Muthoot Institute of Technology and Science

Varikoli P.O, Puthencruz,
Ernakulam – 682308

Master of Computer Applications

LAB RECORD

Course Name: 20MCA134 ADVANCED DBMS LAB

NAME…………………………………………….......

REG. NO……………… SEMESTER………………

MONTH & YEAR…………………………………

Muthoot
Institute of Technology & Science

Certificate

Certified that this is the Bonafide Record of Practical work done in the ………………………………………………. Lab of Muthoot Institute of Technology and Science by ***Name:*** *………………………………………………….* ***Reg.No:*** *……………………….. for the partial fulfillment of the requirement for the award of the degree of Master of Computer Applications during the year…………………….*

Head of the Department Faculty in Charge

University Exam Reg. No…………………. of …………….. 202….

Date of Examination………………………….

Internal Examiner External Examiner

Muthoot
Institute of Technology & Science

Vision of Institute

To be a centre of excellence for learning and research in engineering and technology, producing intellectually well-equipped and socially committed citizens possessing an ethical value system.

Mission of Institute

- Offer well-balanced programme of instruction, practical exercise and opportunities in technology.
- Foster innovation and ideation of technological solutions on sustainable basis.
- Nurture a value system in students and engender in them a spirit of inquiry.

INDEX

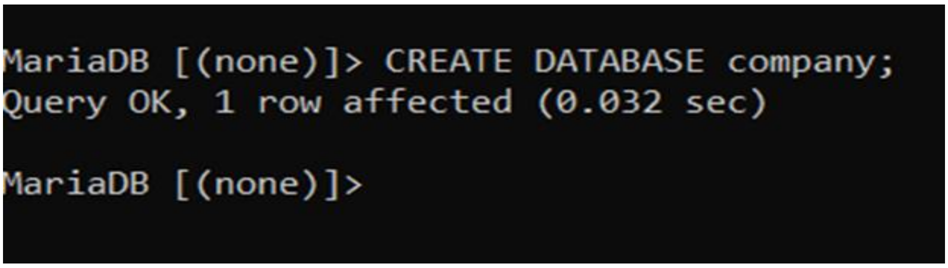
12	Create a database COLLEGE and create a collection named studlist in it. Retrieve data from collection	14-06-2023	36
13	Create a database in MongoDB named MCADB with collections named COURSE and STUDENTS and perform CRUD operations on it. Perform the CRUD operations using mongoshell and pymongo.	16-06-2023	40
14	Create a database in MongoDB named "mcadb" with collections named "course" and "students" and perform aggregate functions, and regular expressions on it.	21-06-2023	48
15	Create a collection employee. Design and run 5 aggregate pipeline functions on employee collection. Use combinations of $match, $group, $sort and $project functions.	23-06-2023	51
CO5	**Understand the basic storage architecture of distributed file systems**		
16	Build collections mcaDB documents students, course and perform shell commands to create replica set, indexing etc	28-06-2023	54
CO6	**Design and deployment of NoSQL databases with real time requirements.**		
17	Develop students’ marks calculation applications using Python and MongoDB.	30-06-2023	56

CO1

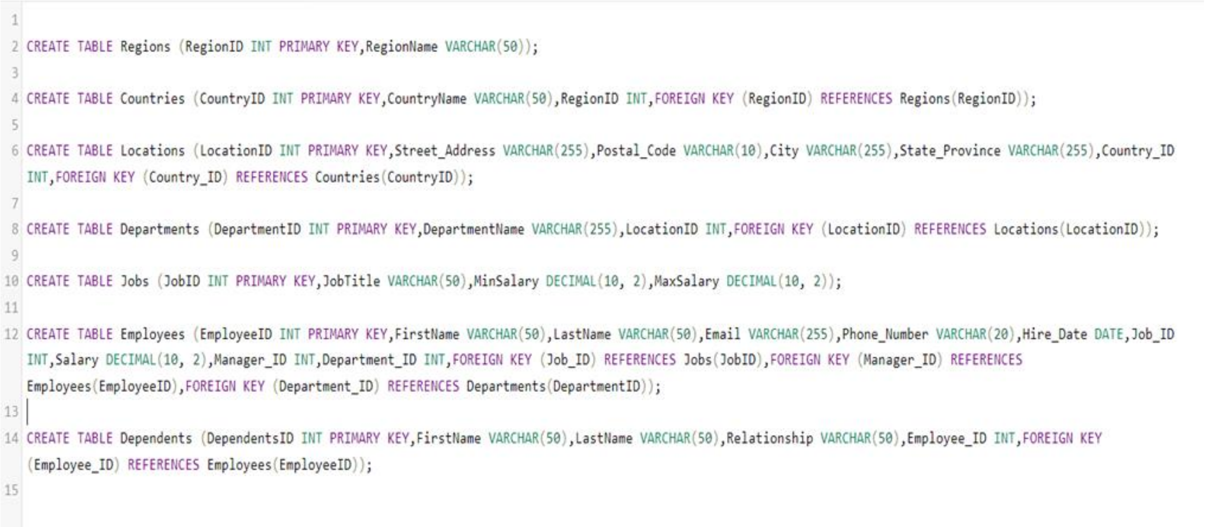
Design and build a simple relational database system and demonstrate competence with the fundamentals tasks involved with modelling, designing and implementing a database.

PROGRAM 1

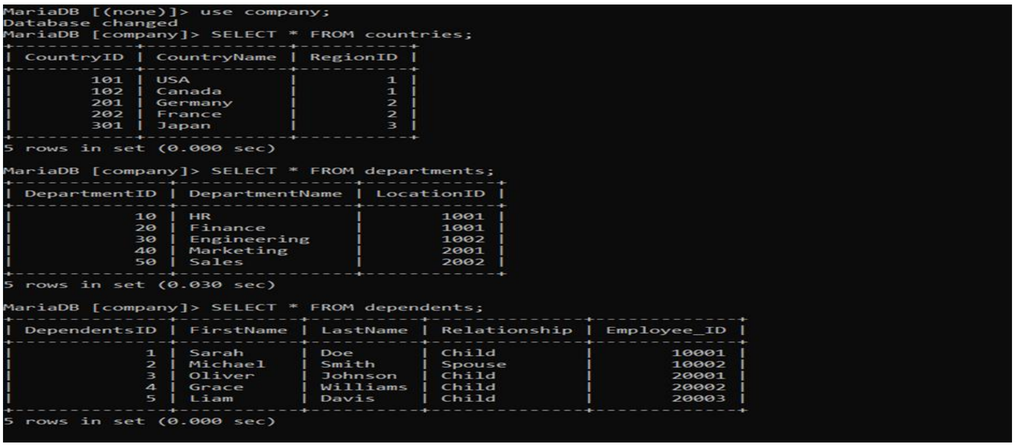
Creation of a database COMPANY, and tables using DDL commands including integrity constraints. Populate the tables with DML commands.



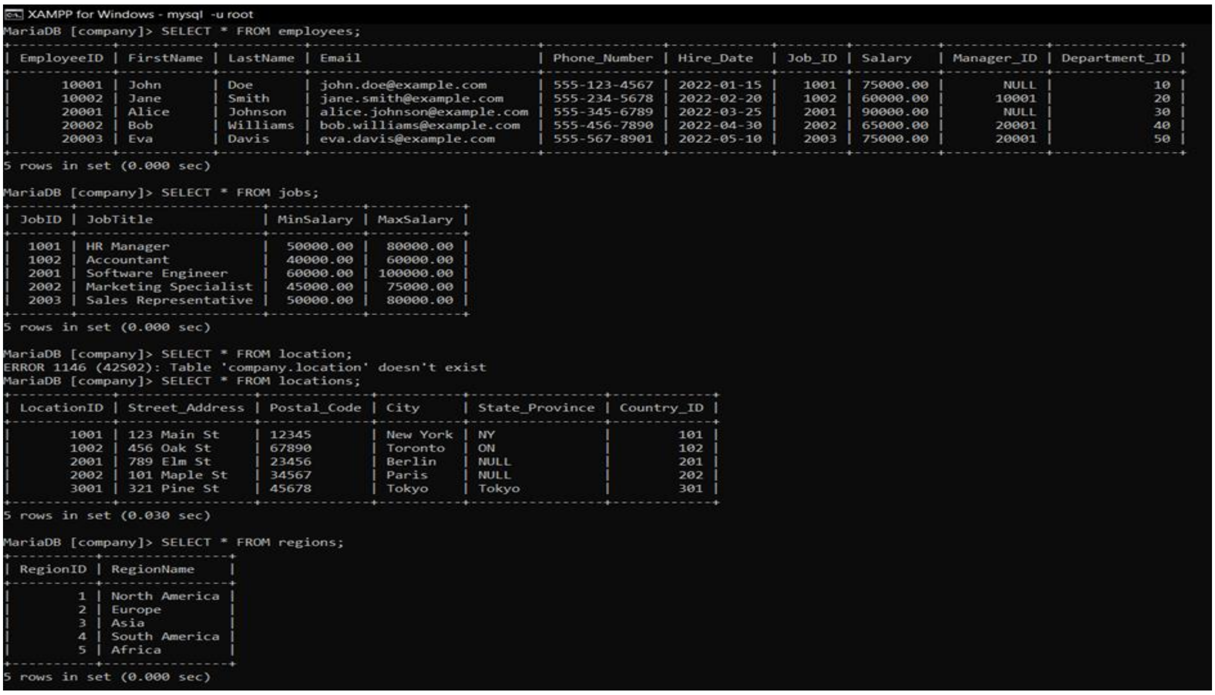
```
MariaDB [(none)]> CREATE DATABASE company;
Query OK, 1 row affected (0.032 sec)

MariaDB [(none)]>
```



```sql

CREATE TABLE Regions (RegionID INT PRIMARY KEY,RegionName VARCHAR(50));

CREATE TABLE Countries (CountryID INT PRIMARY KEY,CountryName VARCHAR(50),RegionID INT,FOREIGN KEY (RegionID) REFERENCES Regions(RegionID));

CREATE TABLE Locations (LocationID INT PRIMARY KEY,Street_Address VARCHAR(255),Postal_Code VARCHAR(10),City VARCHAR(255),State_Province VARCHAR(255),Country_ID INT,FOREIGN KEY (Country_ID) REFERENCES Countries(CountryID));

CREATE TABLE Departments (DepartmentID INT PRIMARY KEY,DepartmentName VARCHAR(255),LocationID INT,FOREIGN KEY (LocationID) REFERENCES Locations(LocationID));

CREATE TABLE Jobs (JobID INT PRIMARY KEY,JobTitle VARCHAR(50),MinSalary DECIMAL(10, 2),MaxSalary DECIMAL(10, 2));

CREATE TABLE Employees (EmployeeID INT PRIMARY KEY,FirstName VARCHAR(50),LastName VARCHAR(50),Email VARCHAR(255),Phone_Number VARCHAR(20),Hire_Date DATE,Job_ID INT,Salary DECIMAL(10, 2),Manager_ID INT,Department_ID INT,FOREIGN KEY (Job_ID) REFERENCES Jobs(JobID),FOREIGN KEY (Manager_ID) REFERENCES Employees(EmployeeID),FOREIGN KEY (Department_ID) REFERENCES Departments(DepartmentID));

CREATE TABLE Dependents (DependentsID INT PRIMARY KEY,FirstName VARCHAR(50),LastName VARCHAR(50),Relationship VARCHAR(50),Employee_ID INT,FOREIGN KEY (Employee_ID) REFERENCES Employees(EmployeeID));

```



```
MariaDB [(none)]> use company;
Database changed
MariaDB [company]> SELECT * FROM countries;
+-----------+-------------+----------+
| CountryID | CountryName | RegionID |
+-----------+-------------+----------+
|       101 | USA         |        1 |
|       102 | Canada      |        1 |
|       201 | Germany     |        2 |
|       202 | France      |        2 |
|       301 | Japan       |        3 |
+-----------+-------------+----------+
5 rows in set (0.000 sec)

MariaDB [company]> SELECT * FROM departments;
+--------------+----------------+------------+
| DepartmentID | DepartmentName | LocationID |
+--------------+----------------+------------+
|           10 | HR             |       1001 |
|           20 | Finance        |       1001 |
|           30 | Engineering    |       1002 |
|           40 | Marketing      |       2001 |
|           50 | Sales          |       2002 |
+--------------+----------------+------------+
5 rows in set (0.030 sec)

MariaDB [company]> SELECT * FROM dependents;
+--------------+-----------+----------+--------------+-------------+
| DependentsID | FirstName | LastName | Relationship | Employee_ID |
+--------------+-----------+----------+--------------+-------------+
|            1 | Sarah     | Doe      | Child        |       10001 |
|            2 | Michael   | Smith    | Spouse       |       10002 |
|            3 | Oliver    | Johnson  | Child        |       20001 |
|            4 | Grace     | Williams | Child        |       20002 |
|            5 | Liam      | Davis    | Child        |       20003 |
+--------------+-----------+----------+--------------+-------------+
5 rows in set (0.000 sec)
```



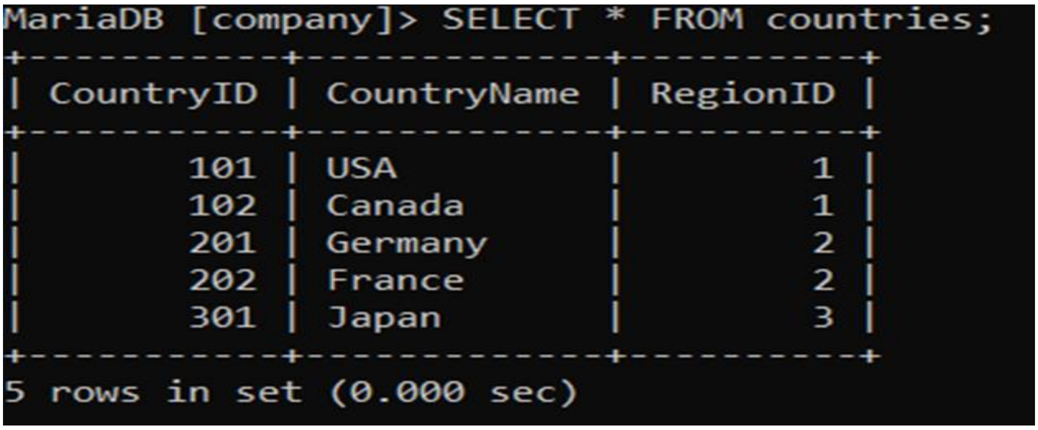
```
MariaDB [company]> SELECT * FROM employees;
+------------+-----------+----------+---------------------------+--------------+------------+--------+----------+------------+---------------+
| EmployeeID | FirstName | LastName | Email                     | Phone_Number | Hire_Date  | Job_ID | Salary   | Manager_ID | Department_ID |
+------------+-----------+----------+---------------------------+--------------+------------+--------+----------+------------+---------------+
|      10001 | John      | Doe      | john.doe@example.com      | 555-123-4567 | 2022-01-15 |   1001 | 75000.00 |       NULL |            10 |
|      10002 | Jane      | Smith    | jane.smith@example.com    | 555-234-5678 | 2022-02-20 |   1002 | 60000.00 |      10001 |            20 |
|      20001 | Alice     | Johnson  | alice.johnson@example.com | 555-345-6789 | 2022-03-25 |   2001 | 90000.00 |       NULL |            30 |
|      20002 | Bob       | Williams | bob.williams@example.com  | 555-456-7890 | 2022-04-30 |   2002 | 65000.00 |      20001 |            40 |
|      20003 | Eva       | Davis    | eva.davis@example.com     | 555-567-8901 | 2022-05-10 |   2003 | 75000.00 |      20001 |            50 |
+------------+-----------+----------+---------------------------+--------------+------------+--------+----------+------------+---------------+
5 rows in set (0.000 sec)

MariaDB [company]> SELECT * FROM jobs;
+-------+----------------------+-----------+-----------+
| JobID | JobTitle             | MinSalary | MaxSalary |
+-------+----------------------+-----------+-----------+
|  1001 | HR Manager           |  50000.00 |  80000.00 |
|  1002 | Accountant           |  40000.00 |  60000.00 |
|  2001 | Software Engineer    |  60000.00 | 100000.00 |
|  2002 | Marketing Specialist |  45000.00 |  75000.00 |
|  2003 | Sales Representative |  50000.00 |  80000.00 |
+-------+----------------------+-----------+-----------+
5 rows in set (0.000 sec)

MariaDB [company]> SELECT * FROM location;
ERROR 1146 (42S02): Table 'company.location' doesn't exist
MariaDB [company]> SELECT * FROM locations;
+------------+----------------+-------------+----------+----------------+------------+
| LocationID | Street_Address | Postal_Code | City     | State_Province | Country_ID |
+------------+----------------+-------------+----------+----------------+------------+
|       1001 | 123 Main St    | 12345       | New York | NY             |        101 |
|       1002 | 456 Oak St     | 67890       | Toronto  | ON             |        102 |
|       2001 | 789 Elm St     | 23456       | Berlin   | NULL           |        201 |
|       2002 | 101 Maple St   | 34567       | Paris    | NULL           |        202 |
|       3001 | 321 Pine St    | 45678       | Tokyo    | Tokyo          |        301 |
+------------+----------------+-------------+----------+----------------+------------+
5 rows in set (0.030 sec)

MariaDB [company]> SELECT * FROM regions;
+----------+---------------+
| RegionID | RegionName    |
+----------+---------------+
|        1 | North America |
|        2 | Europe        |
|        3 | Asia          |
|        4 | South America |
|        5 | Africa        |
+----------+---------------+
5 rows in set (0.000 sec)
```

PROGRAM 2

AIM:

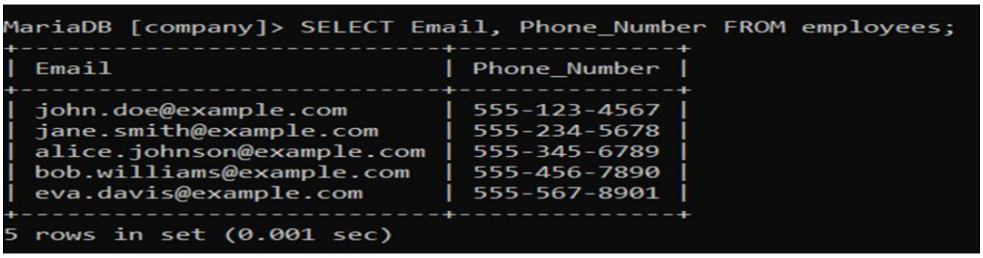
Data Retrieval using DQL commands with conditions, and subqueries in COMPANY database

a. Write a query to display all the countries.



