

# **VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

"JnanaSangama", Belgaum -590014, Karnataka.



## **LAB REPORT on**

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

*Submitted by*

**PRANAV R REDDY(1BM24CS216)**

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



**B.M.S. COLLEGE OF ENGINEERING**

(Autonomous Institution under VTU)

**BENGALURU-560019**

**Sep-2025 to Jan-2026**

**B.M.S. College of Engineering,**  
**Bull Temple Road, Bangalore 560019**  
(Affiliated To Visvesvaraya Technological University, Belgaum)  
**Department of Computer Science and Engineering**



**CERTIFICATE**

This is to certify that the Lab work entitled “Database Management Systems (23CS3PCDBM)” carried out by **PRANAV R REDDY (1BM24CS216)**, who is a 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.

|  |  |
|--|--|
| Surabhi S<br>Assistant Professor<br>Department of CSE, BMSCE | Dr. Kavitha Sooda<br>Professor & HOD<br>Department of CSE, BMSCE |
|--|--|

## Index

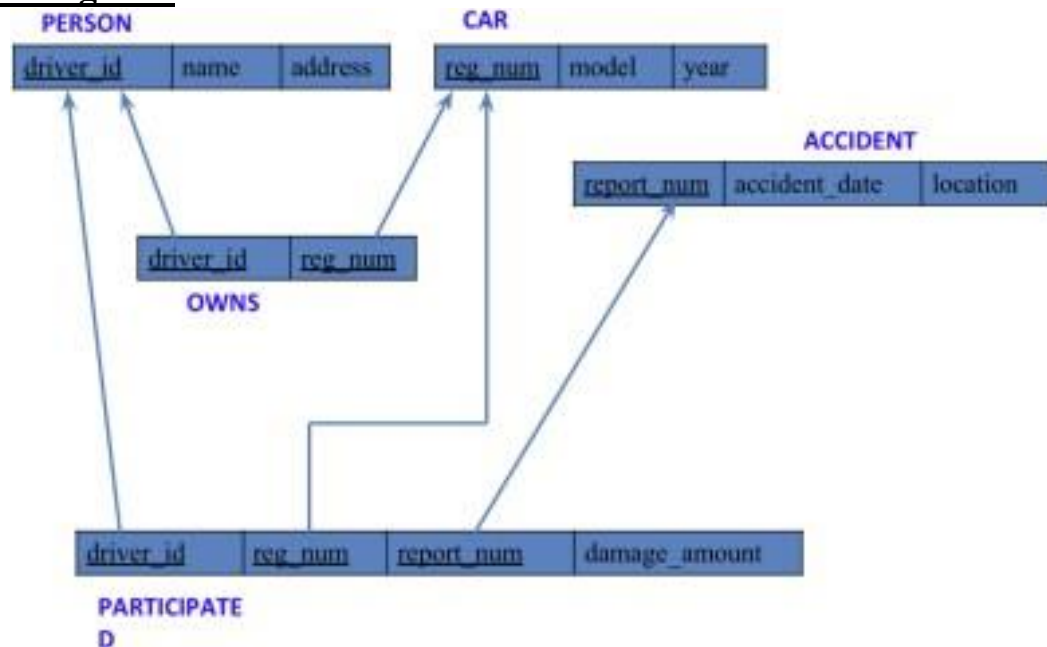
| <b>Sl. No.</b> | <b>Date</b> | <b>Experiment Title</b>            | <b>Page No.</b> |
|----------------|-------------|------------------------------------|-----------------|
| 1              | 3-10-2025   | Insurance Database                 | 4-9             |
| 2              | 10-10-2025  | More Queries on Insurance Database | 10-12           |
| 3              | 17-10-2025  | Bank Database                      | 13-18           |
| 4              | 24-10-2025  | More Queries on Bank Database      | 19-21           |
| 5              | 31-10-2025  | Employee Database                  | 22-27           |
| 6              | 7-11-2025   | More Queries on Employee Database  | 28-29           |
| 7              | 14-11-2025  | Supplier Database                  | 30-34           |
| 8              | 21-11-2025  | NO SQL - Student Database          | 35-37           |
| 9              | 05-12-2025  | NO SQL - Customer Database         | 38-40           |
| 10             | 12-12-2025  | NO SQL – Restaurant Database       | 41-45           |

**Program 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)
  - i. Create the above tables by properly specifying the primary keys and the foreign keys.
  - ii. Enter at least five tuples for each relation.
  - iii. Display Accident date and location.
  - iv. 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.
  - v. Add a new accident to the database.
  - vi. Display Accident date and location.
  - vii. 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(20) | NO   | PRI | NULL    |       |
|   | reg_num       | varchar(10) | NO   | PRI | NULL    |       |
|   | report_num    | int         | NO   | PRI | NULL    |       |
|   | damage_amount | int         | 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(50) | YES  |     | NULL    |       |

DESC participated;

|             |   |              |         |   |                    |   |
|-------------|---|--------------|---------|---|--------------------|---|
| Result Grid |  | Filter Rows: | Export: |  | Wrap Cell Content: |  |
|             | Field   | Type         | Null    | Key   | Default            | Extra   |
| ▶           | driver_id   | varchar(20)  | NO      | PRI   | NULL               |   |
|             | reg_num   | varchar(10)  | NO      | PRI   | NULL               |   |
|             | report_num  | int          | NO      | PRI   | NULL               |   |
|             | damage_amount   | int          | YES     |   | NULL               |   |

DESC car;

Result Grid

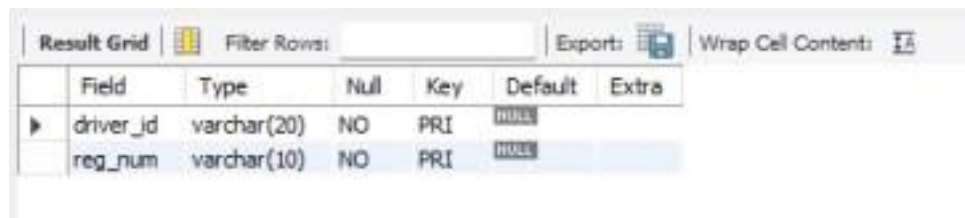
Filter Rows:

Export:

Wrap Cell Contents:

|  | Field   | Type        | Null | Key | Default | Extra |
|--|---------|-------------|------|-----|---------|-------|
|  | reg_num | varchar(15) | NO   | PRI | NULL    |       |
|  | model   | varchar(10) | YES  |     | NULL    |       |
|  | year    | int         | YES  |     | NULL    |       |

DESC owns;



|   | Field     | Type        | Null | Key | Default | Extra |
|---|-----------|-------------|------|-----|---------|-------|
| ▶ | driver_id | varchar(20) | NO   | PRI | NULL    |       |
|   | reg_num   | varchar(10) | NO   | PRI | NULL    |       |

### **Inserting Values To The Table:**

```
INSERT INTO PERSON VALUES("A01","Richard", "Srinivas nagar");
```

```
INSERT INTO PERSON VALUES("A02","Pradeep", "Rajaji nagar");
```

```
INSERT INTO PERSON VALUES("A03","Smith", "Ashok nagar");
```

```
INSERT INTO PERSON VALUES("A04","Venu", "N R Colony");
```

```
INSERT INTO PERSON VALUES("A05","John", "Hanumanth nagar");
```

```
SELECT * FROM PERSON;
```



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

```
INSERT INTO CAR VALUES("KA052250","Indica", "1990");
```

