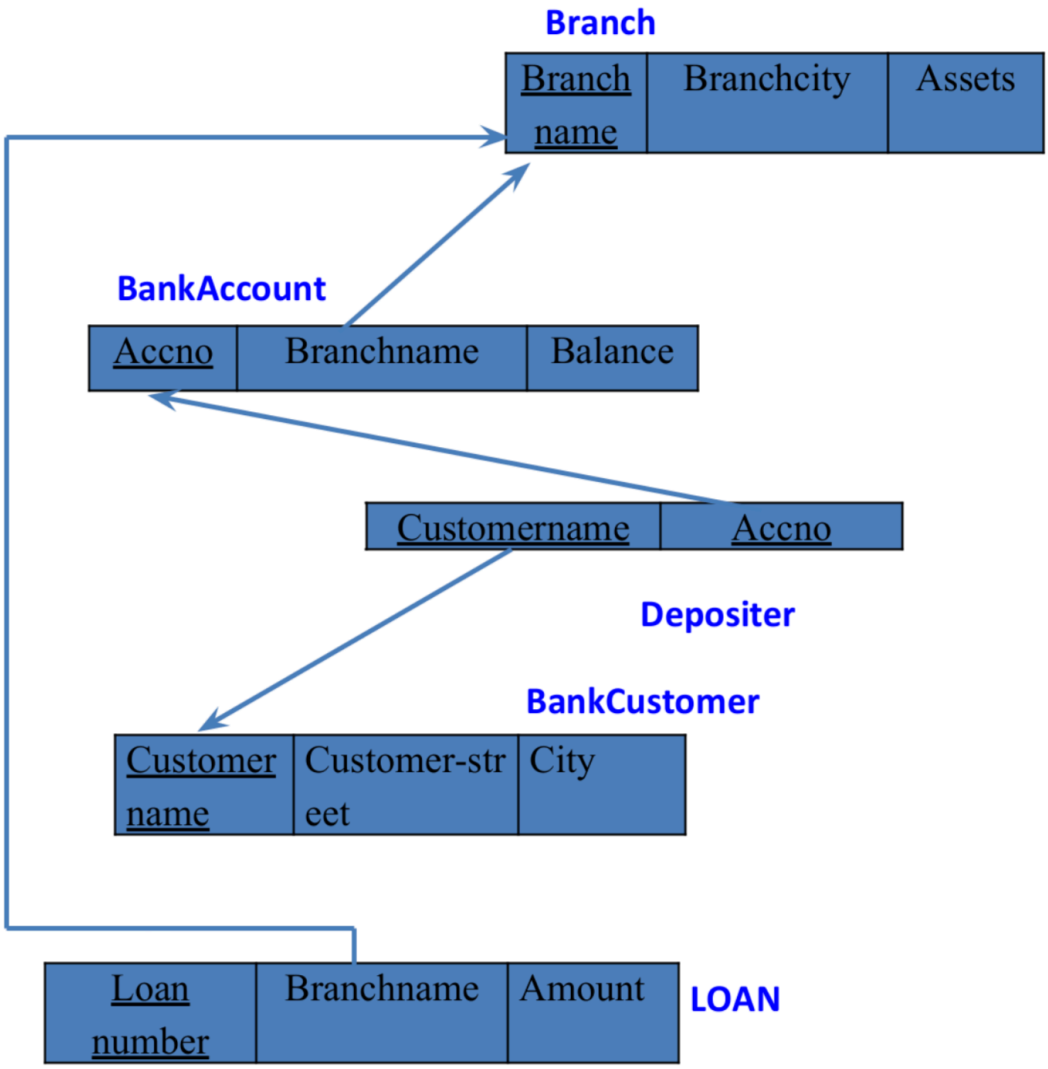
Depositer(customer-name: String, accno: int)

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

Entity Relationship Diagram



Schema Diagram



Create database

```
create database dhiksha_bank;
use dhiksha_bank;
```

Create table

```
create table dhiksha_bank.branch(
Branch_name varchar(30),
Branch_city varchar(25),
assets int,
PRIMARY KEY (Branch_name)
);
create table dhiksha_bank.BankAccount(
Accno int,
Branch_name varchar(30),
Balance int,
PRIMARY KEY(Accno),
foreign key (Branch_name) references branch(Branch_name)
);
create table dhiksha_bank.BankCustomer(
Customername varchar(20),
Customer_street varchar(30),
CustomerCity varchar (35),
PRIMARY KEY(Customername)
);
```

```
create table dhiksha_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 dhiksha_bank.Loan(
Loan_number int,
Branch_name varchar(30),
Amount int,
PRIMARY KEY(Loan_number),
foreign key (Branch_name) references branch(Branch_name)
);
```

Inserting Values to the table

```
INSERT INTO Branch VALUES ('SBI_Chamrajpet', 'Bangalore', 50000)
('SBI_ResidencyRoad', 'Bangalore', 10000)
('SBI_ShivajiRoad', 'Bombay', 20000)
('SBI_ParlimentRoad', 'Delhi', 10000)
('SBI_Jantarmantar', 'Delhi', 20000);
select * from branch;
```

Result Grid | Filter Rows:

Branch_name	Branch_city	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

```
INSERT INTO BankAccount VALUES (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);
select * from BankAccount;
```

Result Grid | Filter Rows:

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

```
INSERT INTO BankCustomer VALUES ('Avinash', 'Bull_Temple_Road', 'Bangalore')
('Dinesh', 'Bannergatta_Road', 'Bangalore')
('Mohan', 'NationalCollege_Road', 'Bangalore')
('Nikil', 'Akbar_Road', 'Delhi')
('Ravi', 'Prithviraj_Road', 'Delhi');
select * from BankCustomer;
```

Result Grid | Filter Rows: | Edit:

Customername	Customer_street	CustomerCity
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

```
61 •  INSERT INTO Depositer VALUES ('Avinash', 1)
62 •  ('Dinesh', 2)
63 •  ('Nikil', 4)
64 •  ('Ravi', 5)
65 •  ('Avinash', 8)
66 •  ('Nikil', 9)
67 •  ('Dinesh', 10)
68 •  ('Nikil', 11);
69 •  select * from Depositer;
```

Result Grid | Filter Rows:

Customername	Accno
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11
NULL	NULL

```
70 •  INSERT INTO Loan VALUES (1, 'SBI_Chamrajpet', 1000)
71 •  (2, 'SBI_ResidencyRoad', 2000)
72 •  (3, 'SBI_ShivajiRoad', 3000)
73 •  (4, 'SBI_ParlimentRoad', 4000)
74 •  (5, 'SBI_Jantarmantar', 5000);

76 •  select * from Loan;
```

Result Grid | Filter Rows:

Loan_number	Branch_name	Amount
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_ParlimentRoad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

Queries

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

```
86 •  select Branch_name, CONCAT(assets/100000,'lakhs')assets_in_lakhs from branch;
```

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

Branch_name	assets_in_lakhs
SBI_Chamrajpet	0.5000lakhs
SBI_Jantarmantar	0.2000lakhs
SBI_ParlimentRoad	0.1000lakhs
SBI_ResidencyRoad	0.1000lakhs
SBI_ShivajiRoad	0.2000lakhs

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

```
88 •  select d.Customername from Depositer d, BankAccount b
89    where b.Branch_name='SBI_ResidencyRoad' and
90    d.Accno=b.Accno group by d.Customername having count(d.Accno)>=2;
```

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

Customername
Dinesh

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

```
92 •  create view sum_of_loan
93    as select Branch_name, SUM(Balance)
94    from BankAccount
95    group by Branch_name;
96 •  select * from sum_of_loan;
97
```

Result Grid | Filter Rows: | Expo

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

Week 4: More Queries on Bank Database

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

```
1
2 •  Select D.customername
3    From depositer D, bankaccount BA, branch B
4    where D.accno=BA.accno and BA.branch_name= B.branch_name and B.branch_city='Delhi'
5    group by D.customername
6    Having count(distinct(B.branch_name)) =
7    (select count(branch_name) from branch where branch_city = 'Delhi');
8
```

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

customername
Nikil

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

```
18 •  select customername from borrower
19    where customername not in (select customername from depositer);
20
```

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

customername
Mohan

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

```
21 •  select branch_name from branch
22    where branch_city='Bangalore'
23    and assets = (select max(assets) from branch where branch_city='Bangalore');
24
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

branch_name
SBI_Chamrajpet
NULL

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

```
25 •  select customername from borrower b, loan l
26    where b.loannumber=l.loannumber and
27    l.branch_name in
28    (select branch_name from depositer,bankaccount ba where depositer.accno=ba.branch_name in
29    (select branch_name from branch where branch.branch_city="Bangalore"));
```

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

customername
Avinash
Dinesh

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

```
31 •  delete from bankaccount where branch_name in (select branch_name from branch where branch_city='Bombay');
32 •  select * from bankaccount;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