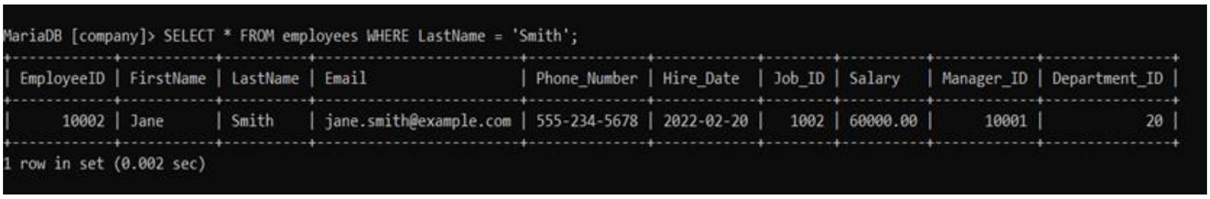
```
MariaDB [company]> SELECT * FROM countries;
+-----------+-------------+----------+
| CountryID | CountryName | RegionID |
+-----------+-------------+----------+
|       101 | USA         |        1 |
|       102 | Canada      |        1 |
|       201 | Germany     |        2 |
|       202 | France      |        2 |
|       301 | Japan       |        3 |
+-----------+-------------+----------+
5 rows in set (0.000 sec)
```

b.Write a query to display specific columns like email and phone number for all the employees.



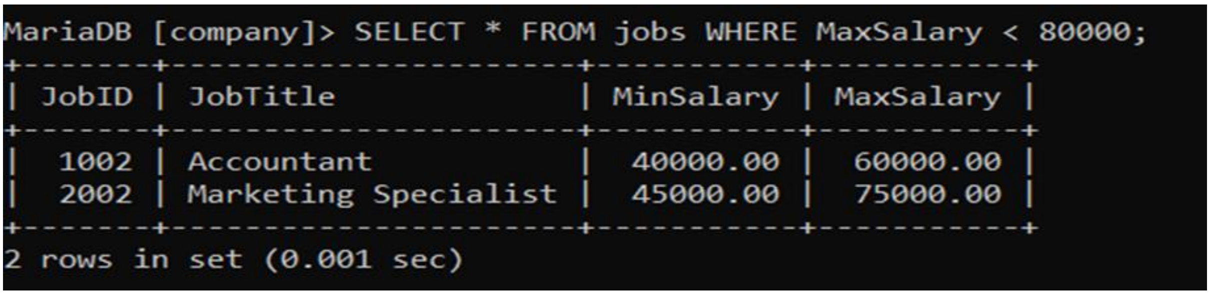
```
MariaDB [company]> SELECT Email, Phone_Number FROM employees;
+---------------------------+--------------+
| Email                     | Phone_Number |
+---------------------------+--------------+
| john.doe@example.com      | 555-123-4567 |
| jane.smith@example.com    | 555-234-5678 |
| alice.johnson@example.com | 555-345-6789 |
| bob.williams@example.com  | 555-456-7890 |
| eva.davis@example.com     | 555-567-8901 |
+---------------------------+--------------+
5 rows in set (0.001 sec)
```

c.Write a query to display the data of employee whose last name is “smith”.



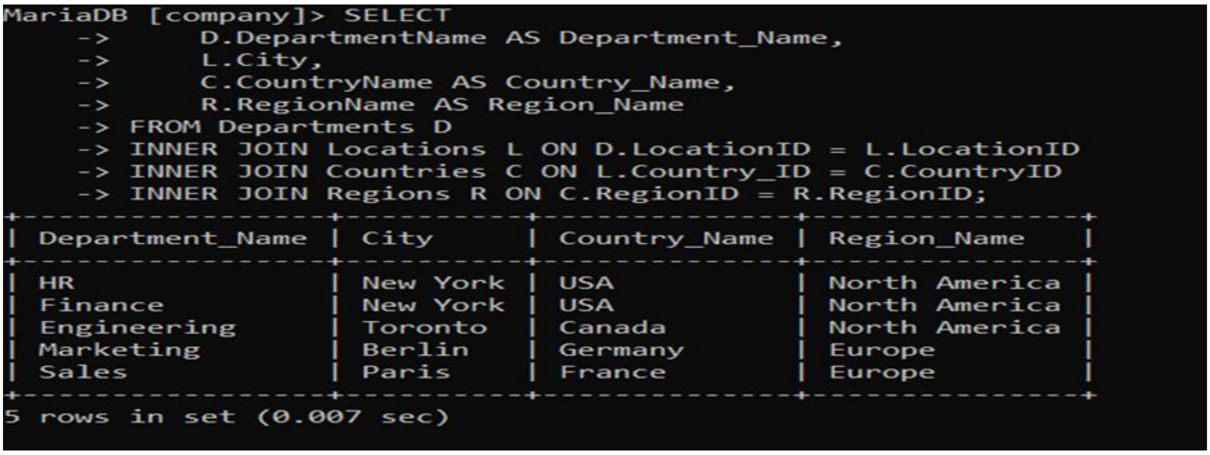
```
MariaDB [company]> SELECT * FROM employees WHERE LastName = 'Smith';
+------------+-----------+----------+------------------------+--------------+------------+--------+----------+------------+---------------+
| EmployeeID | FirstName | LastName | Email                  | Phone_Number | Hire_Date  | Job_ID | Salary   | Manager_ID | Department_ID |
+------------+-----------+----------+------------------------+--------------+------------+--------+----------+------------+---------------+
|      10002 | Jane      | Smith    | jane.smith@example.com | 555-234-5678 | 2022-02-20 |   1002 | 60000.00 |      10001 |            20 |
+------------+-----------+----------+------------------------+--------------+------------+--------+----------+------------+---------------+
1 row in set (0.002 sec)
```

d.Write a query to display jobs where the maximum salary is less than 80000.



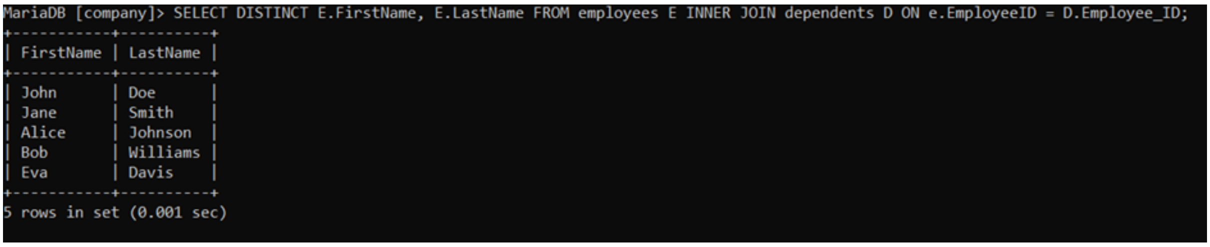
```
MariaDB [company]> SELECT * FROM jobs WHERE MaxSalary < 80000;
+-------+----------------------+-----------+-----------+
| JobID | JobTitle             | MinSalary | MaxSalary |
+-------+----------------------+-----------+-----------+
|  1002 | Accountant           |  40000.00 |  60000.00 |
|  2002 | Marketing Specialist |  45000.00 |  75000.00 |
+-------+----------------------+-----------+-----------+
2 rows in set (0.001 sec)
```

e.Display names of all departments, its city, country, and region names with a single query.



```
MariaDB [company]> SELECT
    ->     D.DepartmentName AS Department_Name,
    ->     L.City,
    ->     C.CountryName AS Country_Name,
    ->     R.RegionName AS Region_Name
    -> FROM Departments D
    -> INNER JOIN Locations L ON D.LocationID = L.LocationID
    -> INNER JOIN Countries C ON L.Country_ID = C.CountryID
    -> INNER JOIN Regions R ON C.RegionID = R.RegionID;
+-----------------+----------+--------------+---------------+
| Department_Name | City     | Country_Name | Region_Name   |
+-----------------+----------+--------------+---------------+
| HR              | New York | USA          | North America |
| Finance         | New York | USA          | North America |
| Engineering     | Toronto  | Canada       | North America |
| Marketing       | Berlin   | Germany      | Europe        |
| Sales           | Paris    | France       | Europe        |
+-----------------+----------+--------------+---------------+
5 rows in set (0.007 sec)
```

f.Write a query to display the name of the employees who has dependents.



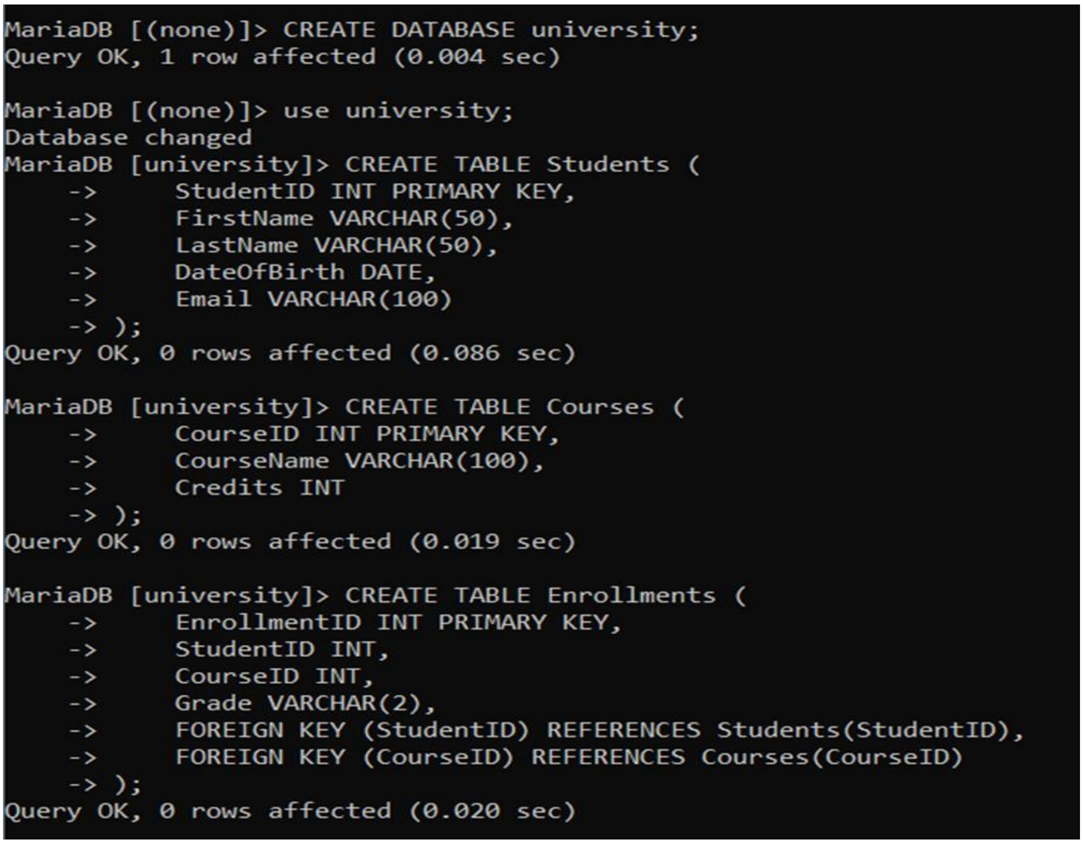
```
MariaDB [company]> SELECT DISTINCT E.FirstName, E.LastName FROM employees E INNER JOIN dependents D ON e.EmployeeID = D.Employee_ID;
+-----------+----------+
| FirstName | LastName |
+-----------+----------+
| John      | Doe      |
| Jane      | Smith    |
| Alice     | Johnson  |
| Bob       | Williams |
| Eva       | Davis    |
+-----------+----------+
5 rows in set (0.001 sec)
```

PROGRAM 3

AIM:

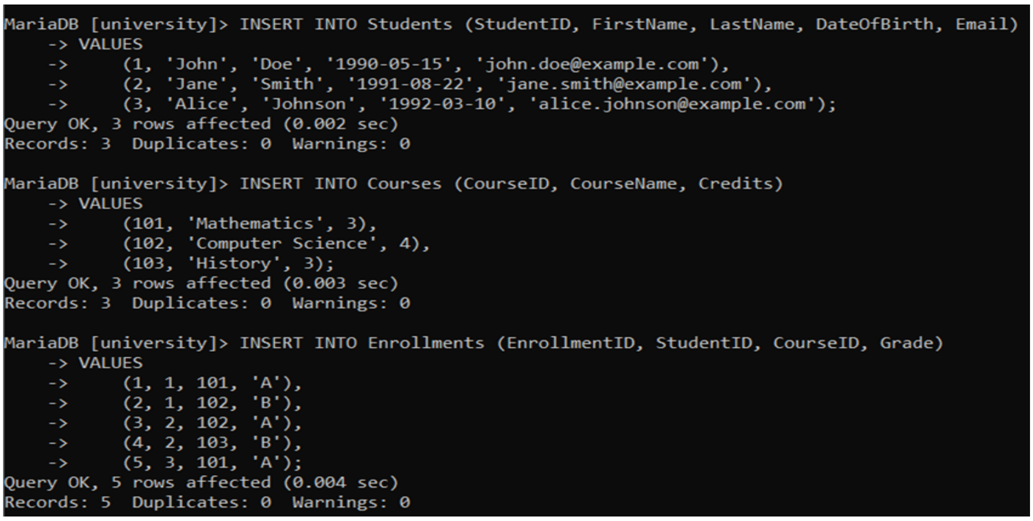
Creation of a UNIVERSITY database for maintaining information concerning students, courses, and grades in a university environment. DDl, and DML ---commands.