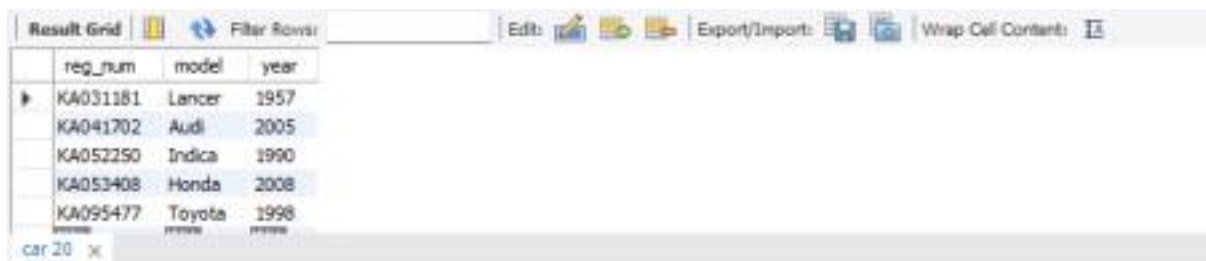
```
INSERT INTO CAR VALUES("KA031181","Lancer", "1957");
```

```
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 |

```

INSERT INTO OWNS VALUES("A01","KA052250");
INSERT INTO OWNS VALUES("A02","KA031181");
INSERT INTO OWNS VALUES("A03","KA095477");
INSERT INTO OWNS VALUES("A04","KA053408");
INSERT INTO OWNS VALUES("A05","KA041702");
SELECT * FROM OWNS;

```

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

```

INSERT INTO ACCIDENT VALUES(11,'2003-01-01',"Mysore Road");
INSERT INTO ACCIDENT VALUES(12,'2004-02-02',"South end Circle");
INSERT INTO ACCIDENT VALUES (13,'2003-01-21',"Bull temple Road");
INSERT INTO ACCIDENT VALUES (14,'2008-02-17',"Mysore Road");
INSERT INTO ACCIDENT VALUES (15,'2004-03-05',"Kanakpura Road");
SELECT * FROM ACCIDENT;

```

| report_num | accident_date | location         |
|------------|---------------|------------------|
| 11         | 2003-01-01    | Mysore Road      |
| 12         | 2004-02-02    | South end Circle |
| 13         | 2003-01-21    | Bull temple Road |
| 14         | 2008-02-17    | Mysore Road      |
| 15         | 2004-03-05    | Kanakpura Road   |

```

INSERT INTO PARTICIPATED VALUES("A01","KA052250",11,10000);
INSERT INTO PARTICIPATED VALUES ("A02","KA053408",12,50000);
INSERT INTO PARTICIPATED VALUES ("A03","KA095477",13,25000);
INSERT INTO PARTICIPATED VALUES ("A04","KA031181",14,3000);
INSERT INTO PARTICIPATED VALUES ("A05","KA041702",15,5000);
SELECT * FROM PARTICIPATED;

```

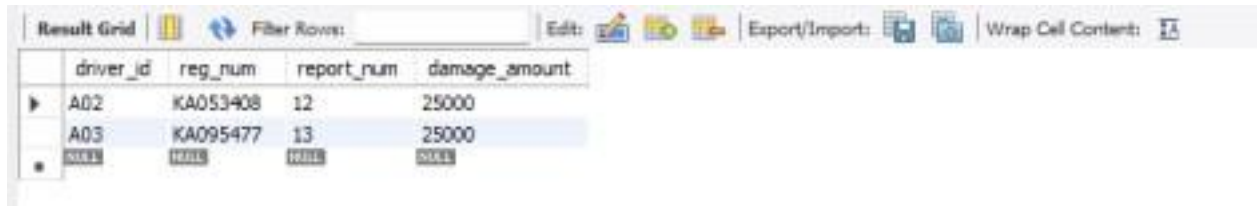
| driver_id | reg_num  | report_num | damage_amount |
|-----------|----------|------------|---------------|
| A01       | KA052250 | 11         | 10000         |
| A02       | KA053408 | 12         | 25000         |
| A03       | KA095477 | 13         | 25000         |
| A04       | KA031181 | 14   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



## Queries:

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

UPDATE PARTICIPATED SET damage\_amount = 25000 WHERE reg\_num = 'KA053408' AND report\_num = 12;

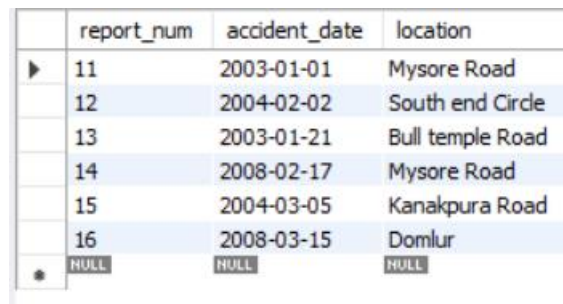


|   | driver_id | reg_num  | report_num | damage_amount |
|---|-----------|----------|------------|---------------|
| ▶ | A02       | KA053408 | 12         | 25000         |
|   | A03       | KA095477 | 13         | 25000         |
| • | NULL      | NULL     | NULL       | NULL          |

- Add a new accident to the database.

INSERT INTO ACCIDENT VALUES(16,'2008-03-15','Domlur');

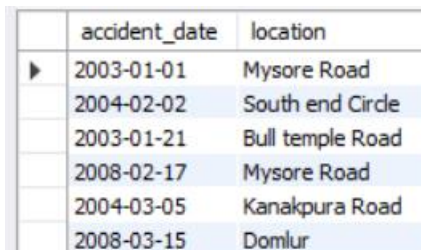
SELECT \* FROM ACCIDENT;



|   | report_num | accident_date | location         |
|---|------------|---------------|------------------|
| ▶ | 11         | 2003-01-01    | Mysore Road      |
|   | 12         | 2004-02-02    | South end Circle |
|   | 13         | 2003-01-21    | Bull temple Road |
|   | 14         | 2008-02-17    | Mysore Road      |
|   | 15         | 2004-03-05    | Kanakpura Road   |
|   | 16         | 2008-03-15    | Domlur           |
| • | NULL       | NULL          | NULL             |

- Display Accident date and location.

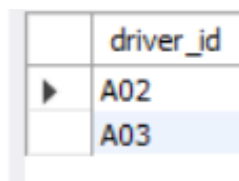
SELECT accident\_date, location FROM ACCIDENT;



|   | accident_date | location         |
|---|---------------|------------------|
| ▶ | 2003-01-01    | Mysore Road      |
|   | 2004-02-02    | South end Circle |
|   | 2003-01-21    | Bull temple Road |
|   | 2008-02-17    | Mysore Road      |
|   | 2004-03-05    | Kanakpura Road   |
|   | 2008-03-15    | 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       |

## **Program 2**

### **More Queries On 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 as done in  
“Program-1”week’s lab and Enter at least five tuples for each relation.

- i. Display the entire CAR relation in the ascending order of manufacturing year.
- ii. Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
- iii. Find the total number of people who owned cars that involved in accidents in 2008.
- iv. List the entire participated relation in the Descending Order of Damage Amount.
- v. Find the Average Damage Amount.
- vi. Delete the tuple whose Damage Amount is below the Average Damage Amount.
- vii. List the name of drivers whose Damage is Greater than the Average Damage Amount.
- viii. Find Maximum Damage Amount.

### Queries:

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