6. Update the Balance of all accounts by 5%

```
3 •  update branch
4    set assets=assets+(assets*0.5);
5 •  select * from branch;
6
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wra

branch_name	branch_city	assets
SBI_Chamrajpet	Bangalore	112500
SBI_Jantarmantar	Delhi	45000
SBI_ParlimentRoad	Delhi	22500
SBI_ResidencyRoad	Bangalore	22500
SBI_ShivajiRoad	Bombay	45000
NULL	NULL	NULL

Week 5: Employee Database

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

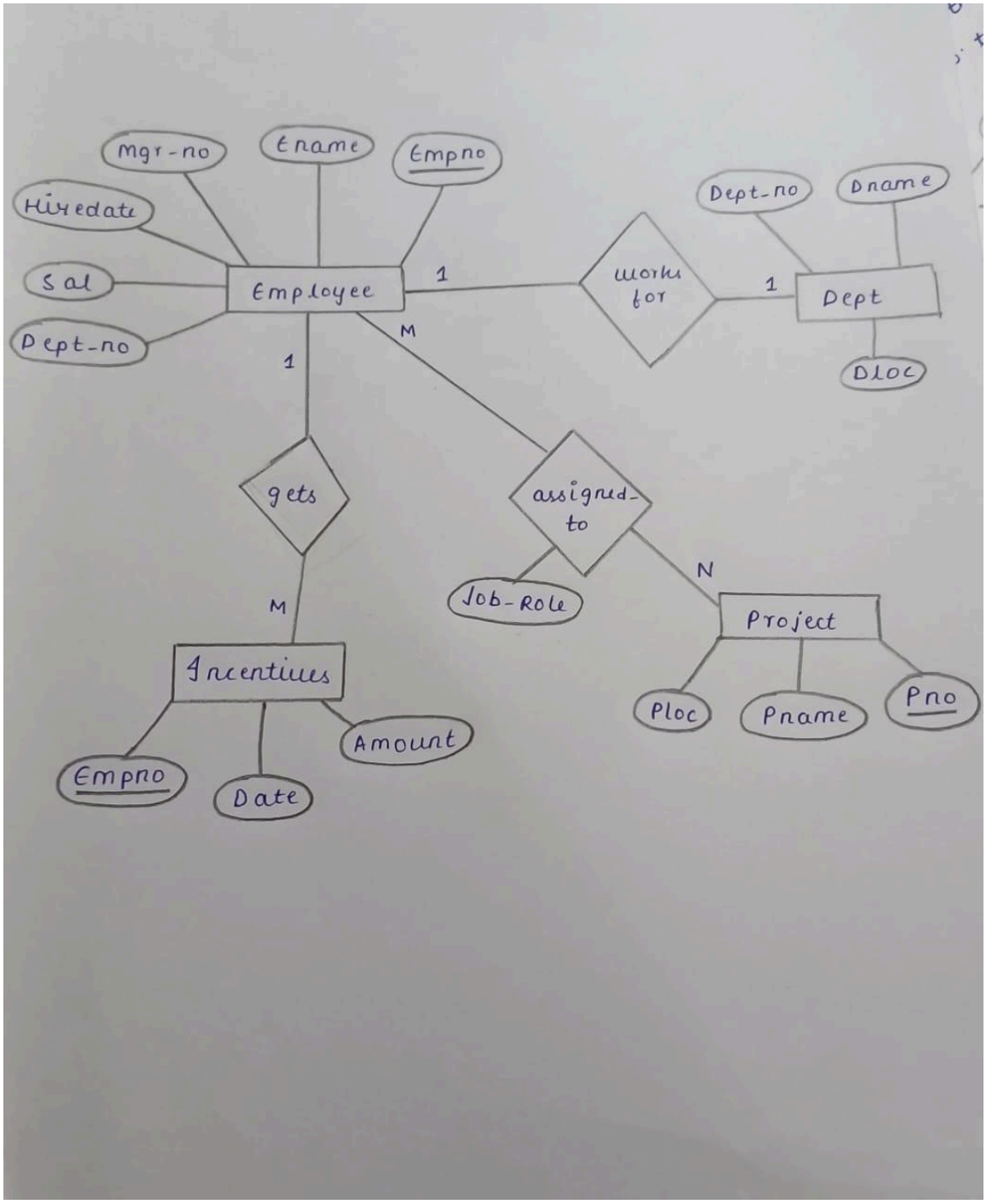
2. Enter greater than five tuples for each table.

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

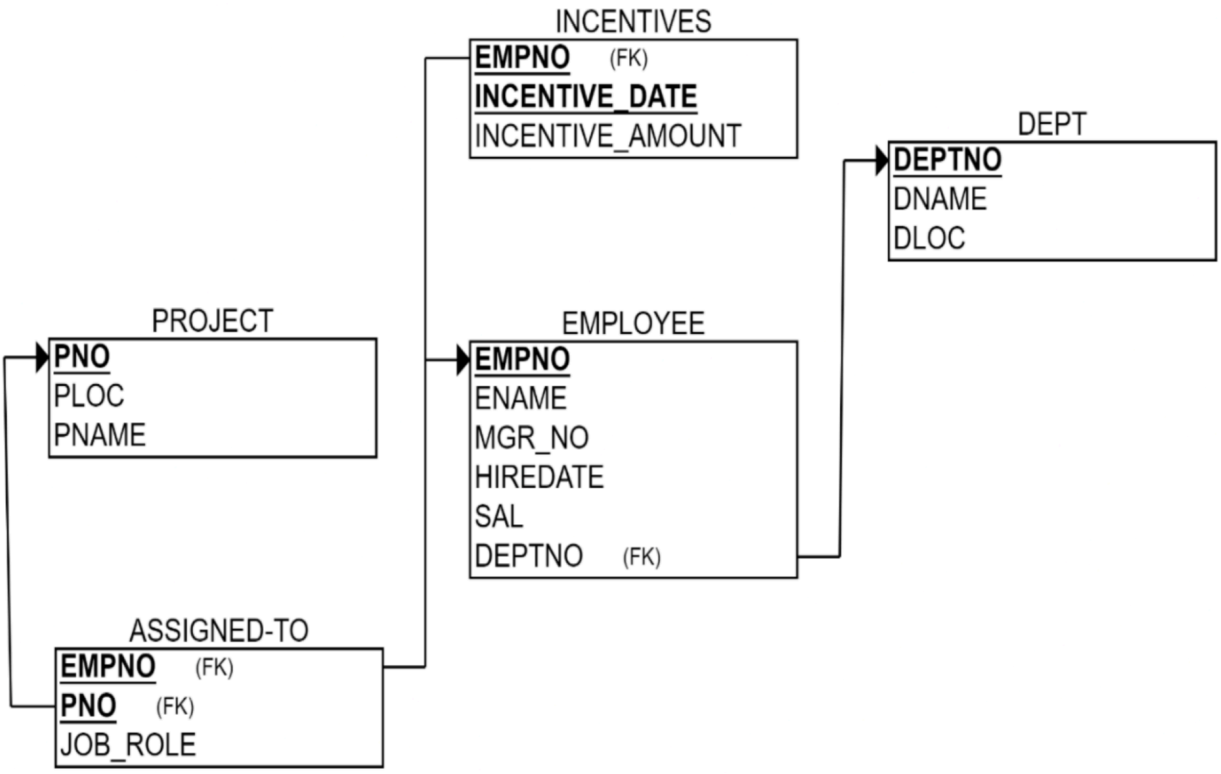
4. Get Employee ID's of those employees who didn't receive incentives

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

Entity Relationship Diagram



Schema Diagram



Create Database