a. ----- Creating the database and defining the tables:



```
MariaDB [(none)]> CREATE DATABASE university;
Query OK, 1 row affected (0.004 sec)

MariaDB [(none)]> use university;
Database changed
MariaDB [university]> CREATE TABLE Students (
    ->     StudentID INT PRIMARY KEY,
    ->     FirstName VARCHAR(50),
    ->     LastName VARCHAR(50),
    ->     DateOfBirth DATE,
    ->     Email VARCHAR(100)
    -> );
Query OK, 0 rows affected (0.086 sec)

MariaDB [university]> CREATE TABLE Courses (
    ->     CourseID INT PRIMARY KEY,
    ->     CourseName VARCHAR(100),
    ->     Credits INT
    -> );
Query OK, 0 rows affected (0.019 sec)

MariaDB [university]> CREATE TABLE Enrollments (
    ->     EnrollmentID INT PRIMARY KEY,
    ->     StudentID INT,
    ->     CourseID INT,
    ->     Grade VARCHAR(2),
    ->     FOREIGN KEY (StudentID) REFERENCES Students(StudentID),
    ->     FOREIGN KEY (CourseID) REFERENCES Courses(CourseID)
    -> );
Query OK, 0 rows affected (0.020 sec)
```

b. insert some sample data into the tables using DML commands:



```
MariaDB [university]> INSERT INTO Students (StudentID, FirstName, LastName, DateOfBirth, Email)
    -> VALUES
    ->     (1, 'John', 'Doe', '1990-05-15', 'john.doe@example.com'),
    ->     (2, 'Jane', 'Smith', '1991-08-22', 'jane.smith@example.com'),
    ->     (3, 'Alice', 'Johnson', '1992-03-10', 'alice.johnson@example.com');
Query OK, 3 rows affected (0.002 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [university]> INSERT INTO Courses (CourseID, CourseName, Credits)
    -> VALUES
    ->     (101, 'Mathematics', 3),
    ->     (102, 'Computer Science', 4),
    ->     (103, 'History', 3);
Query OK, 3 rows affected (0.003 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [university]> INSERT INTO Enrollments (EnrollmentID, StudentID, CourseID, Grade)
    -> VALUES
    ->     (1, 1, 101, 'A'),
    ->     (2, 1, 102, 'B'),
    ->     (3, 2, 102, 'A'),
    ->     (4, 2, 103, 'B'),
    ->     (5, 3, 101, 'A');
Query OK, 5 rows affected (0.004 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

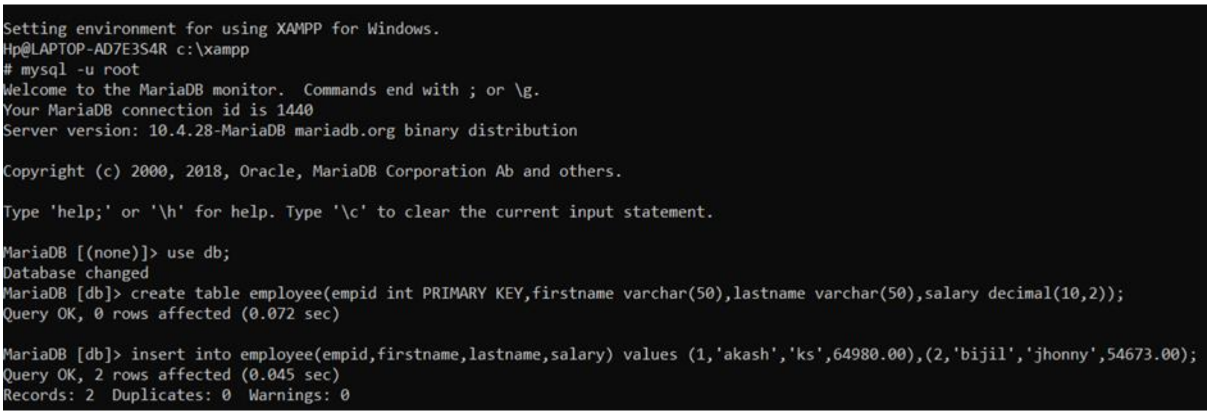
PROGRAM 4

AIM:

Apply DCL and TCL commands to impose restrictions on database.

1.

Connect as ‘root’

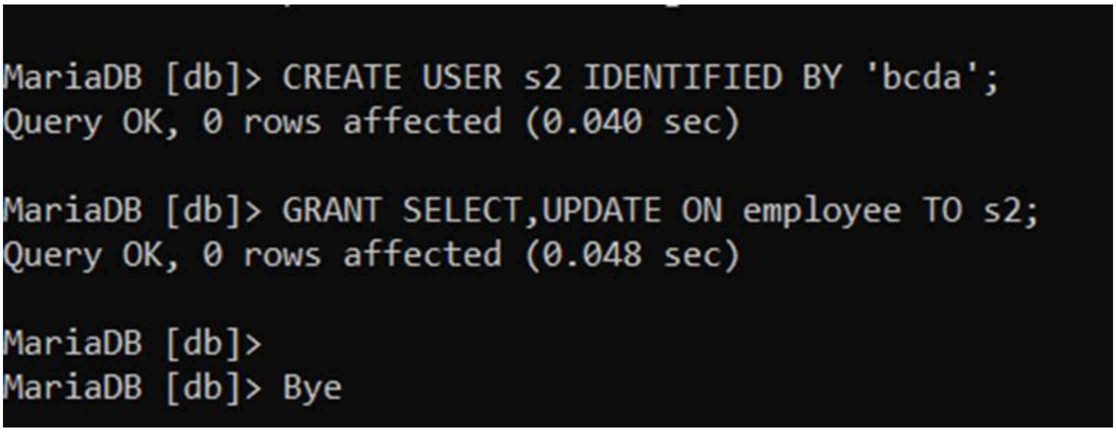


```
Setting environment for using XAMPP for Windows.
Hp@LAPTOP-AD7E3S4R c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1440
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use db;
Database changed
MariaDB [db]> create table employee(empid int PRIMARY KEY,firstname varchar(50),lastname varchar(50),salary decimal(10,2));
Query OK, 0 rows affected (0.072 sec)

MariaDB [db]> insert into employee(empid,firstname,lastname,salary) values (1,'akash','ks',64980.00),(2,'bijil','jhonny',54673.00);
Query OK, 2 rows affected (0.045 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

2.

a. Create a user s2 with a password

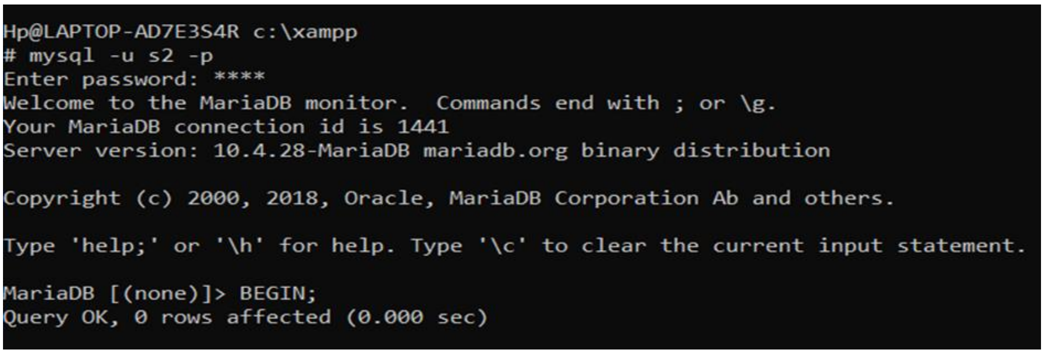
b. Grant SELECT and UPDATE privileges on the employee table to ‘s2’



```
MariaDB [db]> CREATE USER s2 IDENTIFIED BY 'bcda';
Query OK, 0 rows affected (0.040 sec)

MariaDB [db]> GRANT SELECT,UPDATE ON employee TO s2;
Query OK, 0 rows affected (0.048 sec)

MariaDB [db]>
MariaDB [db]> Bye
```

3.

Connect as ‘s2’ user and start a transaction



```
Hp@LAPTOP-AD7E3S4R c:\xampp
# mysql -u s2 -p
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1441
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> BEGIN;
Query OK, 0 rows affected (0.000 sec)
```

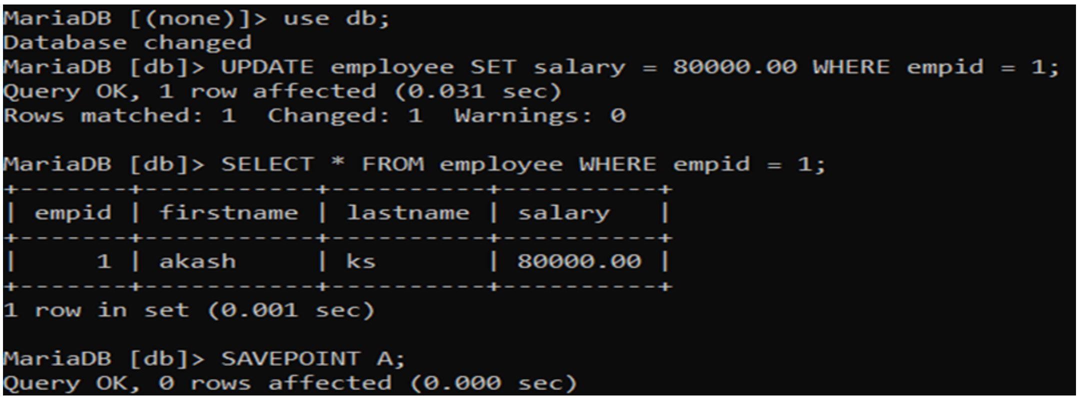
4.

a. Select the database ‘db’

b. attempt to update akash’s salary

c. check the update record.

d. Set a savepoint ‘A’.



```
MariaDB [(none)]> use db;
Database changed
MariaDB [db]> UPDATE employee SET salary = 80000.00 WHERE empid = 1;
Query OK, 1 row affected (0.031 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [db]> SELECT * FROM employee WHERE empid = 1;
+-------+-----------+----------+----------+
| empid | firstname | lastname | salary   |
+-------+-----------+----------+----------+
|     1 | akash     | ks       | 80000.00 |
+-------+-----------+----------+----------+
1 row in set (0.001 sec)

MariaDB [db]> SAVEPOINT A;
Query OK, 0 rows affected (0.000 sec)
```

5.

a. attempt to update bijil’s salary (this will also work)

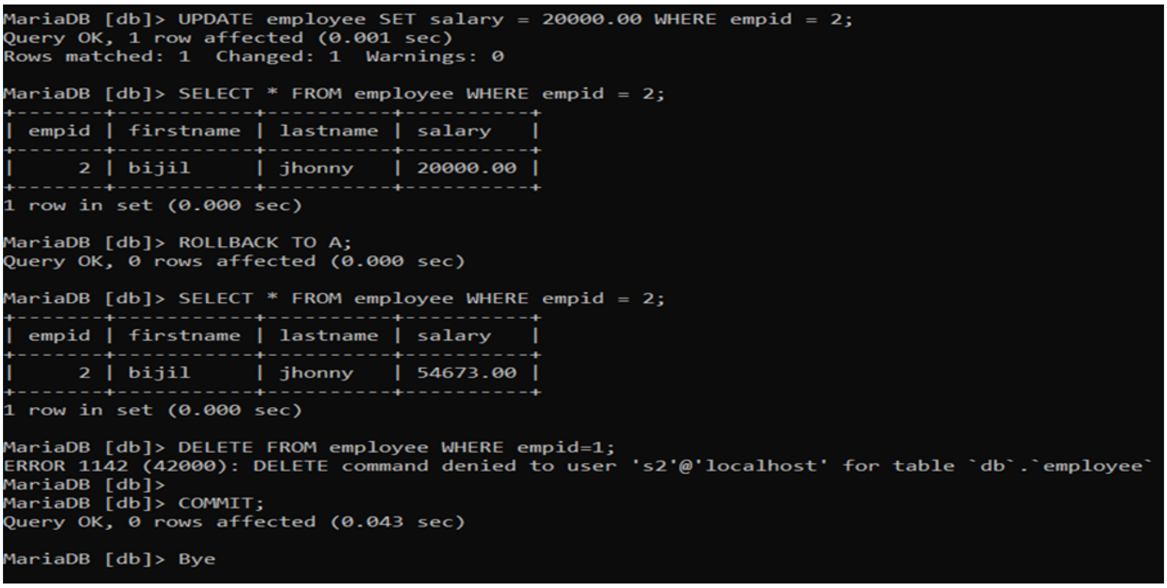
b. check the update record

c. made a mistake,let’s roll back to savepoint ‘A’

d. check bijil’s salary after the rollback (should be unchanged)

e. attempt to delete empid = 1

f. commit the transaction.

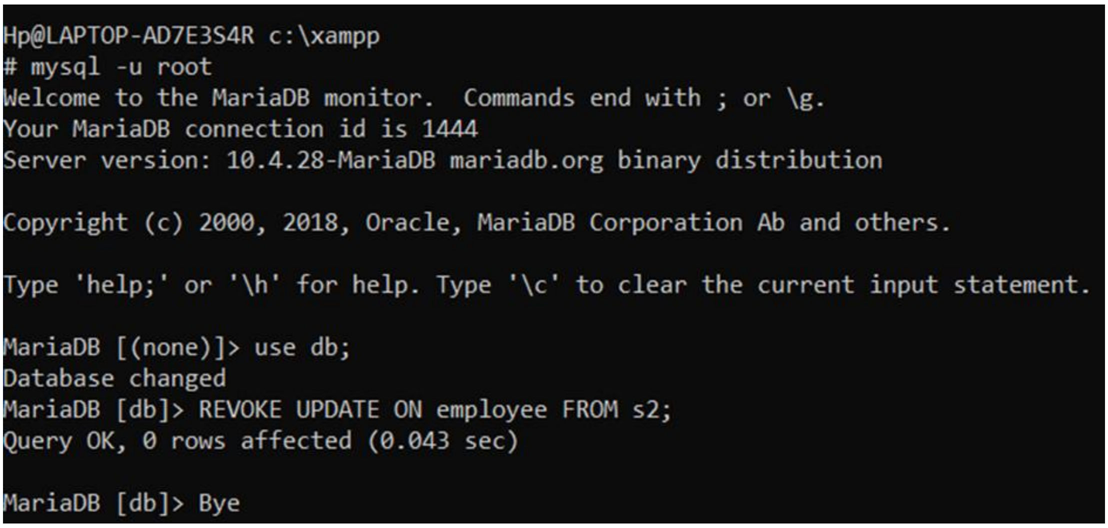


```
MariaDB [db]> UPDATE employee SET salary = 20000.00 WHERE empid = 2;
Query OK, 1 row affected (0.001 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [db]> SELECT * FROM employee WHERE empid = 2;
+-------+-----------+----------+----------+
| empid | firstname | lastname | salary   |
+-------+-----------+----------+----------+
|     2 | bijil     | jhonny   | 20000.00 |
+-------+-----------+----------+----------+
1 row in set (0.000 sec)

MariaDB [db]> ROLLBACK TO A;
Query OK, 0 rows affected (0.000 sec)

MariaDB [db]> SELECT * FROM employee WHERE empid = 2;
+-------+-----------+----------+----------+
| empid | firstname | lastname | salary   |
+-------+-----------+----------+----------+
|     2 | bijil     | jhonny   | 54673.00 |
+-------+-----------+----------+----------+
1 row in set (0.000 sec)

MariaDB [db]> DELETE FROM employee WHERE empid=1;
ERROR 1142 (42000): DELETE command denied to user 's2'@'localhost' for table `db`.`employee`
MariaDB [db]>
MariaDB [db]> COMMIT;
Query OK, 0 rows affected (0.043 sec)

MariaDB [db]> Bye
```

6.Connect as ‘root’ user

a. use database db

b. revoke update privilege on a table from ‘s2’ user

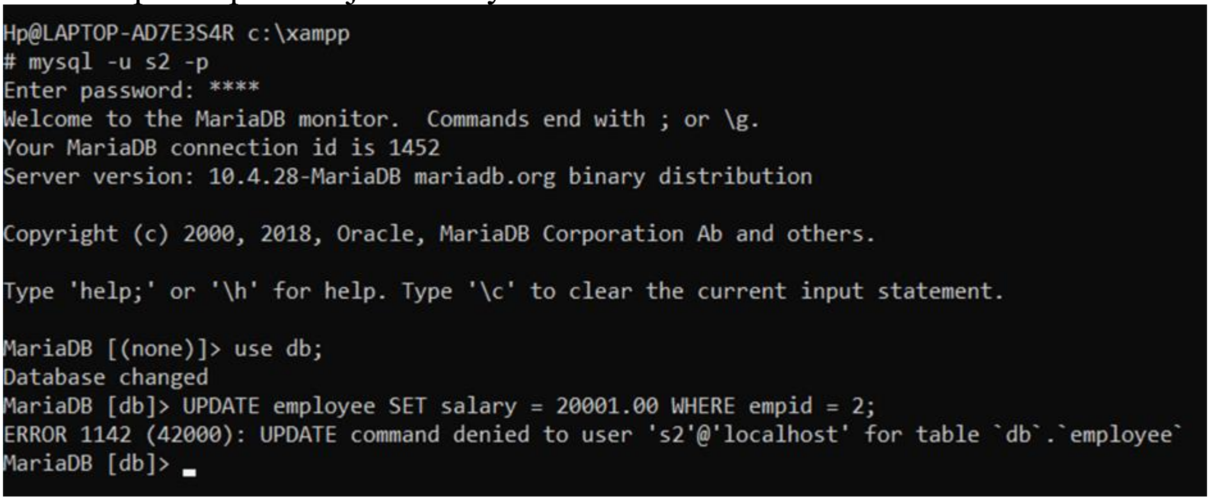


```
Hp@LAPTOP-AD7E3S4R c:\xampp
# mysql -u root
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1444
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use db;
Database changed
MariaDB [db]> REVOKE UPDATE ON employee FROM s2;
Query OK, 0 rows affected (0.043 sec)

MariaDB [db]> Bye
```