```
SELECT * FROM CAR ORDER BY year ASC;
```

|   | reg_num  | model  | year |
|---|----------|--------|------|
| ▶ | KA03181  | 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 C.reg_num =  
P.reg_num AND C.model = 'Lancer';
```

|   | count(report_num) |
|---|-------------------|
| ▶ | 1                 |

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

```
SELECT COUNT(DISTINCT driver_id) AS CNT FROM ACCIDENT A, PARTICIPATED  
P WHERE P.report_num = A.report_num AND A.accident_date LIKE '__08%';
```

|   | CNT |
|---|-----|
| ▶ | 1   |

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

```
SELECT * FROM PARTICIPATED ORDER BY damage_amount DESC;
```

|   | driver_id | reg_num  | report_num | damage_amount |
|---|-----------|----------|------------|---------------|
| ▶ | A02       | KA053408 | 12         | 40000         |
|   | A03       | KA095477 | 13         | 25000         |
|   | A01       | KA052250 | 11         | 10000         |
|   | A05       | KA041702 | 15         | 5000          |
|   | A04       | KA03181  | 14         | 3000          |
| * | NULL      | NULL     | NULL       | NULL          |

- **Find the Average Damage Amount.**

```
SELECT AVG(damage_amount) AS avg_damage
FROM PARTICIPATED;
```

|   |            |
|---|------------|
|   | avg_damage |
| ▶ | 16600.0000 |

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

```
SELECT P.name FROM PERSON P
JOIN PARTICIPATED PA ON P.driver_id = PA.driver_id
WHERE PA.damage_amount > (
    SELECT AVG(damage_amount)
    FROM PARTICIPATED
);
```

|   |         |
|---|---------|
|   | name    |
| ▶ | Pradeeo |
|   | Smith   |

- **Find Maximum Damage Amount.**

```
SELECT MAX(damage_amount) AS max_damage FROM PARTICIPATED;
```

|   |            |
|---|------------|
|   | max_damage |
| ▶ | 40000      |

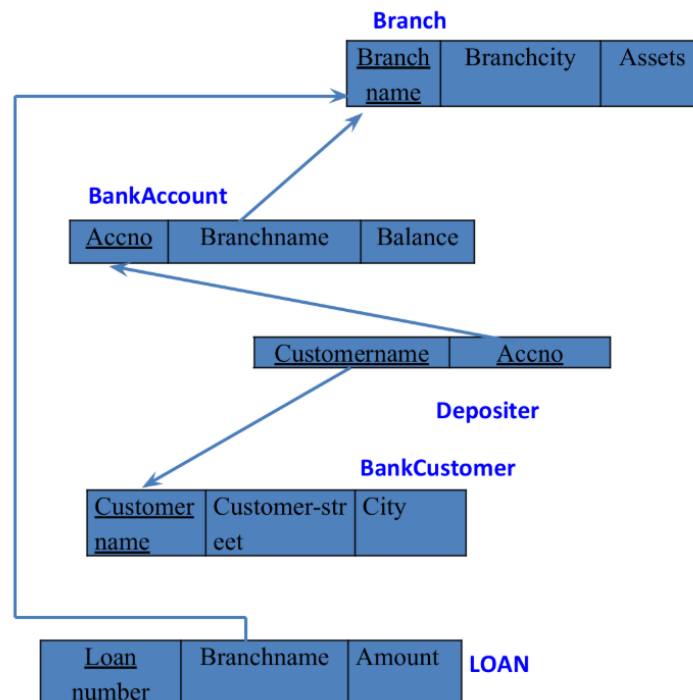
### Program 3

## 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)
- i. Create the above tables by properly specifying the primary keys and the foreign keys.
  - ii. Enter at least five tuples for each relation.
  - iii. Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
  - iv. Find all the customers who have at least two accounts at the same branch (ex. SBI\_ResidencyRoad).
  - v. 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 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));
```

## 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 |
|---|-----------------|-------------|------|-----|---------|-------|
| ▶ | Custormername   | varchar(20) | NO   | PRI | NULL    |       |
|   | Customer_street | varchar(30) | YES  |     | NULL    |       |
|   | CustomerCity    | varchar(35) | YES  |     | NULL    |       |

DESC DEPOSITER;

|   | Field         | Type        | Null | Key | Default | Extra |
|---|---------------|-------------|------|-----|---------|-------|
| ▶ | Custormername | varchar(20) | NO   | PRI | NULL    |       |
|   | Accno         | int         | NO   | PRI | NULL    |       |

DESC LOAN;

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

### **Inserting Values To The Table:**

```
INSERT INTO BRANCH VALUES("SBI_Chamrajpet","Banglore",50000);
INSERT INTO BRANCH VALUES("SBI_ResidencyRoad","Banglore",10000);
INSERT INTO BRANCH VALUES("SBI_ShivajiRoad","Banglore",20000);
INSERT INTO BRANCH VALUES("SBI_Parliament","Banglore",10000);
INSERT INTO BRANCH VALUES("SBI_Jantarmentar","Banglore",20000);
SELECT * FROM BRANCH;
```

|   | Branch_name       | Branch_city | assets |
|---|-------------------|-------------|--------|
| ▶ | SBI_Chamrajpet    | Banglore    | 50000  |
|   | SBI_Jantarmentar  | Banglore    | 20000  |
|   | SBI_Parliament    | Banglore    | 10000  |
|   | SBI_ResidencyRoad | Banglore    | 10000  |
|   | SBI_ShivajiRoad   | Banglore    | 20000  |
| * | NULL              | NULL        | NULL   |

```

INSERT INTO BANKACCOUNT VALUES(1,"SBI_Chamrajpet",2000);
INSERT INTO BANKACCOUNT VALUES(2,"SBI_ResidencyRoad",5000);
INSERT INTO BANKACCOUNT VALUES(3,"SBI_ShivajiRoad",6000);
INSERT INTO BANKACCOUNT VALUES(4,"SBI_Parliament",9000);
INSERT INTO BANKACCOUNT VALUES(5,"SBI_Jantarmanatar",8000);
INSERT INTO BANKACCOUNT VALUES(6,"SBI_ShivajiRoad",4000);
INSERT INTO BANKACCOUNT VALUES(8,"SBI_ResidencyRoad",4000);
INSERT INTO BANKACCOUNT VALUES(9,"SBI_Parliament",3000);
INSERT INTO BANKACCOUNT VALUES(10,"SBI_ResidencyRoad",5000);
INSERT INTO BANKACCOUNT VALUES(11,"SBI_Jantarmanatar",2000);
SELECT * FROM BANKACCOUNT;

```

|   | Accno | Branch_name       | Balance |
|---|-------|-------------------|---------|
| ▶ | 1     | SBI_Chamrajpet    | 2000    |
|   | 2     | SBI_ResidencyRoad | 5000    |
|   | 3     | SBI_ShivajiRoad   | 6000    |
|   | 4     | SBI_Parliament    | 9000    |
|   | 5     | SBI_Jantarmanatar | 8000    |
|   | 6     | SBI_ShivajiRoad   | 4000    |
|   | 8     | SBI_ResidencyRoad | 4000    |
|   | 9     | SBI_Parliament    | 3000    |
|   | 10    | SBI_ResidencyRoad | 5000    |
|   | 11    | SBI_Jantarmanatar | 2000    |
| • | NULL  | NULL              | NULL    |

```

INSERT INTO BANKCUSTOMER VALUES("Avinash","Bull_Temple_Road","Bangalore");
INSERT INTO BANKCUSTOMER VALUES("Dinesh","Bannerghatta_Road","Bangalore");
INSERT INTO BANKCUSTOMER VALUES("Mohan","NationalCollege_Road","Bangalore");
INSERT INTO BANKCUSTOMER VALUES("Nikil","Akbar_Road","Delhi");
INSERT INTO BANKCUSTOMER VALUES("Ravi","Prithviraj_Road","Delhi");
SELECT * FROM BANKCUSTOMER;

```

|   | Customername | Customer_street      | CustomerCity |
|---|--------------|----------------------|--------------|
| ▶ | Avinash      | Bull_Temple_Road     | Bangalore    |
|   | Dinesh       | Bannerghatta_Road    | Bangalore    |
|   | Mohan        | NationalCollege_Road | Bangalore    |
|   | Nikil        | Akbar_Road           | Delhi        |
|   | Ravi         | Prithviraj_Road      | Delhi        |
| • | NULL         | NULL                 | NULL         |