```
1  create database employee;
2  use employee;
```

Create Table

```
3  create table dept(
4  deptno int PRIMARY KEY,
5  dname varchar(20),
6  dloc varchar(20));
7
8  desc dept;
```

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

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

```
10 create table employee(
11 empno int PRIMARY KEY,
12 ename varchar(40),
13 mgr_no int,
14 hiredate date,
15 sal decimal(7,2),
16 deptno int,
17 FOREIGN KEY (deptno) references dept (deptno)
18 );
19
20 desc employee;
```

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

Field	Type	Null	Key	Default	Extra
empno	int	NO	PRI	NULL	
ename	varchar(40)	YES		NULL	
mgr_no	int	YES		NULL	
hiredate	date	YES		NULL	
sal	decimal(7,2)	YES		NULL	
deptno	int	YES	MUL	NULL	

```
31 • create table project(
32     pno int PRIMARY KEY,
33     pname varchar(40),
34     ploc varchar(20));
35
36 •  desc project;
```

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

Field	Type	Null	Key	Default	Extra
pno	int	NO	PRI	NULL	
pname	varchar(40)	YES		NULL	
ploc	varchar(20)	YES		NULL	

```
38 • create table assigned_to(
39     empno int,
40     pno int,
41     job_role varchar(40),
42     FOREIGN KEY (empno) references employee (empno),
43     FOREIGN KEY (pno) references project (pno));
44
45 •  desc assigned_to;
```

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

Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
pno	int	YES	MUL	NULL	
job_role	varchar(40)	YES		NULL	

```
22 • create table incentives(
23     empno int,
24     incentive_date date PRIMARY KEY,
25     incentive_amount decimal(7,5),
26     FOREIGN KEY (empno) references employee (empno)
27     );
28
29 •  desc incentives;
```

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

Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
incentive_date	date	NO	PRI	NULL	
incentive_amount	decimal(7,5)	YES		NULL	

Inserting Values to the table

```
54 • INSERT INTO employee VALUES (7369,'Adarsh',7902,'2012-12-17','80000.00',20),
55   (7499,'Shruthi',7698,'2013-02-20','16000.00',30),
56   (7521,'Anvitha',7698,'2015-02-22','12500.00',30),
57   (7566,'Tanvir',7839,'2008-04-02','29750.00',20),
58   (7654,'Ramesh',7698,'2014-09-28','12500.00',30),
59   (7698,'Kumar',7839,'2015-05-01','28500.00',30),
60   (7782,'CLARK',7839,'2017-06-09','24500.00',10),
61   (7788,'SCOTT',7566,'2010-12-09','30000.00',20),
62   (7839,'KING',NULL,'2009-11-17','500000.00',10),
63   (7844,'TURNER',7698,'2010-09-08','15000.00',30),
64   (7876,'ADAMS',7788,'2013-01-12','11000.00',20),
65   (7900,'JAMES',7698,'2017-12-03','9500.00',30),
66   (7902,'FORD',7566,'2010-12-03','30000.00',20);
67
68 • select * from employee;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	ename	mgr_no	hiredate	sal	deptno
7369	Adarsh	7902	2012-12-17	80000.00	20
7499	Shruthi	7698	2013-02-20	16000.00	30
7521	Anvitha	7698	2015-02-22	12500.00	30
7566	Tanvir	7839	2008-04-02	29750.00	20
7654	Ramesh	7698	2014-09-28	12500.00	30
7698	Kumar	7839	2015-05-01	28500.00	30
7782	CLARK	7839	2017-06-09	24500.00	10
7788	SCOTT	7566	2010-12-09	30000.00	20
7839	KING	NULL	2009-11-17	500000.00	10
7844	TURNER	7698	2010-09-08	15000.00	30
7876	ADAMS	7788	2013-01-12	11000.00	20
7900	JAMES	7698	2017-12-03	9500.00	30
7902	FORD	7566	2010-12-03	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

```
47 • INSERT INTO dept VALUES (10,'ACCOUNTING','MUMBAI'),
48   (20,'RESEARCH','BENGALURU'),
49   (30,'SALES','DELHI'),
50   (40,'OPERATIONS','CHENNAI');
51
52 • select * from dept;
53
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

deptno	dname	dloc
10	ACCOUNTING	MUMBAI
20	RESEARCH	BENGALURU
30	SALES	DELHI
40	OPERATIONS	CHENNAI
NULL	NULL	NULL