7.Connect as ‘s2’ user

a. select database ‘db’

b. attempt to update bijil’s salary

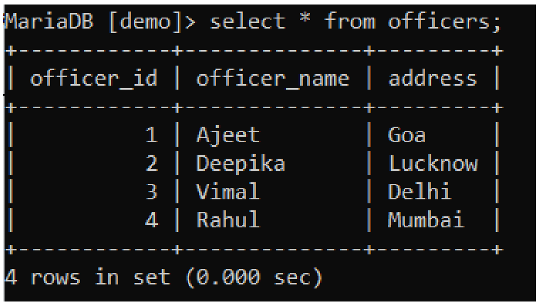


```
Hp@LAPTOP-AD7E3S4R c:\xampp
# mysql -u s2 -p
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 1452
Server version: 10.4.28-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use db;
Database changed
MariaDB [db]> UPDATE employee SET salary = 20001.00 WHERE empid = 2;
ERROR 1142 (42000): UPDATE command denied to user 's2'@'localhost' for table `db`.`employee`
MariaDB [db]> _
```

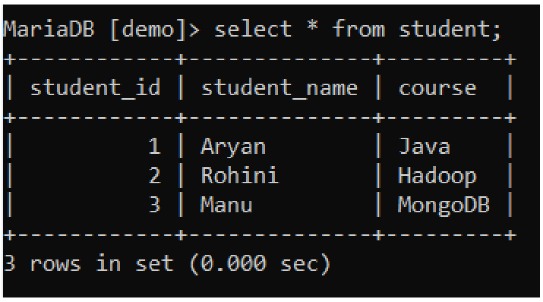
PROGRAM 5

AIM:

Use views and joins for query optimisation in MySQL.



```
MariaDB [demo]> select * from officers;
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          1 | Ajeet        | Goa     |
|          2 | Deepika      | Lucknow |
|          3 | Vimal        | Delhi   |
|          4 | Rahul        | Mumbai  |
+------------+--------------+---------+
4 rows in set (0.000 sec)
```

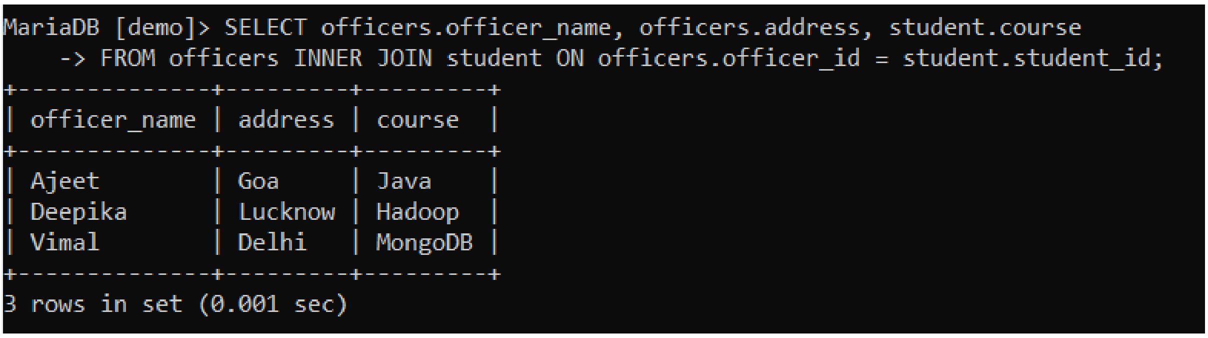


```
MariaDB [demo]> select * from student;
+------------+--------------+---------+
| student_id | student_name | course  |
+------------+--------------+---------+
|          1 | Aryan        | Java    |
|          2 | Rohini       | Hadoop  |
|          3 | Manu         | MongoDB |
+------------+--------------+---------+
3 rows in set (0.000 sec)
```

Join

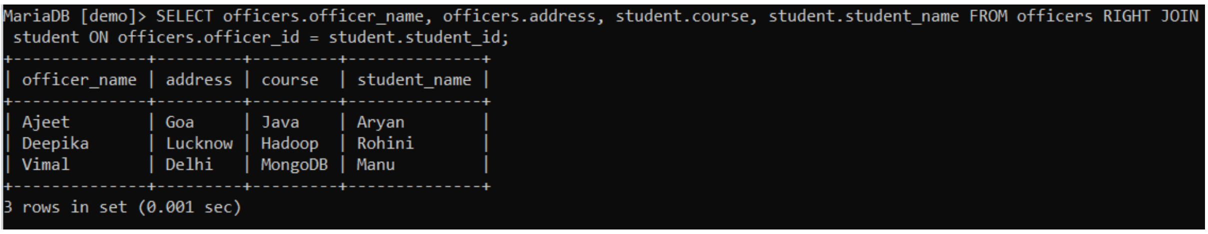
1. SELECT officers.officer_name, officers.address, student.course

   FROM officers INNER JOIN student ON officers.officer_id = student.student_id;



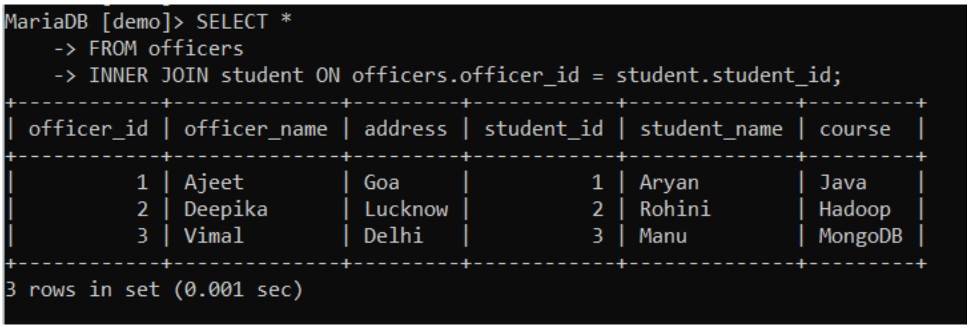
```
MariaDB [demo]> SELECT officers.officer_name, officers.address, student.course
    -> FROM officers INNER JOIN student ON officers.officer_id = student.student_id;
+--------------+---------+---------+
| officer_name | address | course  |
+--------------+---------+---------+
| Ajeet        | Goa     | Java    |
| Deepika      | Lucknow | Hadoop  |
| Vimal        | Delhi   | MongoDB |
+--------------+---------+---------+
3 rows in set (0.001 sec)
```

2. SELECT officers.officer_name, officers.address, student.course, student.student_name FROM officers RIGHT JOIN student ON officers.officer_id = student.student_id;



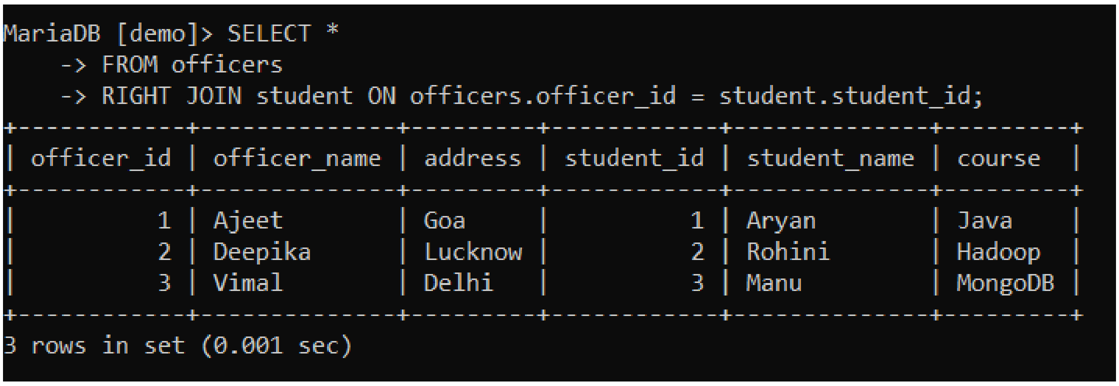
```
MariaDB [demo]> SELECT officers.officer_name, officers.address, student.course, student.student_name FROM officers RIGHT JOIN
 student ON officers.officer_id = student.student_id;
+--------------+---------+---------+--------------+
| officer_name | address | course  | student_name |
+--------------+---------+---------+--------------+
| Ajeet        | Goa     | Java    | Aryan        |
| Deepika      | Lucknow | Hadoop  | Rohini       |
| Vimal        | Delhi   | MongoDB | Manu         |
+--------------+---------+---------+--------------+
3 rows in set (0.001 sec)
```

3. SELECT * FROM officers INNER JOIN student ON officers.officer_id = student.student_id;



```
MariaDB [demo]> SELECT *
    -> FROM officers
    -> INNER JOIN student ON officers.officer_id = student.student_id;
+------------+--------------+---------+------------+--------------+---------+
| officer_id | officer_name | address | student_id | student_name | course  |
+------------+--------------+---------+------------+--------------+---------+
|          1 | Ajeet        | Goa     |          1 | Aryan        | Java    |
|          2 | Deepika      | Lucknow |          2 | Rohini       | Hadoop  |
|          3 | Vimal        | Delhi   |          3 | Manu         | MongoDB |
+------------+--------------+---------+------------+--------------+---------+
3 rows in set (0.001 sec)
```

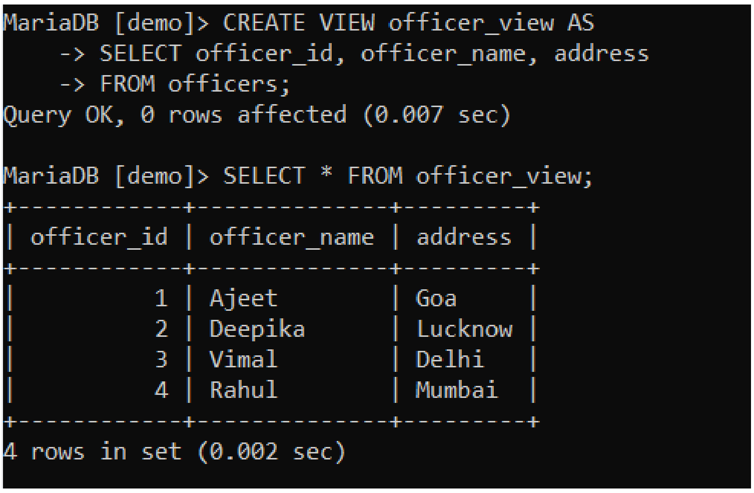
4. SELECT * FROM officers RIGHT JOIN student ON officers.officer_id = student.student_id;



```
MariaDB [demo]> SELECT *
    -> FROM officers
    -> RIGHT JOIN student ON officers.officer_id = student.student_id;
+------------+--------------+---------+------------+--------------+---------+
| officer_id | officer_name | address | student_id | student_name | course  |
+------------+--------------+---------+------------+--------------+---------+
|          1 | Ajeet        | Goa     |          1 | Aryan        | Java    |
|          2 | Deepika      | Lucknow |          2 | Rohini       | Hadoop  |
|          3 | Vimal        | Delhi   |          3 | Manu         | MongoDB |
+------------+--------------+---------+------------+--------------+---------+
3 rows in set (0.001 sec)
```

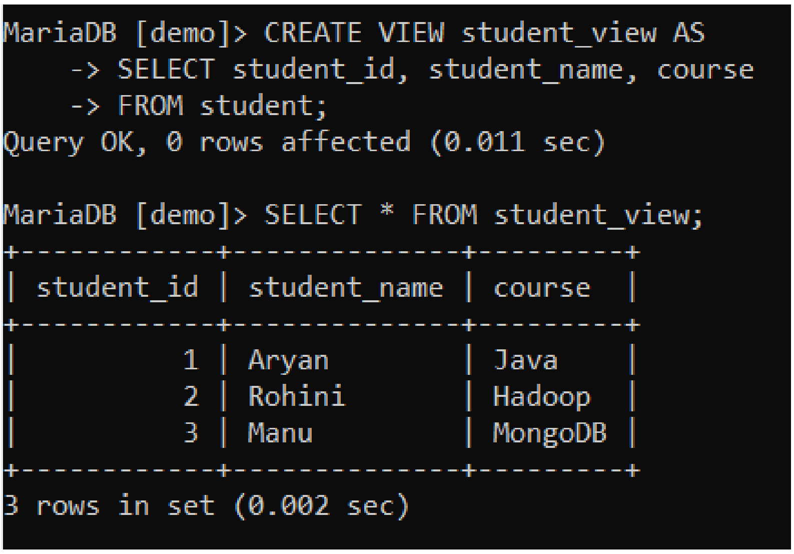
View

1. CREATE VIEW officer_view AS SELECT officer_id, officer_name, address FROM officers;



```
MariaDB [demo]> CREATE VIEW officer_view AS
    -> SELECT officer_id, officer_name, address
    -> FROM officers;
Query OK, 0 rows affected (0.007 sec)

MariaDB [demo]> SELECT * FROM officer_view;
+------------+--------------+---------+
| officer_id | officer_name | address |
+------------+--------------+---------+
|          1 | Ajeet        | Goa     |
|          2 | Deepika      | Lucknow |
|          3 | Vimal        | Delhi   |
|          4 | Rahul        | Mumbai  |
+------------+--------------+---------+
4 rows in set (0.002 sec)
```

2. CREATE VIEW student_view AS SELECT student_id, student_name, course FROM student;



```
MariaDB [demo]> CREATE VIEW student_view AS
    -> SELECT student_id, student_name, course
    -> FROM student;
Query OK, 0 rows affected (0.011 sec)

MariaDB [demo]> SELECT * FROM student_view;
+------------+--------------+---------+
| student_id | student_name | course  |
+------------+--------------+---------+
|          1 | Aryan        | Java    |
|          2 | Rohini       | Hadoop  |
|          3 | Manu         | MongoDB |
+------------+--------------+---------+
3 rows in set (0.002 sec)
```

CO2

Apply PL/SQL for processing databases.

PROGRAM 6.1

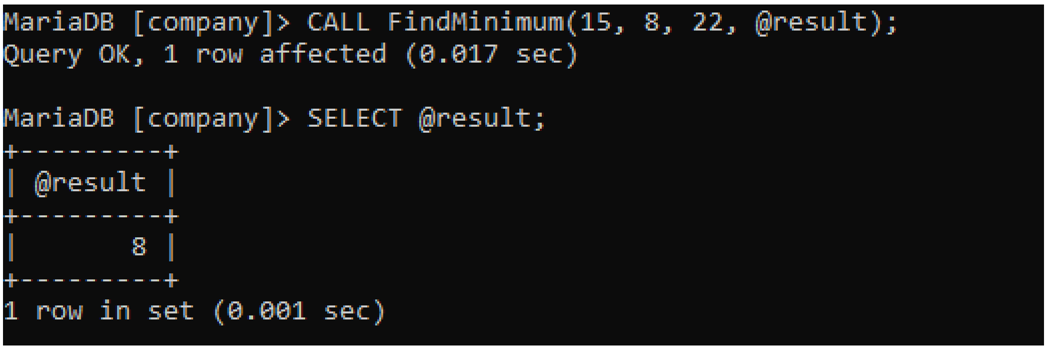
AIM:

Create a procedure to find the minimum of 3 numbers.(with Parameter)

CODE:

```
DELIMITER //

CREATE PROCEDURE FindMinimum(IN val1 INT, IN val2 INT, IN val3
INT, OUT minValue INT)
BEGIN
   SELECT LEAST(val1, val2, val3) INTO minValue;
END //

DELIMITER ;

CALL FindMinimum(15, 8, 22, @result);
```

OUTPUT:



```
MariaDB [company]> CALL FindMinimum(15, 8, 22, @result);
Query OK, 1 row affected (0.017 sec)

MariaDB [company]> SELECT @result;
+---------+
| @result |
+---------+
|       8 |
+---------+
1 row in set (0.001 sec)
```

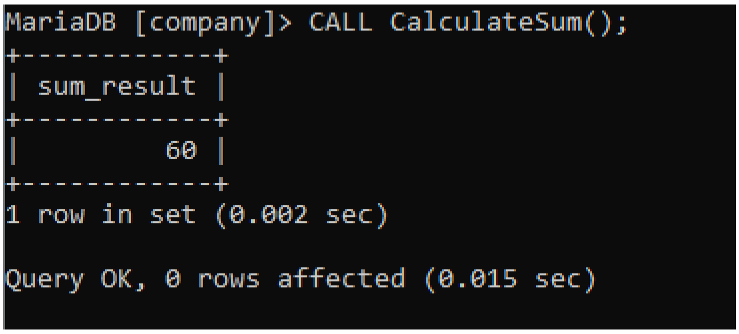
PROGRAM 6.2

AIM: Create a procedure to find sum of three numbers.(Without parameters).