```

INSERT INTO DEPOSITER VALUES("Avinash",1);
INSERT INTO DEPOSITER VALUES("Dinesh",2);
INSERT INTO DEPOSITER VALUES("Nikil",4);
INSERT INTO DEPOSITER VALUES("Ravi",5);
INSERT INTO DEPOSITER VALUES("Avinash",8);
INSERT INTO DEPOSITER VALUES("Nikil",9);
INSERT INTO DEPOSITER VALUES("Dinesh",10);
INSERT INTO DEPOSITER VALUES("Nikil",11);
SELECT * FROM DEPOSITER;

```

|   | Customername | Accno |
|---|--------------|-------|
| ▶ | Avinash      | 1     |
|   | Dinesh       | 2     |
|   | Nikil        | 4     |
|   | Ravi         | 5     |
|   | Avinash      | 8     |
|   | Nikil        | 9     |
|   | Dinesh       | 10    |
|   | Nikil        | 11    |
| • | NULL         | NULL  |

```

INSERT INTO Loan VALUES(1,"SBI_Chamrajpet",1000);
INSERT INTO Loan VALUES(2,"SBI_ResidencyRoad",2000);
INSERT INTO Loan VALUES(3,"SBI_ShivajiRoad",3000);
INSERT INTO Loan VALUES(4,"SBI_Parliament",4000);
INSERT INTO Loan VALUES(5,"SBI_Jantarantar",5000);
SELECT * FROM Loan;

```

|   | Loan_number | Branch_name       | Amount |
|---|-------------|-------------------|--------|
| ▶ | 1           | SBI_Chamrajpet    | 1000   |
|   | 2           | SBI_ResidencyRoad | 2000   |
|   | 3           | SBI_ShivajiRoad   | 3000   |
|   | 4           | SBI_Parliament    | 4000   |
|   | 5           | SBI_Jantarantar   | 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') AS assets_in_lakhs FROM  
BRANCH;
```

|   | Branch_name       | assets_in_lakhs |
|---|-------------------|-----------------|
| ▶ | SBI_Chamrajpet    | 0.5000lakhs     |
|   | SBI_Jantarmanatar | 0.2000lakhs     |
|   | SBI_Parliament    | 0.1000lakhs     |
|   | SBI_ResidencyRoad | 0.1000lakhs     |
|   | SBI_ShivajiRoad   | 0.2000lakhs     |

- 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='SBI_ResidencyRoad' AND d.Accno=b.Accno GROUP BY d.Customername  
HAVING COUNT(d.Accno)>=2;
```

|   | Customername |
|---|--------------|
| ▶ | Dinesh       |

- 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) AS total_balance FROM  
BANKACCOUNT GROUP BY Branch_name;
```

```
SELECT * FROM sum_of_loan;
```

|   | Branch_name       | SUM(Balance) |
|---|-------------------|--------------|
| ▶ | SBI_Chamrajpet    | 2000         |
|   | SBI_Jantarmanatar | 10000        |
|   | SBI_Parliament    | 12000        |
|   | SBI_ResidencyRoad | 14000        |
|   | SBI_ShivajiRoad   | 10000        |

## Program 4

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

#### Queries:

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

```
SELECT d.customer_name FROM DEPOSITER d WHERE NOT EXISTS (SELECT  
b.branch_name FROM BRANCH b WHERE b.branch_city='Delhi' EXCEPT SELECT  
ba.branch_name FROM BANKACCOUNT ba WHERE ba.accno=d.accno);
```

|   | CUSTOMERNAME |
|---|--------------|
| ► | Avinash      |
|   | Dinesh       |
|   | Nikil        |
|   | Ravi         |
|   | Avinash      |
|   | Nikil        |
|   | Dinesh       |
|   | Nikil        |

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

```
SELECT DISTINCT c.customer_name FROM BANKCUSTOMER c, LOAN l WHERE
c.customer_name=l.loan_number AND c.customer_name NOT IN (SELECT
d.customer_name FROM DEPOSITER d);
```

|              |
|--------------|
| customername |
|--------------|

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

```
SELECT DISTINCT d.customer_name FROM DEPOSITER d, BANKACCOUNT ba, LOAN
l WHERE d.accno=ba.accno AND ba.branch_name=l.branch_name AND
ba.branch_name='Bangalore';
```

|              |
|--------------|
| customername |
|--------------|

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

```
SELECT branch_name FROM BRANCH WHERE assets > ALL (SELECT assets FROM
BRANCH WHERE branch_city='Bangalore');
```

|                   |
|-------------------|
| branch_name       |
| ▶ SBI_Chamrajpet  |
| SBI_Jantarmanatar |
| SBI_Parliament    |
| SBI_ResidencyRoad |
| SBI_ShivajiRoad   |

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

```
DELETE FROM BANKACCOUNT WHERE branch_name IN (SELECT branch_name
FROM BRANCH WHERE branch_city='Bombay');
```

```
SELECT * FROM BANKACCOUNT;
```

|   | Accno | Branch_name       | Balance |
|---|-------|-------------------|---------|
| ▶ | 1     | SBI_Chamrajpet    | 2000    |
|   | 2     | SBI_ResidencyRoad | 5000    |
|   | 3     | SBI_ShivajiRoad   | 6000    |
|   | 4     | SBI_Parliament    | 9000    |
|   | 5     | SBI_Jantarmanatar | 8000    |
|   | 6     | SBI_ShivajiRoad   | 4000    |
|   | 8     | SBI_ResidencyRoad | 4000    |
|   | 9     | SBI_Parliament    | 3000    |
|   | 10    | SBI_ResidencyRoad | 5000    |
|   | 11    | SBI_Jantarmanatar | 2000    |
| • | NULL  | NULL              | NULL    |