```
82 •  INSERT INTO project VALUES(101,'AI Project','BENGALURU'),
83    (102,'IOT','HYDERABAD'),
84    (103,'BLOCKCHAIN','BENGALURU'),
85    (104,'DATA SCIENCE','MYSURU'),
86    (105,'AUTONOMUS SYSTEMS','PUNE');
87
88 •  select * from project;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

```
70 •  INSERT INTO incentives VALUES(7499,'2019-02-01',5000.00),
71    (7521,'2019-03-01',2500.00),
72    (7566,'2022-02-01',5070.00),
73    (7654,'2020-02-01',2000.00),
74    (7654,'2022-04-01',879.00),
75    (7521,'2018-02-01',8000.00),
76    (7698,'2018-03-01',500.00),
77    (7698,'2020-03-01',9000.00),
78    (7698,'2022-04-02',4500.00);
79
80 •  select * from incentives;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	incentive_date	incentive_amount
7521	2018-02-01	8000.00
7698	2018-03-01	500.00
7499	2019-02-01	5000.00
7521	2019-03-01	2500.00
7654	2020-02-01	2000.00
7698	2020-03-01	9000.00
7566	2022-02-01	5070.00
7654	2022-04-01	879.00
7698	2022-04-02	4500.00
NULL	NULL	NULL

```
90 •  INSERT INTO assigned_to VALUES(7499,101,'Software Engineer'),
91    (7521,101,'Software Architect'),
92    (7566,101,'Project Manager'),
93    (7654,102,'Sales'),
94    (7521,102,'Software Engineer'),
95    (7499,102,'Software Engineer'),
96    (7654,103,'Cyber Security'),
97    (7698,104,'Software Engineer'),
98    (7900,105,'Software Engineer'),
99    (7839,104,'General Manager');
100
101 •  select * from assigned_to
```

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

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

Queries:

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

```
103 •  select e.empno from employee e,assigned_to a,project p
104    where e.empno=a.empno and a.pno=p.pno
105    and ploc in ('BENGALURU','HYDERABAD','MYSURU');
```

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

empno
7499
7521
7566
7654
7521
7499
7654
7698
7839

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

```
107 •  select empno from employee where empno not in (select empno from incentives);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno
7782
7839
7369
7788
7876
7902
7844
7900
NULL

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

```
109 •  select e.ename,e.empno,d.dname,a.job_role,d.dloc,p.ploc
110    from employee e,dept d,project p,assigned_to a
111    where p.ploc=d.dloc and e.empno=a.empno and p.pno=a.pno and d.deptno=e.deptno;
```

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

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

Week 6: More Queries on Employee Database

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

```
113 •  select m.ename, count(*) from employee e,employee m
114    where e.mgr_no = m.empno
115    group by m.ename
116    having count(*) =(select max(mycount)
117    from (SELECT COUNT(*) mycount
118    FROM employee
119    GROUP BY mgr_no) a);
120
```

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

ename	count(*)
Kumar	5

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

```
121 •  select * from employee m where m.empno in
122    (select mgr_no from employee)
123    and m.sal > (select avg(e.sal) from employee e where e.mgr_no = m.empno );
124
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	ename	mgr_no	hiredate	sal	deptno
7698	Kumar	7839	2015-05-01	28500.00	30
7839	KING	NULL	2009-11-17	500000.00	10
7788	SCOTT	7566	2010-12-09	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

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

```
125 •  select distinct m.mgr_no,m.ename from employee e,employee m
126    where e.mgr_no=m.mgr_no and e.deptno=m.deptno
127    and e.empno in
128    (select distinct m.mgr_no from employee e, employee m where e.mgr_no=m.mgr_no and e.deptno=m.deptno) and
129    e.sal<(select max(e.sal) from employee e, employee m where e.mgr_no=m.mgr_no);
```

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

mgr_no	ename
7839	Tanvir
7839	Kumar
7566	SCOTT
7566	FORD

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

```
136 •  select e.*,i.incentive_amount from employee e, incentives i where e.empno=i.empno and
137    incentive_date like '2019-01%' and incentive_amount<
138    (select max(incentive_amount) from incentives where incentive_date like '2019-01%');
```

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

empno	ename	mgr_no	hiredate	sal	deptno	incentive_amount

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

```
140 •  select * from employee e where
141    e.deptno=(select m.deptno from employee m
142    where m.empno=e.mgr_no);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

Week 7: Supplier Database

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.