CODE:

```
DELIMITER //

CREATE PROCEDURE CalculateSum()
BEGIN
   DECLARE num1 INT;
   DECLARE num2 INT;
   DECLARE num3 INT;
   DECLARE total INT;

   SET num1 = 10;
   SET num2 = 20;
   SET num3 = 30;

   SET total = num1 + num2 + num3;

   SELECT total AS sum_result;
END //

DELIMITER ;
```

OUTPUT:



```
MariaDB [company]> CALL CalculateSum();
+------------+
| sum_result |
+------------+
|         60 |
+------------+
1 row in set (0.002 sec)

Query OK, 0 rows affected (0.015 sec)
```

PROGRAM :7

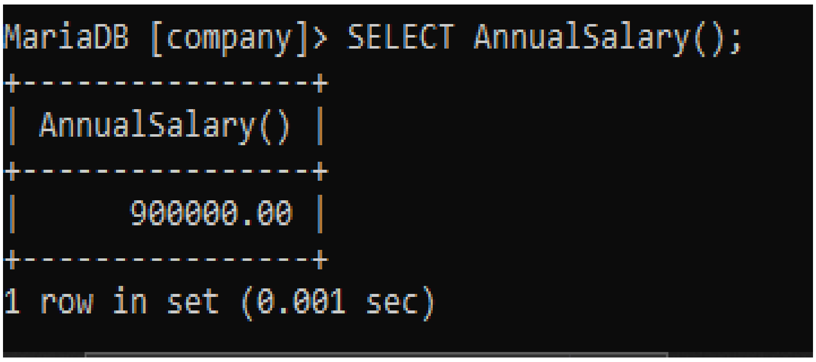
AIM:

Create a function to print annual salary of the employees in HR department.

PROGRAM:

```
DELIMITER //

CREATE OR REPLACE FUNCTION AnnualSalary() RETURNS DECIMAL(10, 2)
BEGIN
   DECLARE total_salary DECIMAL(10, 2);
   SELECT SUM(e.Salary * 12) INTO total_salary
   FROM employees e
   INNER JOIN departments d ON e.Department_ID = d.DepartmentID
   WHERE d.DepartmentName = 'HR';

   RETURN total_salary;
END;

//

DELIMITER ;
```

OUTPUT:



```
MariaDB [company]> SELECT AnnualSalary();
+----------------+
| AnnualSalary() |
+----------------+
|      900000.00 |
+----------------+
1 row in set (0.001 sec)
```

PROGRAM :8

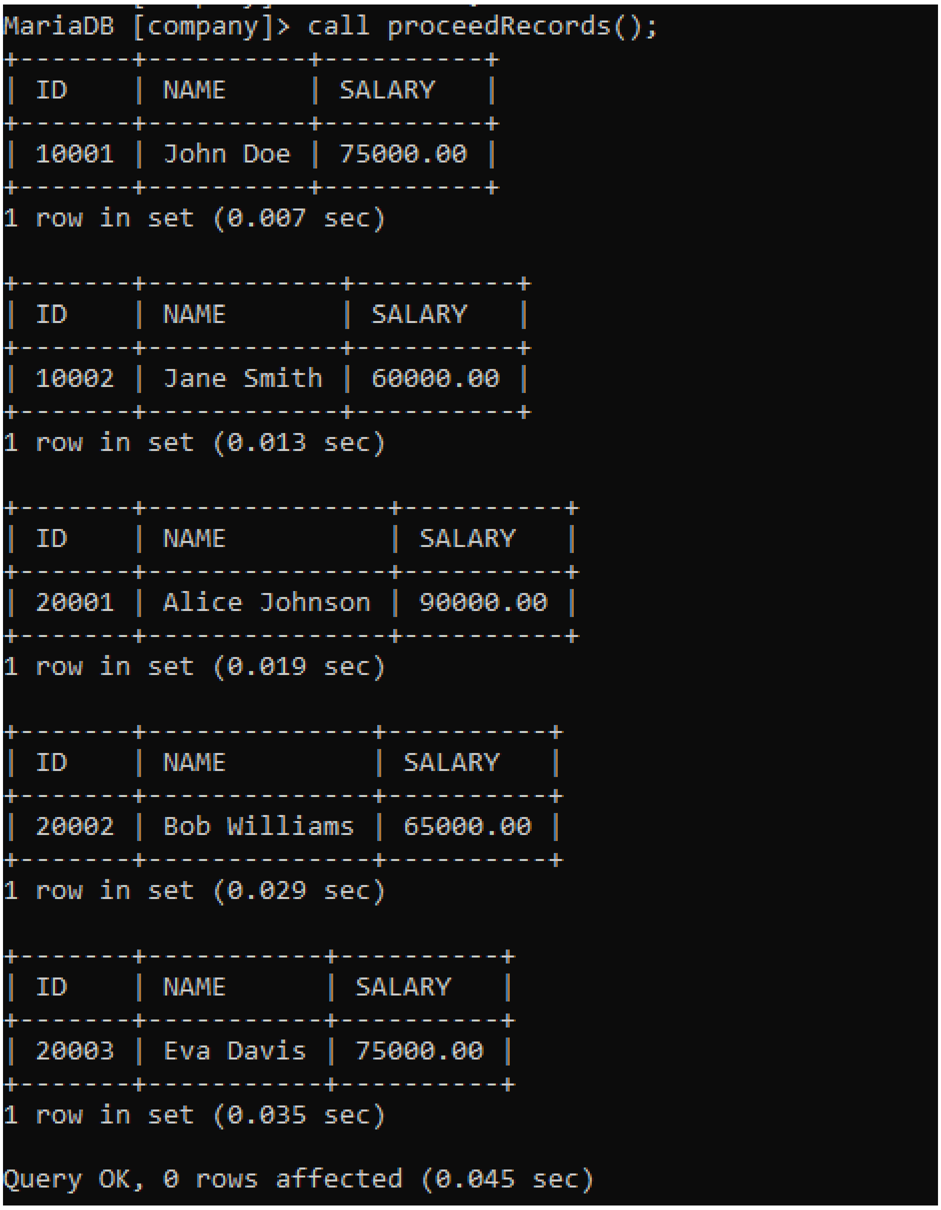
AIM:

Create a cursor to print employee name of employees whose salary is greater than 50000

PROGRAM:

```
DELIMITER //
CREATE OR REPLACE PROCEDURE proceedRecords()
BEGIN
   DECLARE e_id INT(5);
   DECLARE e_name VARCHAR(25);
   DECLARE e_sal FLOAT(8, 2);
   DECLARE done INT DEFAULT FALSE;

 DECLARE c_employees CURSOR FOR SELECT employeeID,
CONCAT(FirstName,” “,LastName) AS ename, salary FROM employees;
 DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

   OPEN c_employees;

   read_loop: LOOP
      FETCH c_employees INTO e_id, e_name, e_sal;
      IF done THEN
         LEAVE read_loop;
      END IF;

                  SELECT e_id AS ID, e_name AS NAME,e_sal AS SALARY;

   END LOOP read_loop;

   CLOSE c_employees;
END//

DELIMITER ;
```

OUTPUT:



```
MariaDB [company]> call proceedRecords();
+-------+----------+----------+
| ID    | NAME     | SALARY   |
+-------+----------+----------+
| 10001 | John Doe | 75000.00 |
+-------+----------+----------+
1 row in set (0.007 sec)

+-------+------------+----------+
| ID    | NAME       | SALARY   |
+-------+------------+----------+
| 10002 | Jane Smith | 60000.00 |
+-------+------------+----------+
1 row in set (0.013 sec)

+-------+---------------+----------+
| ID    | NAME          | SALARY   |
+-------+---------------+----------+
| 20001 | Alice Johnson | 90000.00 |
+-------+---------------+----------+
1 row in set (0.019 sec)

+-------+--------------+----------+
| ID    | NAME         | SALARY   |
+-------+--------------+----------+
| 20002 | Bob Williams | 65000.00 |
+-------+--------------+----------+
1 row in set (0.029 sec)

+-------+-----------+----------+
| ID    | NAME      | SALARY   |
+-------+-----------+----------+
| 20003 | Eva Davis | 75000.00 |
+-------+-----------+----------+
1 row in set (0.035 sec)

Query OK, 0 rows affected (0.045 sec)
```

PROGRAM :9

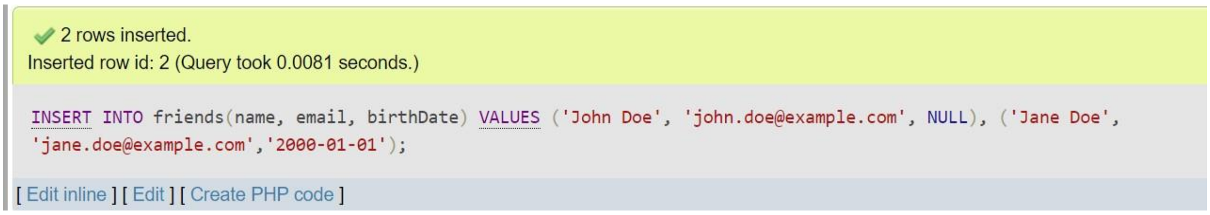
AIM: Construct a Trigger code for a table in database

PROGRAM:

```
DROP TABLE IF EXISTS friends;

CREATE TABLE friends (
   id INT AUTO_INCREMENT,
   name VARCHAR(100) NOT NULL,
   email VARCHAR(255),
   birthDate DATE,
   PRIMARY KEY (id)
);
DROP TABLE IF EXISTS reminders;

CREATE TABLE reminders (
   id INT AUTO_INCREMENT,
   memberId INT,
   message VARCHAR(255) NOT NULL,
   PRIMARY KEY (id , memberId)
);



DELIMITER $$

CREATE TRIGGER after_insert_Trigger
AFTER INSERT
ON friends FOR EACH ROW
BEGIN
   IF NEW.birthDate IS NULL THEN
      INSERT INTO reminders(memberId, message)
      VALUES(new.id,CONCAT('Hi ', NEW.name, ', please update your date of
birth.'));
   END IF;
END$$
```

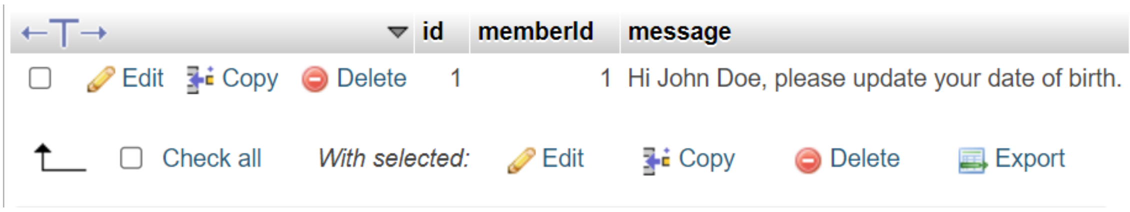
```
DELIMITER ;

INSERT INTO friends(name, email, birthDate)
VALUES
    ('John Doe', 'john.doe@example.com', NULL),
    ('Jane Doe', 'jane.doe@example.com','2000-01-01');

SELECT * FROM reminders;
```

OUTPUT:



✓ 2 rows inserted.
Inserted row id: 2 (Query took 0.0081 seconds.)

```
INSERT INTO friends(name, email, birthDate) VALUES ('John Doe', 'john.doe@example.com', NULL), ('Jane Doe',
'jane.doe@example.com','2000-01-01');
```



id	memberId	message
1	1	Hi John Doe, please update your date of birth.

CO3

Comparison between relational and non-relational (NoSQL) databases and the configuration of NoSQL Databases.

PROGRAM :10

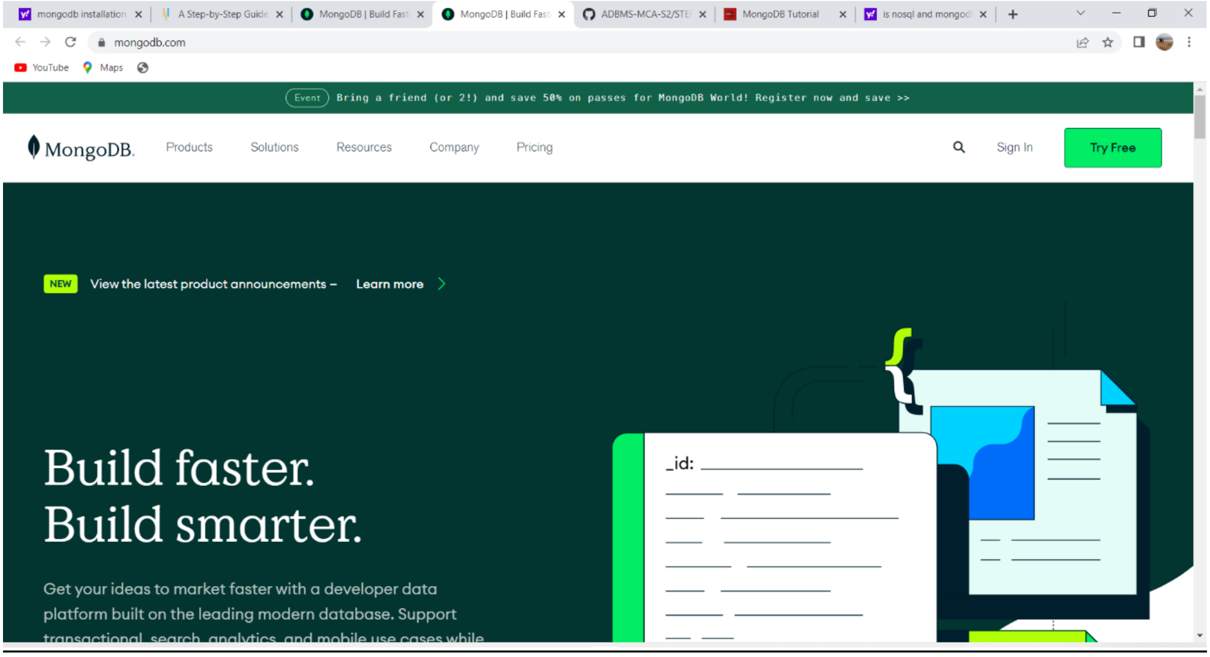
AIM:

Installation and configuration of NoSQL database - MongoDB

MongoDB is a cross-platform, document oriented NoSql database that provides, high performance, high availability, and easy scalability. MongoDB works on concept of collection and document.

STEP 1:

Navigate to the official MongoDB website.



Event Bring a friend (or 2!) and save 50% on passes for MongoDB World! Register now and save >>

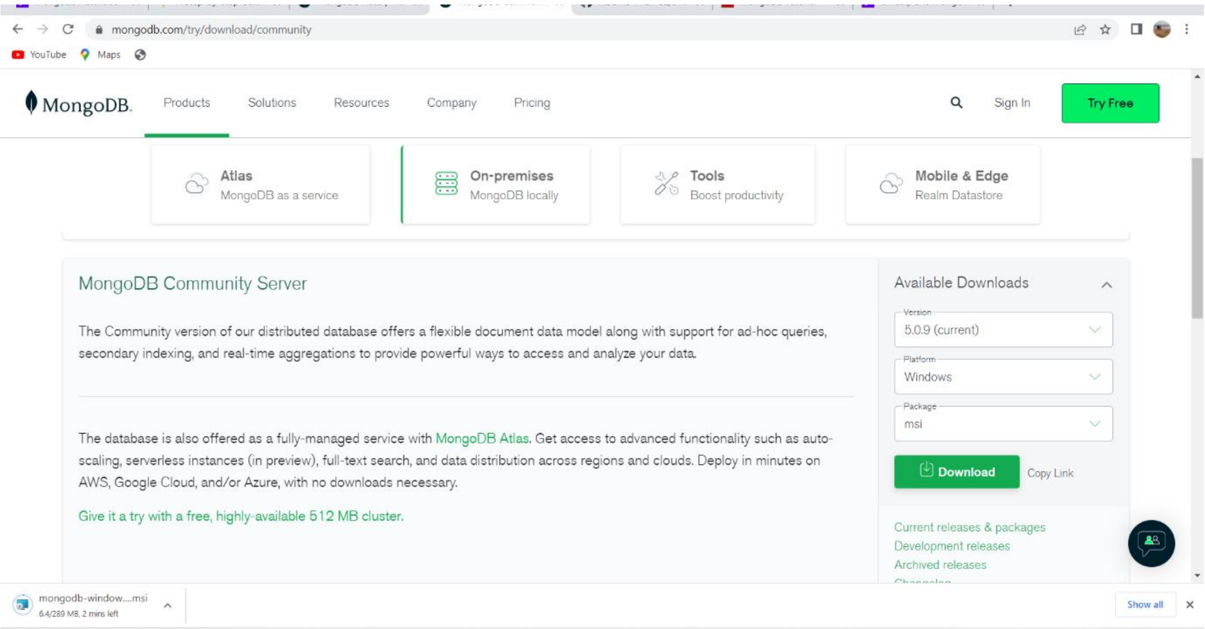
NEW View the latest product announcements – Learn more

Build faster.
Build smarter.