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

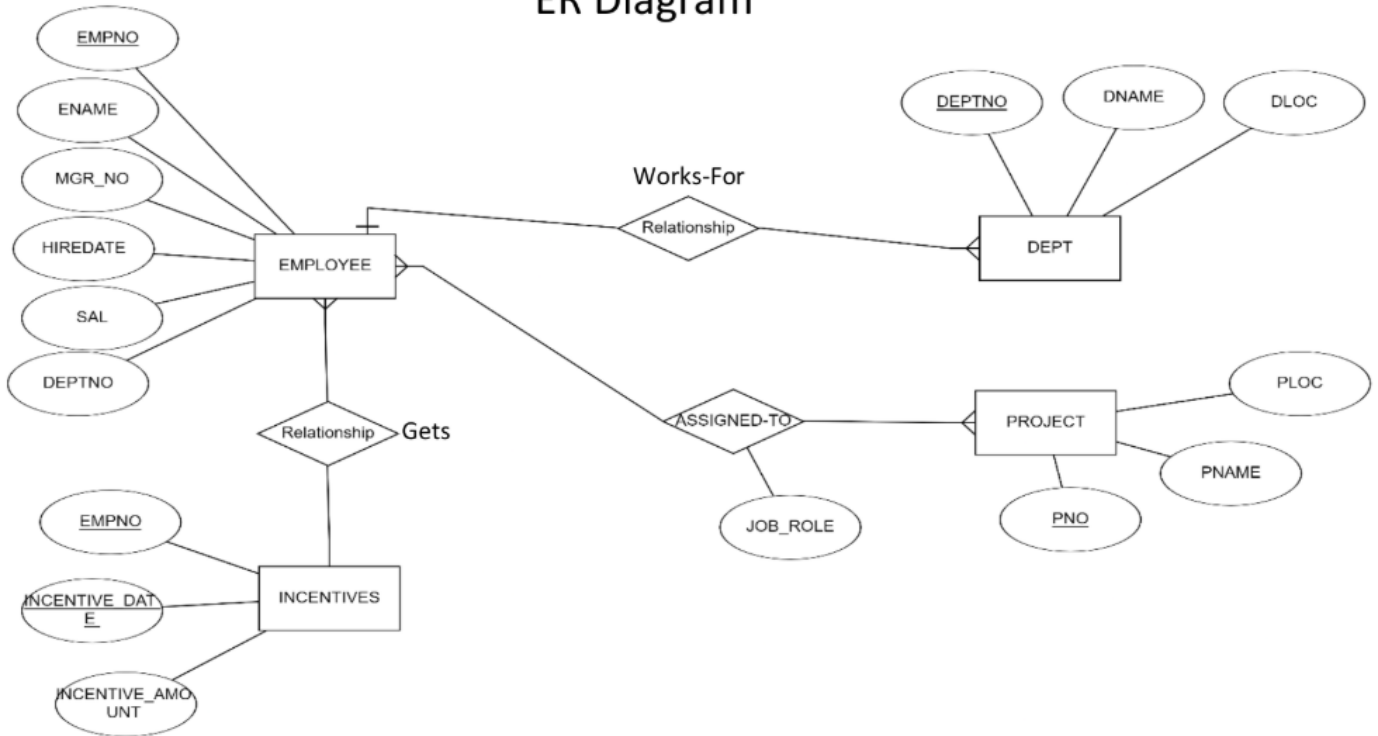
UPDATE BANKACCOUNT SET balance=balance\*1.05;

SELECT \* FROM BANKACCOUNT;

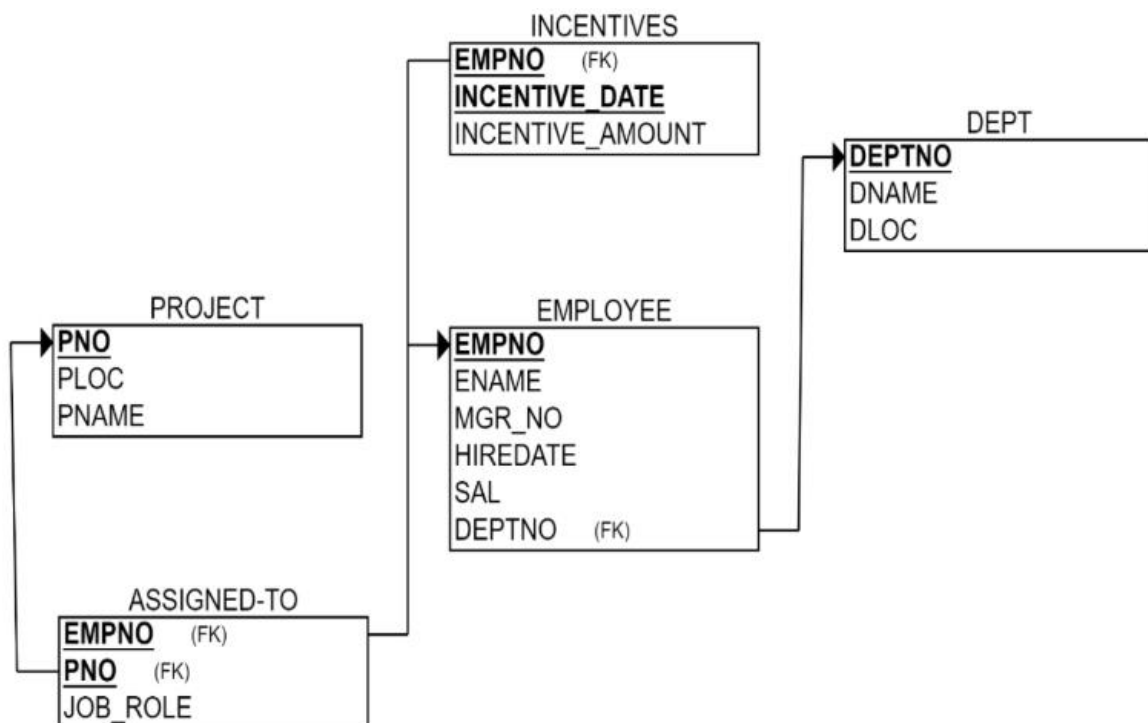
|   | Accno | Branch_name       | Balance |
|---|-------|-------------------|---------|
| ▶ | 1     | SBI_Chamrajpet    | 2100    |
|   | 2     | SBI_ResidencyRoad | 5250    |
|   | 3     | SBI_ShivajiRoad   | 6300    |
|   | 4     | SBI_Parliament    | 9450    |
|   | 5     | SBI_Jantarmanatar | 8400    |
|   | 6     | SBI_ShivajiRoad   | 4200    |
|   | 8     | SBI_ResidencyRoad | 4200    |
|   | 9     | SBI_Parliament    | 3150    |
|   | 10    | SBI_ResidencyRoad | 5250    |
|   | 11    | SBI_Jantarmanatar | 2100    |
| • | NULL  | NULL              | NULL    |

## Program 5 Employee Database

ER Diagram



Schema Diagram:



**Question:**

- i. Using Scheme diagram, create tables by properly specifying the primary keys and the foreign keys.
- ii. Enter greater than five tuples for each table.
- iii. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru.
- iv. Get Employee ID's of those employees who didn't receive incentives.
- v. 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 Employee;  
  
USE Employee;
```

**Create Table:**

```
CREATE TABLE DEPT (deptno INT PRIMARY KEY, dname VARCHAR(20), dloc VARCHAR(20));  
  
CREATE TABLE EMPLOYEE (empno INT PRIMARY KEY, ename VARCHAR(20), mgr_no INT,  
hiredate DATE, sal INT, deptno INT, FOREIGN KEY (deptno) REFERENCES DEPT(deptno));  
  
CREATE TABLE PROJECT (pno INT PRIMARY KEY, ploc VARCHAR(20), pname VARCHAR(20));  
  
CREATE TABLE ASSIGNED_TO (empno INT, pno INT, job_role VARCHAR(20), PRIMARY KEY  
(empno, pno), FOREIGN KEY (empno) REFERENCES EMPLOYEE(empno), FOREIGN KEY (pno)  
REFERENCES PROJECT(pno));  
  