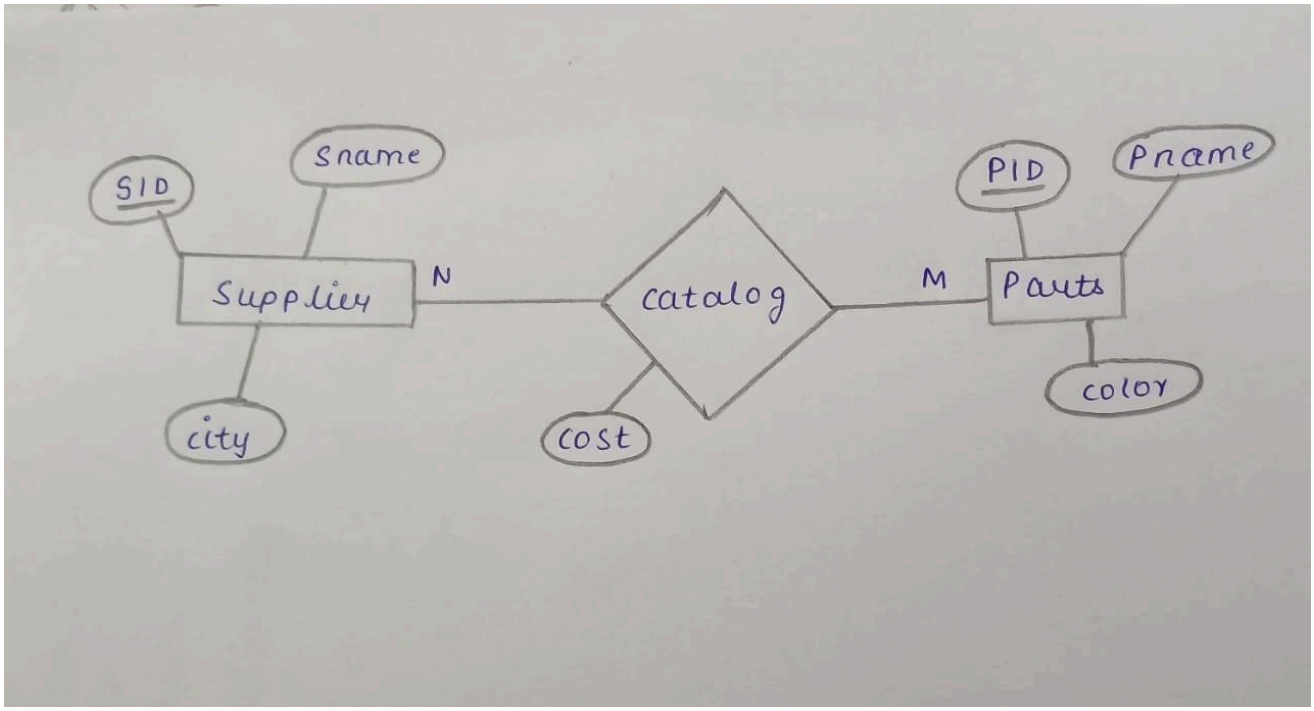
5. Find the snames of suppliers who supply every red part.

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

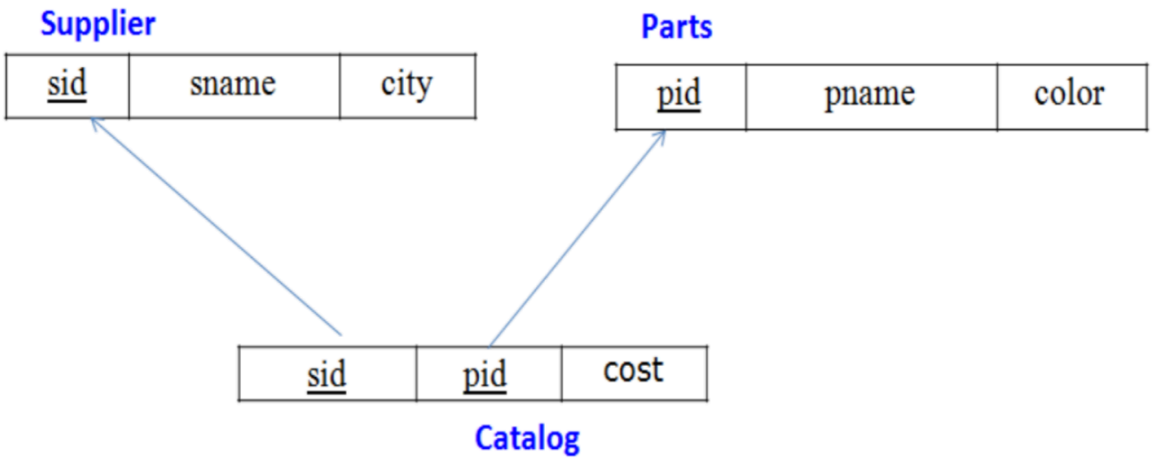
7. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

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

Entity Relationship Diagram



Schema Diagram



Create Database

```
1  create database employee;
2  use employee;
```

Create Table

```
16 create table Catalog(
17 SID int,
18 PID int,
19 Cost int,
20 FOREIGN KEY(SID) references Suppliers(SID),
21 FOREIGN KEY(PID) references Parts(PID));
22 desc catalog;
```

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

Field	Type	Null	Key	Default	Extra
SID	int	YES	MUL	NULL	
PID	int	YES	MUL	NULL	
Cost	int	YES		NULL	

```
4  create table Suppliers(
5  SID int PRIMARY KEY,
6  sname varchar(50),
7  city varchar(30));
8  desc Suppliers;
```

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

Field	Type	Null	Key	Default	Extra
SID	int	NO	PRI	NULL	
sname	varchar(50)	YES		NULL	
city	varchar(30)	YES		NULL	

```
10 create table Parts(
11 PID int PRIMARY KEY,
12 pname varchar(50),
13 color varchar(30));
14 desc Parts;
```

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

Field	Type	Null	Key	Default	Extra
PID	int	NO	PRI	NULL	
pname	varchar(50)	YES		NULL	
color	varchar(30)	YES		NULL	

Inserting Values into the table

```
31 •  insert into Parts values
32    (20001,'Book','Red'),
33    (20002,'Pen','Red'),
34    (20003,'Pencil','Green'),
35    (20004,'Mobile','Green');
36
37 •  insert into Parts values(20005,'Charger','Black');
38 •  select * from parts;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