Get your ideas to market faster with a developer data platform built on the leading modern database. Support transactional, search, analytics, and mobile use cases while

STEP 2:

Under the products section, click on the Community server version. Make sure that the specifications to the right of the screen are correct. At the time of writing, the latest version is 4.4.5. Ensure that the platform is Windows, and the package is MSI. Go ahead and click on download.



mongodb.com/try/download/community

MongoDB Community Server

The Community version of our distributed database offers a flexible document data model along with support for ad-hoc queries, secondary indexing, and real-time aggregations to provide powerful ways to access and analyze your data.

The database is also offered as a fully-managed service with MongoDB Atlas. Get access to advanced functionality such as auto-scaling, serverless instances (in preview), full-text search, and data distribution across regions and clouds. Deploy in minutes on AWS, Google Cloud, and/or Azure, with no downloads necessary.

Give it a try with a free, highly-available 512 MB cluster.

Available Downloads

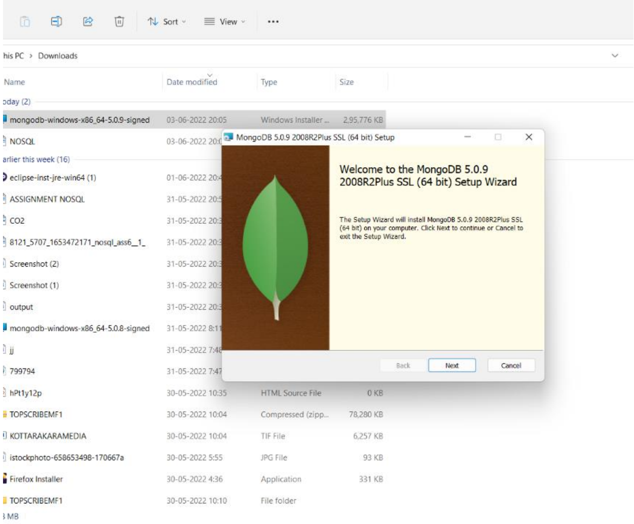
Version: 5.0.9 (current)

Platform: Windows

Package: msi

STEP 3:

You can find the downloaded file in the downloads directory. Install the software step by step.



his PC › Downloads

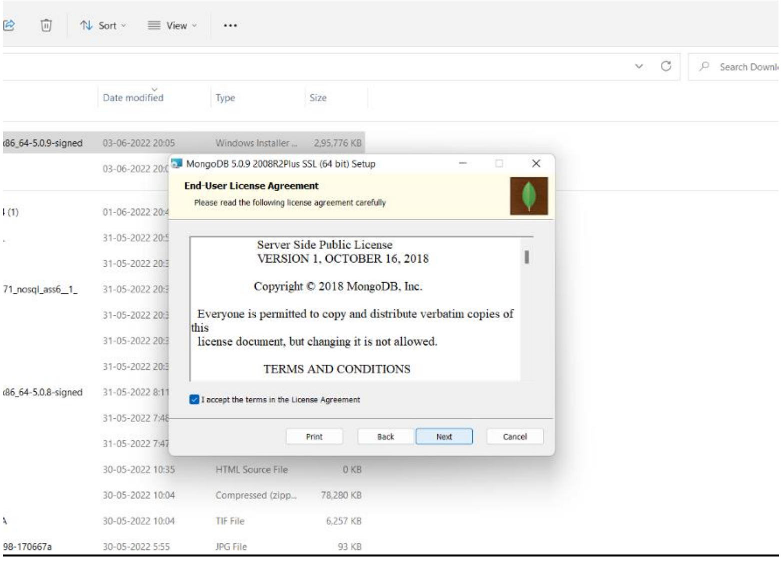
Name	Date modified	Type	Size
oday (2)			
mongodb-windows-x86_64-5.0.9-signed	03-06-2022 20:05	Windows Installer ...	2,95,776 KB
NOSQL	03-06-2022 20:(		
arlier this week (16)			
eclipse-inst-jre-win64 (1)	01-06-2022 20:4		
ASSIGNMENT NOSQL	31-05-2022 20:5		
CO2	31-05-2022 20:3		
8121_5707_1653472171_nosql_ass6__1_	31-05-2022 20:3		
Screenshot (2)	31-05-2022 20:3		
Screenshot (1)	31-05-2022 20:3		
output	31-05-2022 20:3		
mongodb-windows-x86_64-5.0.8-signed	31-05-2022 8:11		
jj	31-05-2022 7:48		
799794	31-05-2022 7:47		
hPt1y12p	30-05-2022 10:35	HTML Source File	0 KB
TOPSCRIBEMF1	30-05-2022 10:04	Compressed (zipp...	78,280 KB
KOTTARAKARAMEDIA	30-05-2022 10:04	TIF File	6,257 KB
istockphoto-658653498-170667a	30-05-2022 5:55	JPG File	93 KB
Firefox Installer	30-05-2022 4:36	Application	331 KB
TOPSCRIBEMF1	30-05-2022 10:10	File folder	

3 MB

MongoDB 5.0.9 2008R2Plus SSL (64 bit) Setup

Welcome to the MongoDB 5.0.9 2008R2Plus SSL (64 bit) Setup Wizard

The Setup Wizard will install MongoDB 5.0.9 2008R2Plus SSL (64 bit) on your computer. Click Next to continue or Cancel to exit the Setup Wizard.



MongoDB 5.0.9 2008R2Plus SSL (64 bit) Setup

End-User License Agreement

Please read the following license agreement carefully

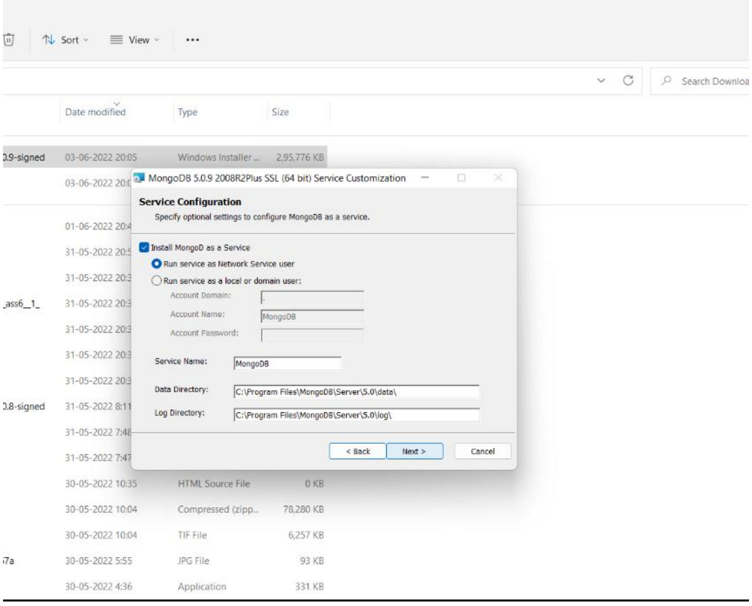
Server Side Public License
VERSION 1, OCTOBER 16, 2018

Copyright © 2018 MongoDB, Inc.

Everyone is permitted to copy and distribute verbatim copies of this
license document, but changing it is not allowed.

TERMS AND CONDITIONS

☑ I accept the terms in the License Agreement



MongoDB 5.0.9 2008R2Plus SSL (64 bit) Service Customization

Service Configuration

Specify optional settings to configure MongoDB as a service.

☑ Install MongoD as a Service

● Run service as Network Service user

○ Run service as a local or domain user:

Account Domain: .

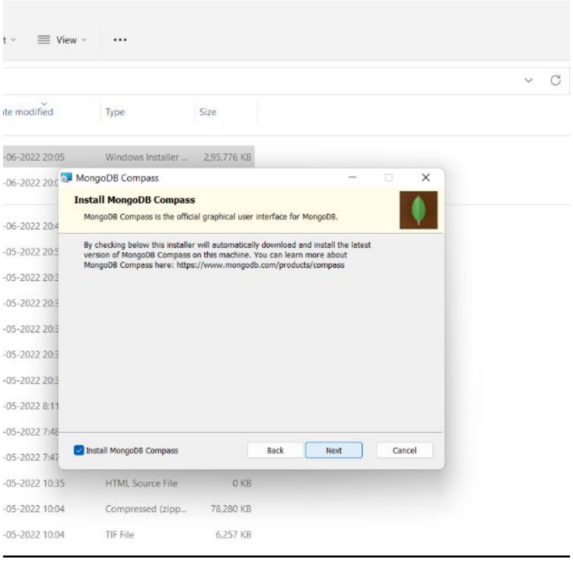
Account Name: MongoDB

Account Password:

Service Name: MongoDB

Data Directory: C:\Program Files\MongoDB\Server\5.0\data\

Log Directory: C:\Program Files\MongoDB\Server\5.0\log\



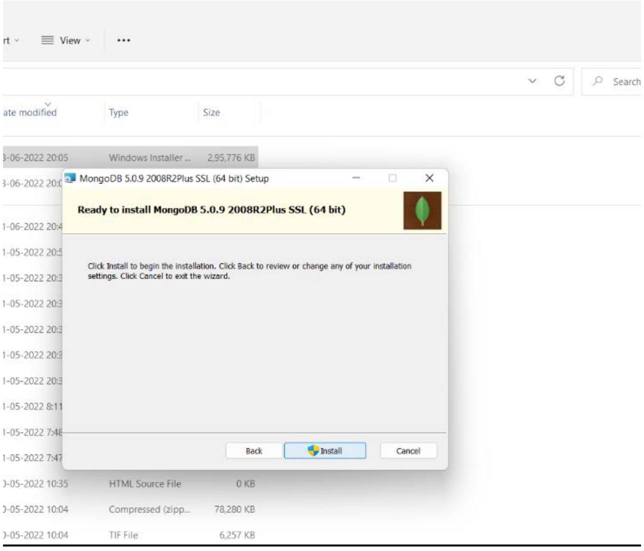
MongoDB Compass

Install MongoDB Compass

MongoDB Compass is the official graphical user interface for MongoDB.

By checking below this installer will automatically download and install the latest version of MongoDB Compass on this machine. You can learn more about MongoDB Compass here: https://www.mongodb.com/products/compass

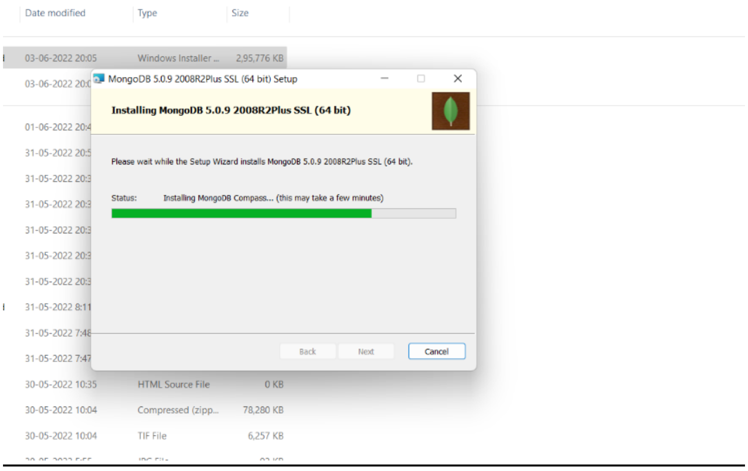
☑ Install MongoDB Compass



MongoDB 5.0.9 2008R2Plus SSL (64 bit) Setup

Ready to install MongoDB 5.0.9 2008R2Plus SSL (64 bit)

Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.



MongoDB 5.0.9 2008R2Plus SSL (64 bit) Setup

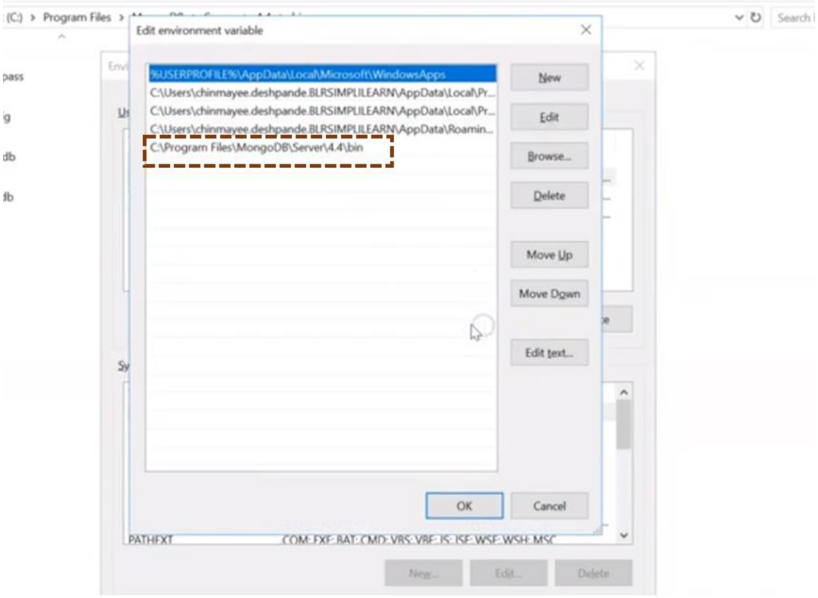
Installing MongoDB 5.0.9 2008R2Plus SSL (64 bit)

Please wait while the Setup Wizard installs MongoDB 5.0.9 2008R2Plus SSL (64 bit).

Status: Installing MongoDB Compass... (this may take a few minutes)

STEP:4

create an environment variable for the executable file so that we don’t have to change the directory structure every time we want to execute the file.