CREATE TABLE INCENTIVES (empno INT, incentive_date DATE, incentive_amount INT, FOREIGN  
KEY (empno) REFERENCES EMPLOYEE(empno));
```

## Structure of the table:

DESC DEPT;

|   | Field  | Type        | Null | Key | Default | Extra |
|---|--------|-------------|------|-----|---------|-------|
| ▶ | DEPTNO | int         | NO   | PRI | NULL    |       |
|   | DNAME  | varchar(20) | YES  |     | NULL    |       |
|   | DLOC   | varchar(20) | YES  |     | NULL    |       |

DESC EMPLOYEE;

|   | Field    | Type        | Null | Key | Default | Extra |
|---|----------|-------------|------|-----|---------|-------|
| ▶ | EMPNO    | int         | NO   | PRI | NULL    |       |
|   | ENAME    | varchar(20) | YES  |     | NULL    |       |
|   | MGR_NO   | int         | YES  |     | NULL    |       |
|   | HIREDATE | date        | YES  |     | NULL    |       |
|   | SAL      | int         | YES  |     | NULL    |       |
|   | DEPTNO   | int         | YES  | MUL | NULL    |       |

DESC PROJECT;

|   | Field | Type        | Null | Key | Default | Extra |
|---|-------|-------------|------|-----|---------|-------|
| ▶ | PNO   | int         | NO   | PRI | NULL    |       |
|   | PLOC  | varchar(20) | YES  |     | NULL    |       |
|   | PNAME | varchar(20) | YES  |     | NULL    |       |

DESC ASSIGNED\_TO;

|   | Field    | Type        | Null | Key | Default | Extra |
|---|----------|-------------|------|-----|---------|-------|
| ▶ | EMPNO    | int         | NO   | PRI | NULL    |       |
|   | PNO      | int         | NO   | PRI | NULL    |       |
|   | JOB_ROLE | varchar(20) | YES  |     | NULL    |       |

DESC INCENTIVES;

|   | Field            | Type | Null | Key | Default | Extra |
|---|------------------|------|------|-----|---------|-------|
| ▶ | EMPNO            | int  | YES  | MUL | NULL    |       |
|   | INCENTIVE_DATE   | date | YES  |     | NULL    |       |
|   | INCENTIVE_AMOUNT | int  | YES  |     | NULL    |       |



## Inserting Values To The Table:

```
INSERT INTO DEPT VALUES (10, 'HR', 'Bengaluru');
INSERT INTO DEPT VALUES (20, 'Finance', 'Hyderabad');
INSERT INTO DEPT VALUES (30, 'IT', 'Mysuru');
INSERT INTO DEPT VALUES (40, 'Admin', 'Chennai');
INSERT INTO DEPT VALUES (50, 'Marketing', 'Mumbai');
SELECT * FROM DEPT;
```

|   | DEPTNO | DNAME     | DLOC      |
|---|--------|-----------|-----------|
| ▶ | 10     | HR        | Bengaluru |
|   | 20     | Finance   | Hyderabad |
|   | 30     | IT        | Mysuru    |
|   | 40     | Admin     | Chennai   |
|   | 50     | Marketing | Mumbai    |
| * | NULL   | NULL      | NULL      |

```
INSERT INTO EMPLOYEE VALUES (1001, 'Ravi', 1005, '2018-01-12', 45000, 10);
INSERT INTO EMPLOYEE VALUES (1002, 'Kiran', 1005, '2019-03-20', 52000, 20);
INSERT INTO EMPLOYEE VALUES (1003, 'Sneha', 1006, '2020-05-10', 48000, 30);
INSERT INTO EMPLOYEE VALUES (1004, 'Deepa', 1006, '2017-11-01', 60000, 40);
INSERT INTO EMPLOYEE VALUES (1005, 'Arun', NULL, '2015-07-14', 75000, 10);
INSERT INTO EMPLOYEE VALUES (1006, 'Ramesh', NULL, '2016-09-30', 80000, 30);
SELECT * FROM EMPLOYEE;
```

|   | EMPNO | ENAME  | MGR_NO | HIREDATE   | SAL   | DEPTNO |
|---|-------|--------|--------|------------|-------|--------|
| ▶ | 1001  | Ravi   | 1005   | 2018-01-12 | 45000 | 10     |
|   | 1002  | Kiran  | 1005   | 2019-03-20 | 52000 | 20     |
|   | 1003  | Sneha  | 1006   | 2020-05-10 | 48000 | 30     |
|   | 1004  | Deepa  | 1006   | 2017-11-01 | 60000 | 40     |
|   | 1005  | Arun   | NULL   | 2015-07-14 | 75000 | 10     |
|   | 1006  | Ramesh | NULL   | 2016-09-30 | 80000 | 30     |
| * | NULL  | NULL   | NULL   | NULL       | NULL  | NULL   |

```
INSERT INTO PROJECT VALUES (501, 'Bengaluru', 'Payroll');
INSERT INTO PROJECT VALUES (502, 'Hyderabad', 'ERP');
INSERT INTO PROJECT VALUES (503, 'Mysuru', 'CRM');
INSERT INTO PROJECT VALUES (504, 'Chennai', 'HRMS');
INSERT INTO PROJECT VALUES (505, 'Mumbai', 'MarketingSuite');
SELECT * FROM PROJECT;
```

|   | PNO  | PLOC      | PNAME          |
|---|------|-----------|----------------|
| ▶ | 501  | Bengaluru | Payroll        |
|   | 502  | Hyderabad | ERP            |
|   | 503  | Mysuru    | CRM            |
|   | 504  | Chennai   | HRMS           |
|   | 505  | Mumbai    | MarketingSuite |
| * | NULL | NULL      | NULL           |

```

INSERT INTO ASSIGNED_TO VALUES (1001, 501, 'Developer');
INSERT INTO ASSIGNED_TO VALUES (1002, 502, 'Analyst');
INSERT INTO ASSIGNED_TO VALUES (1003, 503, 'Tester');
INSERT INTO ASSIGNED_TO VALUES (1004, 504, 'Manager');
INSERT INTO ASSIGNED_TO VALUES (1005, 501, 'Lead');
INSERT INTO ASSIGNED_TO VALUES (1006, 503, 'Architect');
SELECT * FROM ASSIGNED_TO;

```

|   | EMPNO | PNO  | JOB_ROLE  |
|---|-------|------|-----------|
| ▶ | 1001  | 501  | Developer |
|   | 1002  | 502  | Analyst   |
|   | 1003  | 503  | Tester    |
|   | 1004  | 504  | Manager   |
|   | 1005  | 501  | Lead      |
|   | 1006  | 503  | Architect |
| * | NULL  | NULL | NULL      |

```

INSERT INTO INCENTIVES VALUES (1001, '2023-03-10', 5000);
INSERT INTO INCENTIVES VALUES (1002, '2023-03-15', 7000);
INSERT INTO INCENTIVES VALUES (1004, '2023-03-18', 4000);
INSERT INTO INCENTIVES VALUES (1005, '2023-03-22', 8000);
SELECT * FROM INCENTIVES;

```

|   | EMPNO | INCENTIVE_DATE | INCENTIVE_AMOUNT |
|---|-------|----------------|------------------|
| ▶ | 1001  | 2023-03-10     | 5000             |
|   | 1002  | 2023-03-15     | 7000             |
|   | 1004  | 2023-03-18     | 4000             |
|   | 1005  | 2023-03-22     | 8000             |

### Queries:

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

```

SELECT DISTINCT a.empno FROM ASSIGNED_TO a JOIN PROJECT p ON a.pno=p.pno
WHERE p.ploc IN ('Bengaluru','Hyderabad','Mysuru');

```

| EMPNO |
|-------|
| 1001  |
| 1005  |
| 1002  |
| 1003  |
| 1006  |

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

SELECT e.empno FROM EMPLOYEE e WHERE e.empno NOT IN (SELECT empno FROM INCENTIVES);

| EMPNO |
|-------|
| 1003  |
| 1006  |
| HULL  |

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

SELECT e.ename,e.empno,d.dname,a.job\_role,d.dloc AS dept\_location,p.ploc AS project\_location  
FROM EMPLOYEE e JOIN DEPT d ON e.deptno=d.deptno JOIN ASSIGNED\_TO a ON  
e.empno=a.empno JOIN PROJECT p ON a.pno=p.pno;

|   | ENAME  | EMPNO | DNAME   | JOB_ROLE  | Dept_Location | Project_Location |
|---|--------|-------|---------|-----------|---------------|------------------|
| ▶ | Ravi   | 1001  | HR      | Developer | Bengaluru     | Bengaluru        |
|   | Arun   | 1005  | HR      | Lead      | Bengaluru     | Bengaluru        |
|   | Kiran  | 1002  | Finance | Analyst   | Hyderabad     | Hyderabad        |
|   | Sneha  | 1003  | IT      | Tester    | Mysuru        | Mysuru           |
|   | Ramesh | 1006  | IT      | Architect | Mysuru        | Mysuru           |
|   | Deepa  | 1004  | Admin   | Manager   | Chennai       | Chennai          |

## Program 6

### More Queries On Employee Database

#### Question:

- i. Using Scheme diagram (under Program-5), Create tables by properly specifying the primary keys and the foreign keys.
- ii. Enter greater than five tuples for each table.
- iii. List the name of the managers with the maximum employees.
- iv. Display those managers name whose salary is more than average salary of his employee.
- v. Find the name of the second top level managers of each department.
- vi. Find the employee details who got second maximum incentive in January 2019.
- vii. Display those employees who are working in the same department where his manager is working.

#### Queries:

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