```
24 •  insert into Suppliers values
25    (10001,'Acme Widget','Bangalore'),
26    (10002,'Johns','Kolkata'),
27    (10003,'Vimal','Mumbai'),
28    (10004,'Reliance','Delhi');
29 •  SELECT * from Suppliers;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

```
40 •  insert into Catalog values
41    (10001,20001,10),
42    (10001,20002,10),
43    (10001,20003,30),
44    (10001,20004,10),
45    (10001,20005,10),
46    (10002,20001,10),
47    (10002,20002,20),
48    (10003,20003,30),
49    (10004,20003,40);
50 •  select * from catalog;
```

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

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

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

```
52 •  select distinct p.pname from Parts p,Catalog c where p.pid=c.pid;
```

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

pname
Book
Pen
Pencil
Mobile
Charger

2. Find the snames of suppliers who supply every part.

```
54 •  select sname from suppliers where sid in (select sid from catalog group by sid having count(distinct
55    pid)=5);
```

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

sname
Acme Widget

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

```
57 •  select sname from suppliers where sid in (select sid from catalog c,parts p where c.pid=p.pid and
58    p.color='red');
```

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

sname
Acme Widget
Johns

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

```
60 •  select pname from parts where pid in (select c.pid from catalog c,suppliers s where s.sid=c.sid and
61    c.sid=10001) and pid not in (select c.pid from catalog c,suppliers s where s.sid=c.sid and
62    c.sid!=10001);
```

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

pname
Mobile
Charger

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

```
64 •  select c1.sid from catalog c1 where c1.cost > (select avg(c2.cost) from catalog c2 where
```

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

sid
10002
10004

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

```
67 •  select pid,sname from catalog c,suppliers s where c.sid=s.sid and cost = (select MAX(cost) from
68    catalog where pid=c.pid);
```

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

pid	sname
20001	Acme Widget
20004	Acme Widget
20005	Acme Widget
20001	Johns
20002	Johns
20003	Reliance

Week 8: More Queries on Supplier Database

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

```
1 •  select s.sid, s.sname, p.pid, p.pname, c.cost from catalog c
2    join supplier s on c.sid = s.sid
3    join parts p on c.pid = p.pid
4    where c.cost = (select MAX(cost) from catalog);
```

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

sid	sname	pid	pname	cost
10004	Reliance	20003	Pencil	40

2. Find suppliers who do NOT supply any red parts.

```
1 •  select s.sid, s.sname from supplier s
2    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
10003	Vimal
10004	Reliance
NULL	NULL

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

```
1 •  select s.sid, s.sname, SUM(c.cost) as total_value from supplier s
2    join catalog c on s.sid = c.sid
3    group by s.sid, s.sname;
```

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

sid	sname	total_value
10001	Acme Widget	70
10002	Johns	30
10003	Vimal	30
10004	Reliance	40

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

```
1 • select s.sid, s.sname from supplier s
2   join catalog c on s.sid = c.sid
3   where c.cost < 20 group by s.sid, s.sname having COUNT(c.pid) >= 2;
```

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

sid	sname
10001	Acme Widget

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

```
1 • select c.sid, c.pid, c.cost from catalog c
2   where c.cost = (select MIN(c2.cost) from catalog c2 where c2.pid = c.pid);
```

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

sid	pid	cost
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10001	20005	10
10002	20001	10
10003	20003	30

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

```
1  create view SupplierPartCount as
2  select s.sid, s.sname, COUNT(c.pid) as part_count from supplier s
3  left join catalog c on s.sid = c.sid
4  group by s.sid, s.sname;
5
6  select * from SupplierPartCount;
```

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

sid	sname	part_count
10001	Acme Widget	5
10002	Johns	2
10003	Vimal	1
10004	Reliance	1

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

```
1  create view MostExpensiveSupplier as
2  select c.sid, c.pid, c.cost from catalog c
3  where c.cost = (select MAX(c2.cost)from catalog c2 where c2.pid = c.pid);
4  select * from MostExpensiveSupplier;
```

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

sid	pid	cost
10001	20001	10
10001	20004	10
10001	20005	10
10002	20001	10
10002	20002	20
10004	20003	40

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

```
DELIMITER $$;
CREATE TRIGGER trg_cost_min
BEFORE INSERT ON catalog
FOR EACH ROW
BEGIN
    IF NEW.cost < 1 THEN
        SIGNAL SQLSTATE '45000'
            SET MESSAGE_TEXT = 'Cost cannot be below 1';
    END IF;
END$$;

```

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

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

Week 9: NO SQL Student Database

Create Database

```
test> db.createCollection("student");
{ ok: 1 }
```

Inserting values

```
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('6936655093569dc27263b112') }
}
test> db.student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6936656893569dc27263b113') }
}
test> db.student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6936658293569dc27263b114') }
}
test> db.student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6936659a93569dc27263b115') }
}
test> db.student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693665af93569dc27263b116') }
}
```

View values

```
test> db.student.find()
[
  {
    _id: ObjectId('6936655093569dc27263b112'),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId('6936656893569dc27263b113'),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId('6936658293569dc27263b114'),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId('6936659a93569dc27263b115'),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId('693665af93569dc27263b116'),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

<u>Queries:</u>

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

```
test> db.student.update({ RollNo: 10 }, { $set: { email: "Abhinav@gmail.com" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