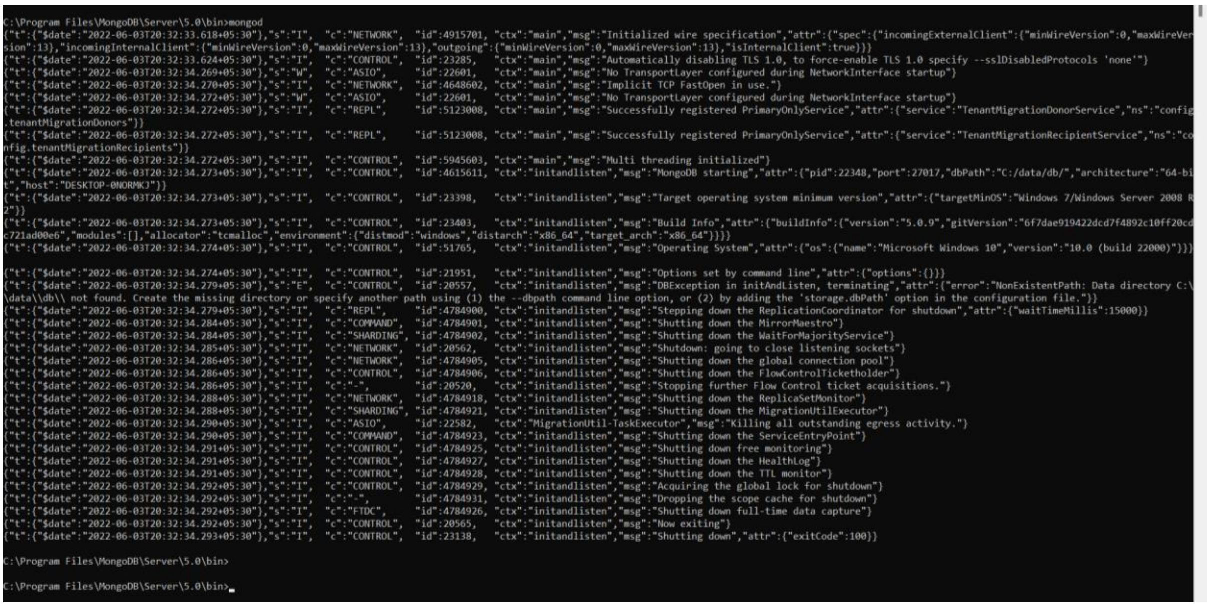


%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\chinmayee.deshpande.BLRSIMPLILEARN\AppData\Local\Pr...
C:\Users\chinmayee.deshpande.BLRSIMPLILEARN\AppData\Local\Pr...
C:\Users\chinmayee.deshpande.BLRSIMPLILEARN\AppData\Roamin...
C:\Program Files\MongoDB\Server\4.4\bin

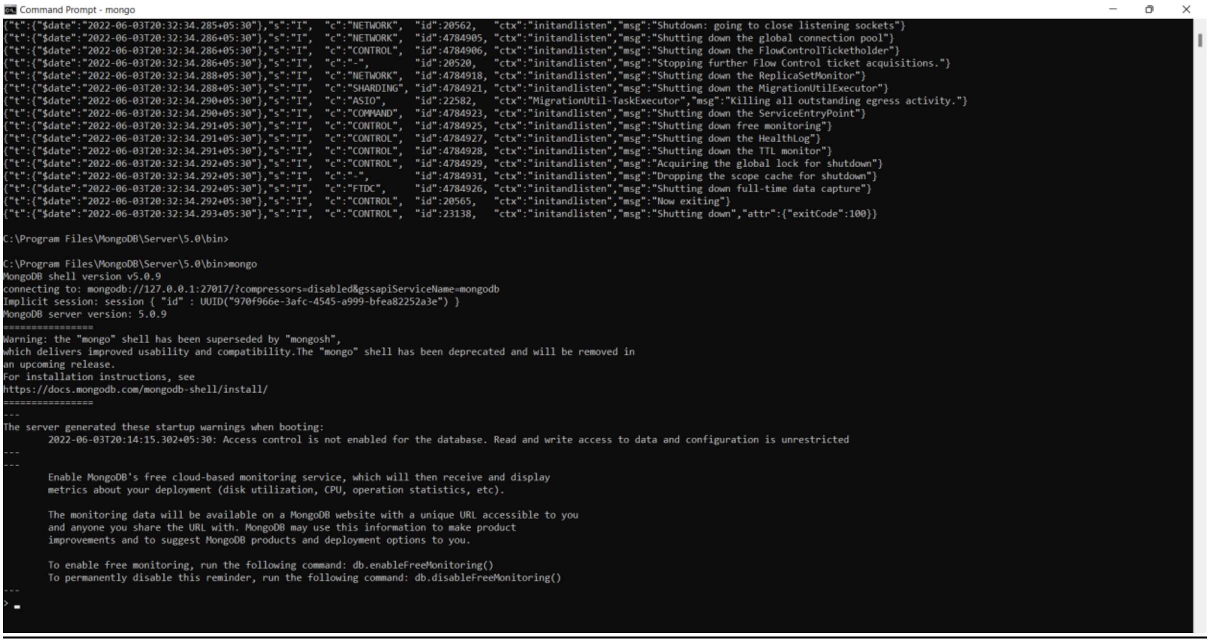
PATHEXT .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC

STEP:5

After creating an environment path, download mongosh and install. we can open the command prompt and type mongod. An instance of mongodb server is started. Now take another terminal and type mongosh. This creates a client instance of mongodb in your local system.



```
C:\Program Files\MongoDB\Server\5.0\bin>mongod
{"t":{"$date":"2022-06-03T20:32:33.618+05:30"},"s":"I",  "c":"NETWORK",  "id":4915701, "ctx":"main","msg":"Initialized wire specification","attr":{"spec":{"incomingExternalClient":{"minWireVersion":0,"maxWireVer
sion":13},"incomingInternalClient":{"minWireVersion":0,"maxWireVersion":13},"outgoing":{"minWireVersion":0,"maxWireVersion":13},"isInternalClient":true}}}
{"t":{"$date":"2022-06-03T20:32:33.624+05:30"},"s":"I",  "c":"CONTROL",  "id":23285,   "ctx":"main","msg":"Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'"}
{"t":{"$date":"2022-06-03T20:32:34.269+05:30"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2022-06-03T20:32:34.270+05:30"},"s":"I",  "c":"NETWORK",  "id":4648602, "ctx":"main","msg":"Implicit TCP FastOpen in use."}
{"t":{"$date":"2022-06-03T20:32:34.272+05:30"},"s":"W",  "c":"ASIO",     "id":22601,   "ctx":"main","msg":"No TransportLayer configured during NetworkInterface startup"}
{"t":{"$date":"2022-06-03T20:32:34.272+05:30"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"service":"TenantMigrationDonorService","ns":"config
.tenantMigrationDonors"}}
{"t":{"$date":"2022-06-03T20:32:34.272+05:30"},"s":"I",  "c":"REPL",     "id":5123008, "ctx":"main","msg":"Successfully registered PrimaryOnlyService","attr":{"service":"TenantMigrationRecipientService","ns":"co
nfig.tenantMigrationRecipients"}}
{"t":{"$date":"2022-06-03T20:32:34.272+05:30"},"s":"I",  "c":"CONTROL",  "id":5945603, "ctx":"main","msg":"Multi threading initialized"}
{"t":{"$date":"2022-06-03T20:32:34.273+05:30"},"s":"I",  "c":"CONTROL",  "id":4615611, "ctx":"initandlisten","msg":"MongoDB starting","attr":{"pid":22348,"port":27017,"dbPath":"C:/data/db/","architecture":"64-bi
t","host":"DESKTOP-0NORMKJ"}}
{"t":{"$date":"2022-06-03T20:32:34.273+05:30"},"s":"I",  "c":"CONTROL",  "id":23398,   "ctx":"initandlisten","msg":"Target operating system minimum version","attr":{"targetMinOS":"Windows 7/Windows Server 2008 R
2"}}
{"t":{"$date":"2022-06-03T20:32:34.273+05:30"},"s":"I",  "c":"CONTROL",  "id":23403,   "ctx":"initandlisten","msg":"Build Info","attr":{"buildInfo":{"version":"5.0.9","gitVersion":"6f7dae919422dcd7f4892c10ff20cd
c721ad00e6","modules":[],"allocator":"tcmalloc","environment":{"distmod":"windows","distarch":"x86_64","target_arch":"x86_64"}}}}
{"t":{"$date":"2022-06-03T20:32:34.274+05:30"},"s":"I",  "c":"CONTROL",  "id":51765,   "ctx":"initandlisten","msg":"Operating System","attr":{"os":{"name":"Microsoft Windows 10","version":"10.0 (build 22000)"}}}

{"t":{"$date":"2022-06-03T20:32:34.274+05:30"},"s":"I",  "c":"CONTROL",  "id":21951,   "ctx":"initandlisten","msg":"Options set by command line","attr":{"options":{}}}
{"t":{"$date":"2022-06-03T20:32:34.279+05:30"},"s":"E",  "c":"CONTROL",  "id":20557,   "ctx":"initandlisten","msg":"DBException in initAndListen, terminating","attr":{"error":"NonExistentPath: Data directory C:\
\data\\db\\ not found. Create the missing directory or specify another path using (1) the --dbpath command line option, or (2) by adding the 'storage.dbPath' option in the configuration file."}}
{"t":{"$date":"2022-06-03T20:32:34.279+05:30"},"s":"I",  "c":"REPL",     "id":4784900, "ctx":"initandlisten","msg":"Stepping down the ReplicationCoordinator for shutdown","attr":{"waitTimeMillis":15000}}
{"t":{"$date":"2022-06-03T20:32:34.284+05:30"},"s":"I",  "c":"COMMAND",  "id":4784901, "ctx":"initandlisten","msg":"Shutting down the MirrorMaestro"}
{"t":{"$date":"2022-06-03T20:32:34.284+05:30"},"s":"I",  "c":"SHARDING", "id":4784902, "ctx":"initandlisten","msg":"Shutting down the WaitForMajorityService"}
{"t":{"$date":"2022-06-03T20:32:34.285+05:30"},"s":"I",  "c":"NETWORK",  "id":20562,   "ctx":"initandlisten","msg":"Shutdown: going to close listening sockets"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I",  "c":"NETWORK",  "id":4784905, "ctx":"initandlisten","msg":"Shutting down the global connection pool"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I",  "c":"CONTROL",  "id":4784906, "ctx":"initandlisten","msg":"Shutting down the FlowControlTicketholder"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I",  "c":"-",        "id":20520,   "ctx":"initandlisten","msg":"Stopping further Flow Control ticket acquisitions."}
{"t":{"$date":"2022-06-03T20:32:34.288+05:30"},"s":"I",  "c":"NETWORK",  "id":4784918, "ctx":"initandlisten","msg":"Shutting down the ReplicaSetMonitor"}
{"t":{"$date":"2022-06-03T20:32:34.288+05:30"},"s":"I",  "c":"SHARDING", "id":4784921, "ctx":"initandlisten","msg":"Shutting down the MigrationUtilExecutor"}
{"t":{"$date":"2022-06-03T20:32:34.290+05:30"},"s":"I",  "c":"ASIO",     "id":22582,   "ctx":"MigrationUtil-TaskExecutor","msg":"Killing all outstanding egress activity."}
{"t":{"$date":"2022-06-03T20:32:34.290+05:30"},"s":"I",  "c":"COMMAND",  "id":4784923, "ctx":"initandlisten","msg":"Shutting down the ServiceEntryPoint"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I",  "c":"CONTROL",  "id":4784925, "ctx":"initandlisten","msg":"Shutting down free monitoring"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I",  "c":"CONTROL",  "id":4784927, "ctx":"initandlisten","msg":"Shutting down the HealthLog"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I",  "c":"CONTROL",  "id":4784928, "ctx":"initandlisten","msg":"Shutting down the TTL monitor"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"CONTROL",  "id":4784929, "ctx":"initandlisten","msg":"Acquiring the global lock for shutdown"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"-",        "id":4784931, "ctx":"initandlisten","msg":"Dropping the scope cache for shutdown"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"FTDC",     "id":4784926, "ctx":"initandlisten","msg":"Shutting down full-time data capture"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"CONTROL",  "id":20565,   "ctx":"initandlisten","msg":"Now exiting"}
{"t":{"$date":"2022-06-03T20:32:34.293+05:30"},"s":"I",  "c":"CONTROL",  "id":23138,   "ctx":"initandlisten","msg":"Shutting down","attr":{"exitCode":100}}

C:\Program Files\MongoDB\Server\5.0\bin>

C:\Program Files\MongoDB\Server\5.0\bin>
```



```
{"t":{"$date":"2022-06-03T20:32:34.285+05:30"},"s":"I",  "c":"NETWORK",  "id":20562,   "ctx":"initandlisten","msg":"Shutdown: going to close listening sockets"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I",  "c":"NETWORK",  "id":4784905, "ctx":"initandlisten","msg":"Shutting down the global connection pool"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I",  "c":"CONTROL",  "id":4784906, "ctx":"initandlisten","msg":"Shutting down the FlowControlTicketholder"}
{"t":{"$date":"2022-06-03T20:32:34.286+05:30"},"s":"I",  "c":"-",        "id":20520,   "ctx":"initandlisten","msg":"Stopping further Flow Control ticket acquisitions."}
{"t":{"$date":"2022-06-03T20:32:34.288+05:30"},"s":"I",  "c":"NETWORK",  "id":4784918, "ctx":"initandlisten","msg":"Shutting down the ReplicaSetMonitor"}
{"t":{"$date":"2022-06-03T20:32:34.288+05:30"},"s":"I",  "c":"SHARDING", "id":4784921, "ctx":"initandlisten","msg":"Shutting down the MigrationUtilExecutor"}
{"t":{"$date":"2022-06-03T20:32:34.290+05:30"},"s":"I",  "c":"ASIO",     "id":22582,   "ctx":"MigrationUtil-TaskExecutor","msg":"Killing all outstanding egress activity."}
{"t":{"$date":"2022-06-03T20:32:34.290+05:30"},"s":"I",  "c":"COMMAND",  "id":4784923, "ctx":"initandlisten","msg":"Shutting down the ServiceEntryPoint"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I",  "c":"CONTROL",  "id":4784925, "ctx":"initandlisten","msg":"Shutting down free monitoring"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I",  "c":"CONTROL",  "id":4784927, "ctx":"initandlisten","msg":"Shutting down the HealthLog"}
{"t":{"$date":"2022-06-03T20:32:34.291+05:30"},"s":"I",  "c":"CONTROL",  "id":4784928, "ctx":"initandlisten","msg":"Shutting down the TTL monitor"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"CONTROL",  "id":4784929, "ctx":"initandlisten","msg":"Acquiring the global lock for shutdown"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"-",        "id":4784931, "ctx":"initandlisten","msg":"Dropping the scope cache for shutdown"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"FTDC",     "id":4784926, "ctx":"initandlisten","msg":"Shutting down full-time data capture"}
{"t":{"$date":"2022-06-03T20:32:34.292+05:30"},"s":"I",  "c":"CONTROL",  "id":20565,   "ctx":"initandlisten","msg":"Now exiting"}
{"t":{"$date":"2022-06-03T20:32:34.293+05:30"},"s":"I",  "c":"CONTROL",  "id":23138,   "ctx":"initandlisten","msg":"Shutting down","attr":{"exitCode":100}}

C:\Program Files\MongoDB\Server\5.0\bin>

C:\Program Files\MongoDB\Server\5.0\bin>mongo
MongoDB shell version v5.0.9
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("970f966e-3afc-4545-a999-bfea82252a3e") }
MongoDB server version: 5.0.9
================
Warning: the "mongo" shell has been superseded by "mongosh",
which delivers improved usability and compatibility.The "mongo" shell has been deprecated and will be removed in
an upcoming release.
For installation instructions, see
https://docs.mongodb.com/mongodb-shell/install/
================
---
The server generated these startup warnings when booting:
        2022-06-03T20:14:15.302+05:30: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
        Enable MongoDB's free cloud-based monitoring service, which will then receive and display
        metrics about your deployment (disk utilization, CPU, operation statistics, etc).

        The monitoring data will be available on a MongoDB website with a unique URL accessible to you
        and anyone you share the URL with. MongoDB may use this information to make product
        improvements and to suggest MongoDB products and deployment options to you.

        To enable free monitoring, run the following command: db.enableFreeMonitoring()
        To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

Step 6: You can start creating new databases and use them.

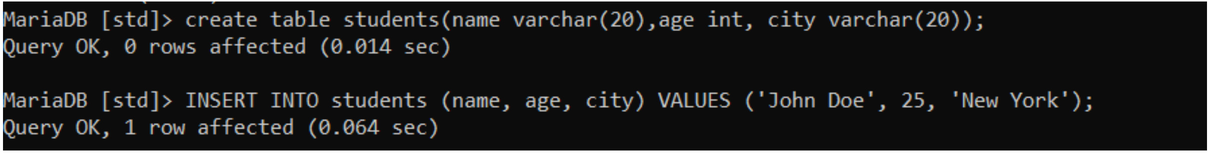
PROGRAM 11

AIM:

Compare relational and non-relational databases.

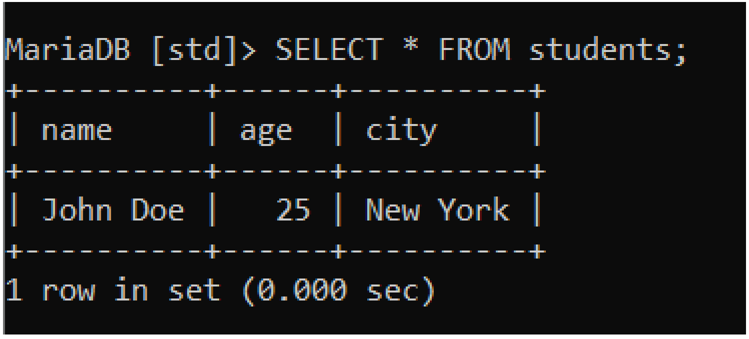
MySQL in MySQL Command-Line Interface:

Create a new row:
INSERT INTO students (name, age, city) VALUES ('John Doe', 25, 'New York');



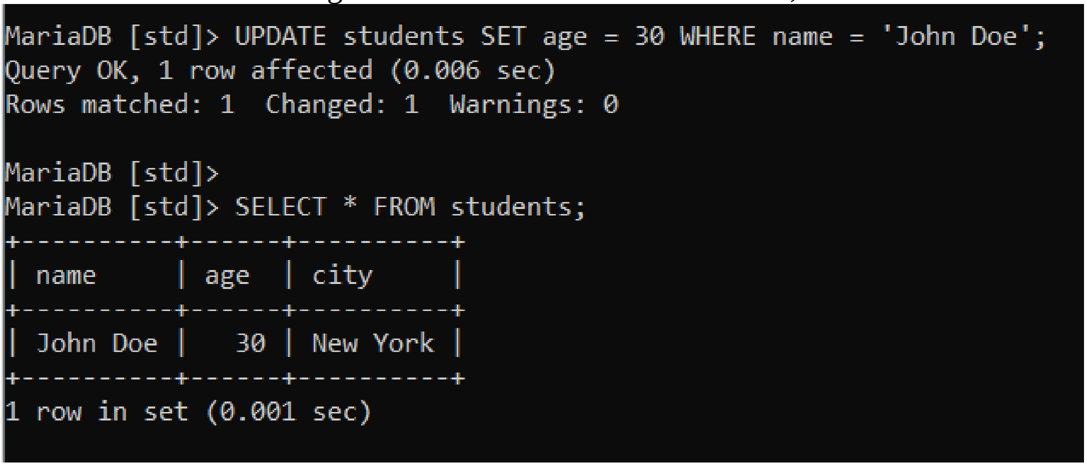
```
MariaDB [std]> create table students(name varchar(20),age int, city varchar(20));
Query OK, 0 rows affected (0.014 sec)