```
SELECT e.ename FROM EMPLOYEE e WHERE e.empno IN (SELECT mgr_no FROM  
EMPLOYEE GROUP BY mgr_no HAVING COUNT(*)=(SELECT MAX(cnt) FROM  
(SELECT COUNT(*) AS cnt FROM EMPLOYEE GROUP BY mgr_no) t));
```

|   |        |
|---|--------|
|   | ename  |
| ▶ | Arun   |
|   | Ramesh |

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

```
SELECT m.ename FROM EMPLOYEE m WHERE m.empno IN (SELECT e.mgr_no FROM  
EMPLOYEE e GROUP BY e.mgr_no HAVING m.sal>AVG(e.sal));
```

|   |        |
|---|--------|
|   | ename  |
| ▶ | Arun   |
|   | Ramesh |

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

```
SELECT e.ename FROM EMPLOYEE e WHERE e.mgr_no IN (SELECT empno FROM
EMPLOYEE WHERE mgr_no IS NULL) GROUP BY e.deptno,e.empno;
```

|   |       |
|---|-------|
|   | ename |
| ▶ | Ravi  |
|   | Kiran |
|   | Sneha |
|   | Deepa |

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

```
SELECT e.* FROM EMPLOYEE e WHERE e.empno=(SELECT empno FROM
INCENTIVES WHERE MONTH(incentive_date)=1 AND YEAR(incentive_date)=2019
ORDER BY incentive_amount DESC LIMIT 1 OFFSET 1);
```

|   |       |       |        |          |      |        |
|---|-------|-------|--------|----------|------|--------|
|   | EMPNO | ENAME | MGR_NO | HIREDATE | SAL  | DEPTNO |
| * | NULL  | NULL  | NULL   | NULL     | NULL | NULL   |

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

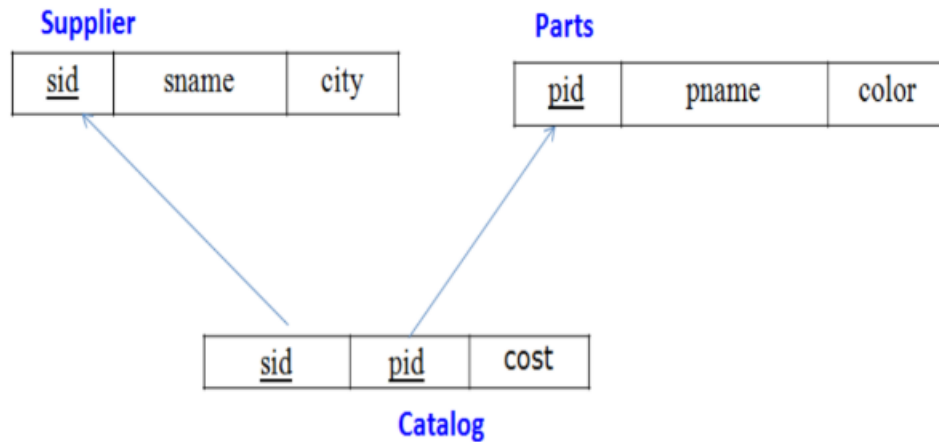
```
SELECT e.ename FROM EMPLOYEE e JOIN EMPLOYEE m ON e.mgr_no=m.empno
WHERE e.deptno=m.deptno;
```

|   |       |
|---|-------|
|   | ename |
| ▶ | Ravi  |
|   | Sneha |

## Program 7

### Supplier Database

#### Schema Diagram:



#### Question:

- i. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
- ii. Insert appropriate records in each table.
- iii. Find the pnames of parts for which there is some supplier.
- iv. Find the snames of suppliers who supply every part.
- v. Find the snames of suppliers who supply every red part.
- vi. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
- vii. 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).
- viii. For each part, find the sname of the supplier who charges the most for that part.

#### Create Database:

```
CREATE DATABASE Supplier;
USE Supplier;
```

### Create Table:

```
CREATE TABLE SUPPLIER (sid INT, sname VARCHAR(20), city VARCHAR(10),  
PRIMARY KEY(sid));
```

```
CREATE TABLE PARTS (pid INT PRIMARY KEY, pname VARCHAR(20), color  
VARCHAR(20));
```

```
CREATE TABLE CATALOG (sid INT, pid INT, cost INT, FOREIGN KEY (sid)  
REFERENCES SUPPLIER(sid), FOREIGN KEY (pid) REFERENCES PARTS(pid));
```

### Structure of the table:

DESC SUPPLIER;

|   | Field | Type        | Null | Key | Default | Extra |
|---|-------|-------------|------|-----|---------|-------|
| ► | sid   | int         | NO   | PRI | NULL    |       |
|   | sname | varchar(20) | YES  |     | NULL    |       |
|   | city  | varchar(10) | YES  |     | NULL    |       |

DESC PARTS;

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

DESC CATALOG;

|   | Field | Type | Null | Key | Default | Extra |
|---|-------|------|------|-----|---------|-------|
| ► | sid   | int  | YES  | MUL | NULL    |       |
|   | pid   | int  | YES  | MUL | NULL    |       |
|   | cost  | int  | YES  |     | NULL    |       |

## Inserting Values To The Table:

```
INSERT INTO SUPPLIER VALUES(10001,'Acme Widget','Bangalore');
INSERT INTO SUPPLIER VALUES(10002,'Johns','Kolkata');
INSERT INTO SUPPLIER VALUES(10003,'Vimal','Mumbai');
INSERT INTO SUPPLIER VALUES(10004,'Reliance','Delhi');
SELECT * FROM SUPPLIER;
```

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

```
INSERT INTO PARTS VALUES(20001,'Book','Red');
INSERT INTO PARTS VALUES(20002,'Pen','Red');
INSERT INTO PARTS VALUES(20003,'Pencil','Green');
INSERT INTO PARTS VALUES(20004,'Mobile','Green');
INSERT INTO PARTS VALUES(20005,'Charger','Black');
SELECT * FROM PARTS;
```

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

```
INSERT INTO CATALOG VALUES(10001,20001,10);
INSERT INTO CATALOG VALUES(10001,20002,10);
INSERT INTO CATALOG VALUES(10001,20003,30);
INSERT INTO CATALOG VALUES(10001,20004,10);
INSERT INTO CATALOG VALUES(10001,20005,10);
INSERT INTO CATALOG VALUES(10002,20001,10);
INSERT INTO CATALOG VALUES(10002,20002,20);
INSERT INTO CATALOG VALUES(10003,20003,30);
INSERT INTO CATALOG VALUES(10004,20003,40);
SELECT * FROM CATALOG;
```



|   | sid   | pid   | cost |
|---|-------|-------|------|
| ▶ | 10001 | 20001 | 10   |
|   | 10001 | 20002 | 10   |
|   | 10001 | 20003 | 30   |
|   | 10001 | 20004 | 10   |
|   | 10001 | 20005 | 10   |
|   | 10002 | 20001 | 10   |
|   | 10002 | 20002 | 20   |
|   | 10003 | 20003 | 30   |
|   | 10004 | 20003 | 40   |

### Queries:

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

```
SELECT DISTINCT p.pname FROM PARTS p WHERE p.pid IN (SELECT pid FROM CATALOG c,SUPPLIER s WHERE s.sid=c.sid);
```

|   | pname   |
|---|---------|
| ▶ | Book    |
|   | Pen     |
|   | Pencil  |
|   | Mobile  |
|   | Charger |

- Find the snames of suppliers who supply every part.