```
test> db.student.insert({RollNo:11,Age:22,Name:"ABC",Cont:2276,email:"rea.de9@gmail.com"});
{ncaught:
  acknowledged: true,ed token (4:0)
  insertedIds: { '0': ObjectId('6936690093569dc27263b118') }
} 2 |
```

```
test> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
  {
    _id: ObjectId('6936690093569dc27263b118'),
    RollNo: 11,
    Age: 22,
    Name: 'FEM',
    Cont: 2276,
    email: 'rea.de9@gmail.com'
  }
]
```

3. Drop the table

```
test> db.student.drop();
true
test> db.student.find();
```

Week 10: NO SQL Customer Database

Create Database

```
test> db.createCollection("customer");
{ ok: 1 }
```

Create table

```
test> db.customer.insert({"Cust_id":1,"Acc_Bal":5000,"Acc_Type":"Savings"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faa69c75a2a4b111e2626') }
}
test> db.customer.insert({"Cust_id":2,"Acc_Bal":7500,"Acc_Type":"Savings"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faa75c75a2a4b111e2627') }
}
test> db.customer.insert({"Cust_id":3,"Acc_Bal":200,"Acc_Type":"Transactions"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faaa4c75a2a4b111e2628') }
}
test> db.customer.insert({"Cust_id":4,"Acc_Bal":10000,"Acc_Type":"Savings"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faabac75a2a4b111e2629') }
}
test> db.customer.insert({"Cust_id":5,"Acc_Bal":1000,"Acc_Type":"Transactions"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faacbc75a2a4b111e262a') }
}
```

Queries

1. Write a query to display those records whose total account balance is greater than 1200 of account type ‘Z’ for each customer_id.

```
Atlas atlas-3acvm6-shard-0 [primary] DBME_DEMO> db.Customers.find({ Acc_Bal: { $gt: 1200 }, Acc_Type:
"Z" })
[
  {
    _id: ObjectId('693f8376701108f7231e262c'),
    Cust_id: 1,
    Acc_Bal: 1500,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('693f8376701108f7231e262e'),
    Cust_id: 3,
    Acc_Bal: 2000,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('693f8376701108f7231e2630'),
    Cust_id: 5,
    Acc_Bal: 1800,
    Acc_Type: 'Z'
  }
]
```

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

```
Atlas atlas-3acvm6-shard-0 [primary] DBME_DEMO> db.Customers.aggregate([
...   {
...     $group: {
...       _id: "$Cust_id",
...       Min_Bal: { $min: "$Acc_Bal" },
...       Max_Bal: { $max: "$Acc_Bal" }
...     }
...   }
... ])
...
[
  { _id: 5, Min_Bal: 1800, Max_Bal: 1800 },
  { _id: 2, Min_Bal: 800, Max_Bal: 800 },
  { _id: 4, Min_Bal: 500, Max_Bal: 500 },
  { _id: 3, Min_Bal: 2000, Max_Bal: 2000 },
  { _id: 1, Min_Bal: 1500, Max_Bal: 1500 }
]
```

3. Export the created collection into local file system

```
C:\Users\BMSCECSE-L3-27>mongoexport --uri="mongodb+srv://annapurna_cs24_user:anusm060706@cluster0.eavk
tnp.mongodb.net/DBME_DEMO" --collection=New_Customer --type=csv --fields=Cust_id,Acc_Bal,Acc_Type --ou
t="C:\Users\BMSCECSE-L3-27\Downloads\New_Customer_Export.csv"
2025-12-15T09:35:30.774+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.eavktnp.mongodb.ne
t/DBME_DEMO
2025-12-15T09:35:30.856+0530    exported 5 records
```

4. Drop the table

```
Atlas atlas-3acvm6-shard-0 [primary] DBME_DEMO> db.Customers.drop()
true
```

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

```
C:\Users\BMSCECSE-L3-27>mongoimport --uri="mongodb+srv://annapurna_cs24_user:anusm060706@cluster0.eavk
tnp.mongodb.net/DBME_DEMO" --collection=New_Customer --type=csv --headerline --file="C:\Users\BMSCECSE
-L3-27\Downloads\New_Customer_Export.csv"
2025-12-15T09:35:58.160+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.eavktnp.mongodb.ne
t/DBME_DEMO
2025-12-15T09:35:58.234+0530    5 document(s) imported successfully. 0 document(s) failed to import.
```

Week 11: NO SQL Restaurant Database

Create Database

```
test> db.createCollection( "restaurants");
{ ok: 1 }
```

Inserting Values

```
test> 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('693fa0ac01c1a3091e1e2621'),
    '1': ObjectId('693fa0ac01c1a3091e1e2622'),
    '2': ObjectId('693fa0ac01c1a3091e1e2623'),
    '3': ObjectId('693fa0ac01c1a3091e1e2624'),
    '4': ObjectId('693fa0ac01c1a3091e1e2625')
  }
}
```

Queries:

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

```
test> db.restaurants.find({})
[
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

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

```
test> db.restaurants.find({}).sort({ name: -1 })
[
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

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

```
test> db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })
[
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
]
test>
```

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

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

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

```
test> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
[
  { name: 'Meghna Foods', address: { street: 'Jayanagar' } },
  { name: 'Empire', address: { street: 'MG Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momos', address: { street: 'Malleshwaram' } }
]
test>
```