```
SELECT sname FROM SUPPLIER WHERE NOT EXISTS (SELECT pid FROM PARTS WHERE pid NOT IN (SELECT pid FROM CATALOG WHERE SUPPLIER.sid=CATALOG.sid));
```

|   | sname       |
|---|-------------|
| ▶ | Acme Widget |

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

```
SELECT sname FROM SUPPLIER WHERE EXISTS (SELECT pid FROM CATALOG WHERE pid IN (SELECT pid FROM PARTS WHERE PARTS.color='red') AND CATALOG.sid=SUPPLIER.sid);
```

|   | sname       |
|---|-------------|
| ▶ | Acme Widget |
|   | Johns       |

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

```
SELECT pname FROM PARTS WHERE pid IN (SELECT pid FROM CATALOG WHERE
sid=10001) AND pid NOT IN (SELECT pid FROM CATALOG WHERE sid!=10001);
```

|   | pname   |
|---|---------|
| ▶ | Mobile  |
|   | Charger |

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

```
SELECT DISTINCT c1.sid FROM CATALOG c1 WHERE c1.cost>(SELECT AVG(c2.cost)
FROM CATALOG c2 WHERE c2.pid=c1.pid);
```

|   | sid   |
|---|-------|
| ▶ | 10002 |
|   | 10004 |

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

```
SELECT p.pid,s.sname FROM PARTS p JOIN CATALOG c ON p.pid=c.pid JOIN
SUPPLIER s ON c.sid=s.sid WHERE c.cost=(SELECT MAX(cost) FROM CATALOG
WHERE p.pid=pid);
```

|   | pid   | sname       |
|---|-------|-------------|
| ▶ | 20001 | Acme Widget |
|   | 20004 | Acme Widget |
|   | 20005 | Acme Widget |
|   | 20001 | Johns       |
|   | 20002 | Johns       |
|   | 20003 | Reliance    |

## **Program 8**

### **NoSQL Student Database**

#### **Question:**

Perform the following DB operations using MongoDB.

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

```
USE Student
```

```
db.createCollection("student")
```

- **Insert appropriate values.**

```
db.student.insertMany([
```

```
  { Rollno: 10, Age: 20, ContactNo: "9876543210", EmailId: "abc10@gmail.com", Name: "ABC" },
```

```
  { Rollno: 11, Age: 21, ContactNo: "9876543211", EmailId: "abc11@gmail.com", Name: "ABC" },
```

```
  { Rollno: 12, Age: 22, ContactNo: "9876543212", EmailId: "abc12@gmail.com", Name: "XYZ" }
]
```

```
)
```

```

{ ok: 1 }
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('694d2f8285c85952c88de665'),
    '1': ObjectId('694d2f8285c85952c88de666'),
    '2': ObjectId('694d2f8285c85952c88de667')
  }
}

```

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

```

db.student.updateOne(
  { Rollno: 10 },
  { $set: { EmailId: "updated10@gmail.com" } }
)

```

```

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

```

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

```

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

```

```

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

```

- **Export the created table into local file system.**

```
mongoexport --db=Student --collection=student --out=student.json
```

- **Drop the table.**

```
db.student.drop()
```

```
Atlas atlas-uyucz2-shard-0 [primary] test> db.Student.drop();  
true
```

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

```
mongoimport --db=Student --collection=student --type=csv --headerline --file=student.csv
```

## **Program 9**

### **NoSQL Customer Database**

#### **Question:**

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 BankDB

```
db.createCollection("Customers")
```

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

```
db.Customers.insertMany([  
  { Cust_id: 1, Acc_Bal: 1500, Acc_Type: "Z" },  
  { Cust_id: 2, Acc_Bal: 800, Acc_Type: "X" },  
  { Cust_id: 3, Acc_Bal: 2000, Acc_Type: "Z" },  
  { Cust_id: 4, Acc_Bal: 1200, Acc_Type: "Y" },  
  { Cust_id: 5, Acc_Bal: 2500, Acc_Type: "Z" }  
)
```

)

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('694d31cab83f3797128de665'),
    '1': ObjectId('694d31cab83f3797128de666'),
    '2': ObjectId('694d31cab83f3797128de667'),
    '3': ObjectId('694d31cab83f3797128de668'),
    '4': ObjectId('694d31cab83f3797128de669')
  }
}
```

- **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: 5, totalBalance: 2500 },
  { _id: 1, totalBalance: 1500 },
  { _id: 3, totalBalance: 2000 }
]
```

- **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: 5, minBalance: 2500, maxBalance: 2500 },
  { _id: 3, minBalance: 2000, maxBalance: 2000 },
  { _id: 1, minBalance: 1500, maxBalance: 1500 },
  { _id: 2, minBalance: 800, maxBalance: 800 },
  { _id: 4, minBalance: 1200, maxBalance: 1200 }
]
```

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

```
mongoexport --db=BankDB --collection=Customers --out=customers.json
```

- **Drop the table.**

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

```
]
Atlas atlas-13yfay-shard-0 [primary] test> db.customer.drop();
true
Atlas atlas-13yfay-shard-0 [primary] test> _
```

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

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



## **Program 10**

### **NoSQL Restaurant Database**

#### **Question:**

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

#### **Queries:**

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

```
db.createCollection("Customers")
```

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

```
db.restaurants.find({})
```

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({})
```

```
[
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'jayanagar' }
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'M G Road' }
  },
  {
    _id: ObjectId("675002dbf345f747889620bb"),
    name: 'Chinese Wok',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    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 })
```

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({})
[
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'jayanagar' }
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'M G Road' }
  },
  {
    _id: ObjectId("675002dbf345f747889620bb"),
    name: 'Chinese Wok',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

- 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(
  { "grades.score": { $lte: 10 } },
  { restaurant_id: 1, name: 1, town: 1, cuisine: 1, _id: 0 }
)
```

```
[
  {
    _id: ObjectId("67500261f345f747889620b9"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67500292f345f747889620ba"),
    name: 'Empire',
    town: 'M G Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67500316f345f747889620bc"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'japanese'
  },
  {
    _id: ObjectId("67500342f345f747889620bd"),
    name: 'WOW Momo',
    town: 'Mallechwaram',
    cuisine: 'Indian'
  }
]
```

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

```
db.restaurants.aggregate([
  { $unwind: "$grades" },
  { $group: {
    _id: "$restaurant_id",
    name: { $first: "$name" },
    avgScore: { $avg: "$grades.score" }
  }}
])
```

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }
... ])
[
  { _id: 'WOW Momo', average_score: 5 },
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Chinese Wok', average_score: 12 },
  { _id: 'Empire', average_score: 7 }
]
```

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

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

```
Atlas atlas-13yfay-shard-0 [primary] test> db.restaurants.find({ "address.zipcode": /^10/, { name: 1, "address.street": 1, _id: 0 }})
[
  { name: 'Meghna Foods', address: { street: 'jayanagar' } },
  { name: 'Empire', address: { street: 'M G Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momo', address: { street: 'Malleshwaram' } }
]
```