MariaDB [std]> INSERT INTO students (name, age, city) VALUES ('John Doe', 25, 'New York');
Query OK, 1 row affected (0.064 sec)
```

Find rows:
SELECT * FROM students;



```
MariaDB [std]> SELECT * FROM students;
+----------+------+----------+
| name     | age  | city     |
+----------+------+----------+
| John Doe |   25 | New York |
+----------+------+----------+
1 row in set (0.000 sec)
```

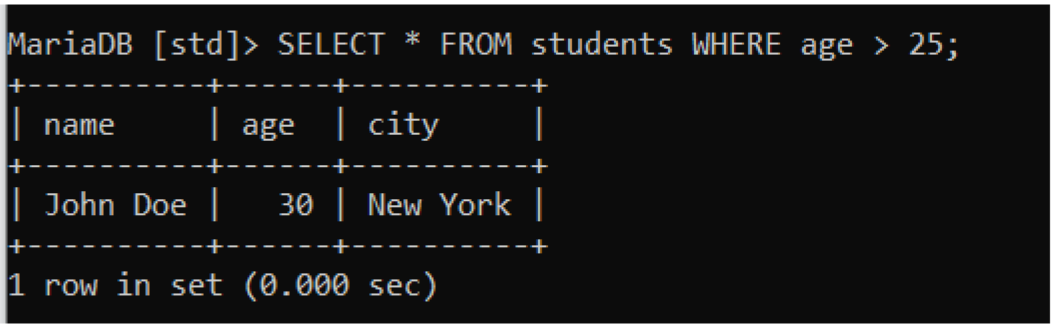
Update a row:
UPDATE students SET age = 30 WHERE name = 'John Doe';



```
MariaDB [std]> UPDATE students SET age = 30 WHERE name = 'John Doe';
Query OK, 1 row affected (0.006 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [std]>
MariaDB [std]> SELECT * FROM students;
+----------+------+----------+
| name     | age  | city     |
+----------+------+----------+
| John Doe |   30 | New York |
+----------+------+----------+
1 row in set (0.001 sec)
```

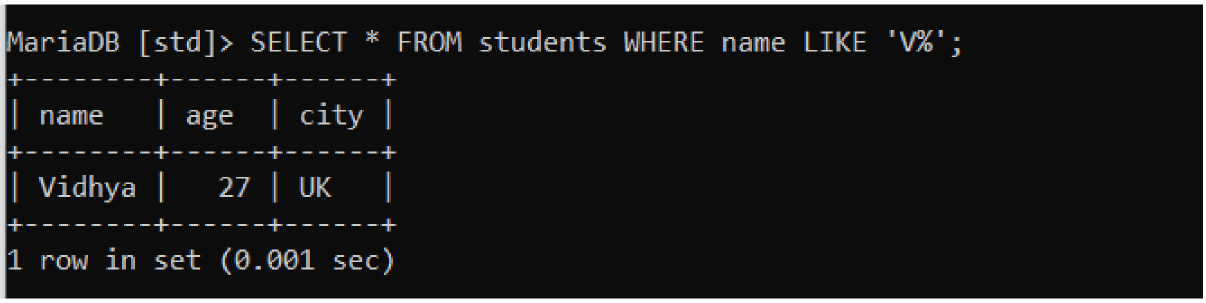
Select rows where the age is greater than 25:

SELECT * FROM students WHERE age > 25;



```
MariaDB [std]> SELECT * FROM students WHERE age > 25;
+----------+------+----------+
| name     | age  | city     |
+----------+------+----------+
| John Doe |   30 | New York |
+----------+------+----------+
1 row in set (0.000 sec)
```

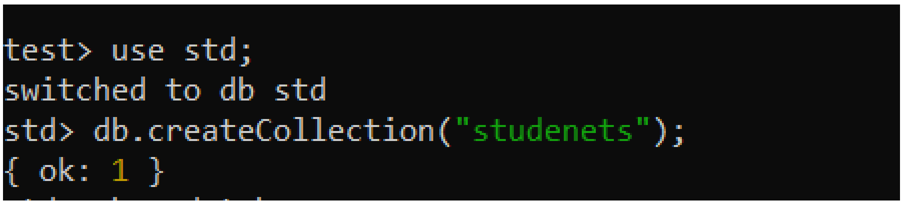
Select rows where the name starts with 'V':
SELECT * FROM students WHERE name LIKE 'V%';



```
MariaDB [std]> SELECT * FROM students WHERE name LIKE 'V%';
+--------+------+------+
| name   | age  | city |
+--------+------+------+
| Vidhya |   27 | UK   |
+--------+------+------+
1 row in set (0.001 sec)
```

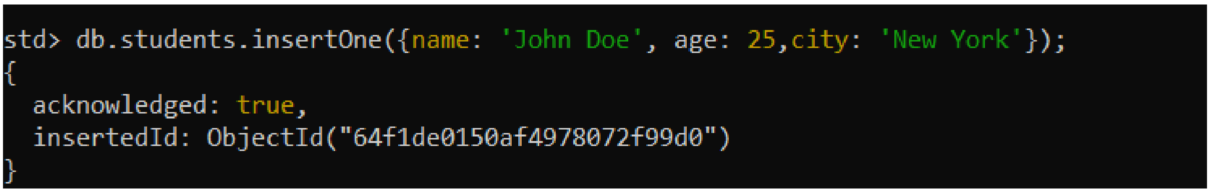
MongoDB in mongosh:

Create a new document:



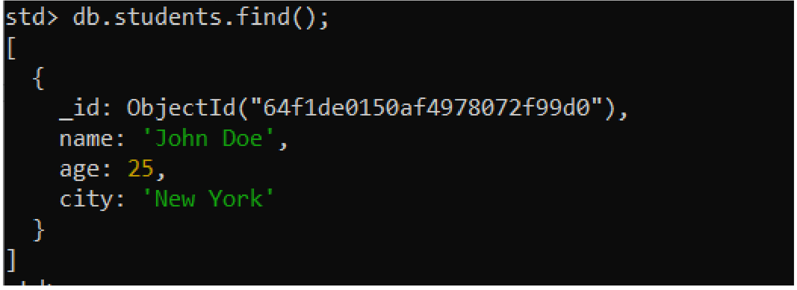
```
test> use std;
switched to db std
std> db.createCollection("studenets");
{ ok: 1 }
```

Insert into document
db.students.insertOne({name: 'John Doe',age: 25,city: 'New York'});



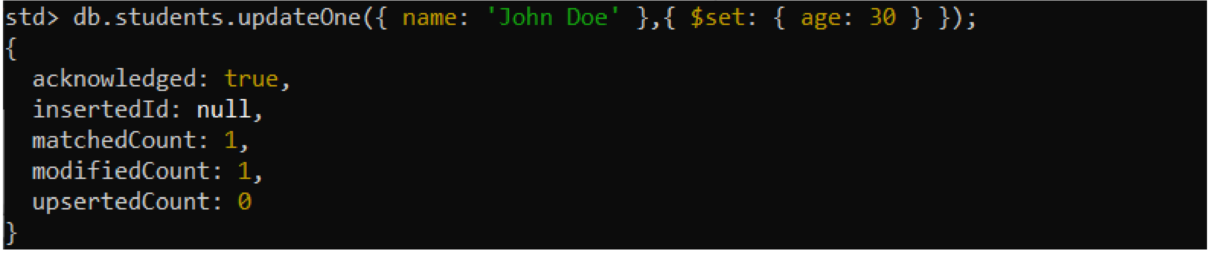
```
std> db.students.insertOne({name: 'John Doe', age: 25,city: 'New York'});
{
  acknowledged: true,
  insertedId: ObjectId("64f1de0150af4978072f99d0")
}
```

Find documents:
// Find all documents
db.students.find();



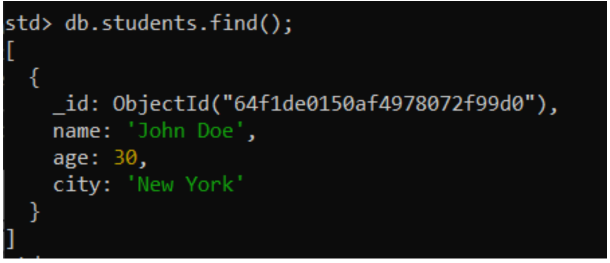
```
std> db.students.find();
[
  {
    _id: ObjectId("64f1de0150af4978072f99d0"),
    name: 'John Doe',
    age: 25,
    city: 'New York'
  }
]
```

Update a document:
db.students.updateOne({ name: 'John Doe' },{ $set: { age: 30 } });



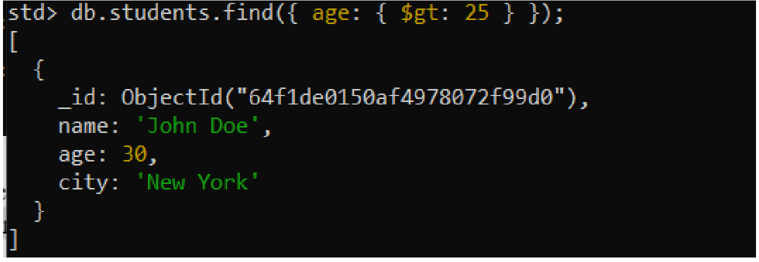
```
std> db.students.updateOne({ name: 'John Doe' },{ $set: { age: 30 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Find all documents in a collection:
db.students.find();



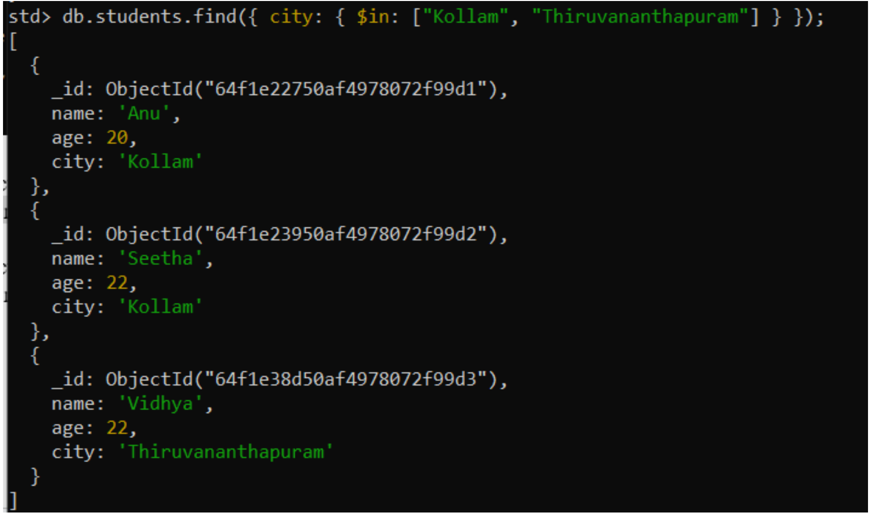
```
std> db.students.find();
[
  {
    _id: ObjectId("64f1de0150af4978072f99d0"),
    name: 'John Doe',
    age: 30,
    city: 'New York'
  }
]
```

Find documents where the age is greater than 25:
db.students.find({ age: { $gt: 25 } });



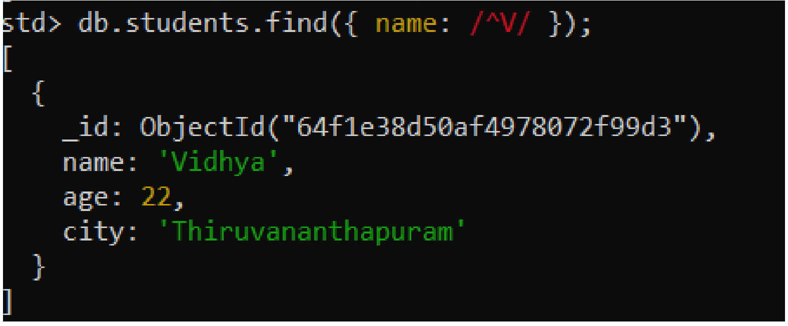
```
std> db.students.find({ age: { $gt: 25 } });
[
  {
    _id: ObjectId("64f1de0150af4978072f99d0"),
    name: 'John Doe',
    age: 30,
    city: 'New York'
  }
]
```

Find documents where the city is either "Kollam" or "Thiruvananthapuram":
db.students.find({ city: { $in: ["Kollam", "Thiruvananthapuram"] } });



```
std> db.students.find({ city: { $in: ["Kollam", "Thiruvananthapuram"] } });
[
  {
    _id: ObjectId("64f1e22750af4978072f99d1"),
    name: 'Anu',
    age: 20,
    city: 'Kollam'
  },
  {
    _id: ObjectId("64f1e23950af4978072f99d2"),
    name: 'Seetha',
    age: 22,
    city: 'Kollam'
  },
  {
    _id: ObjectId("64f1e38d50af4978072f99d3"),
    name: 'Vidhya',
    age: 22,
    city: 'Thiruvananthapuram'
  }
]
```

Find documents where the name starts with 'V':
db.students.find({ fname: /^V/ });



```
std> db.students.find({ name: /^V/ });
[
  {
    _id: ObjectId("64f1e38d50af4978072f99d3"),
    name: 'Vidhya',
    age: 22,
    city: 'Thiruvananthapuram'
  }
]
```

For MySQL, MySQL command-line interface, and for MongoDB, mongosh shell are used.

CO 4

Apply CRUD operations and retrieve data in a NoSQL environment

PROGRAM 12

AIM: Create a database COLLEGE and create a collection named students in it. Retrieve data from collection

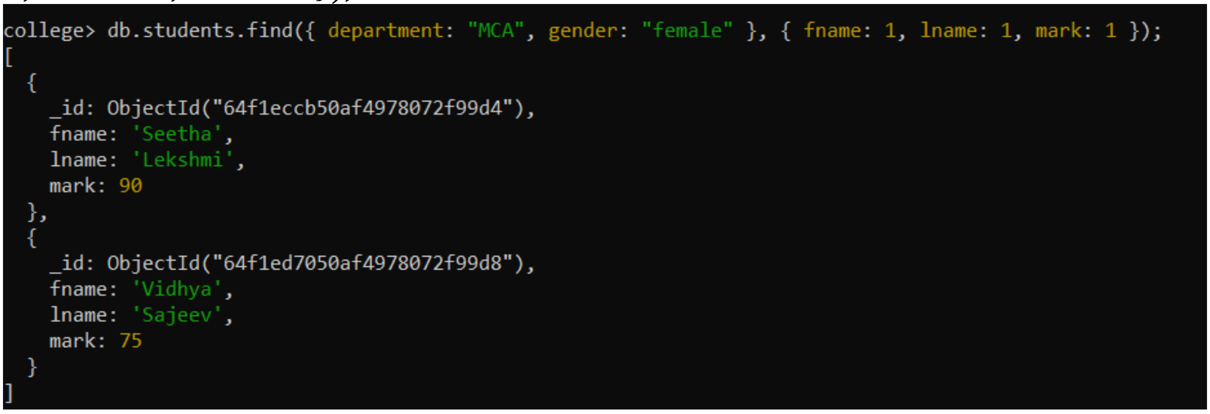
Retrieval of data from Mongodb

Using MongoDB shell (mongosh) commands :

use college;

1. Display name (both fname and lname) and mark of all female students in MCA department.

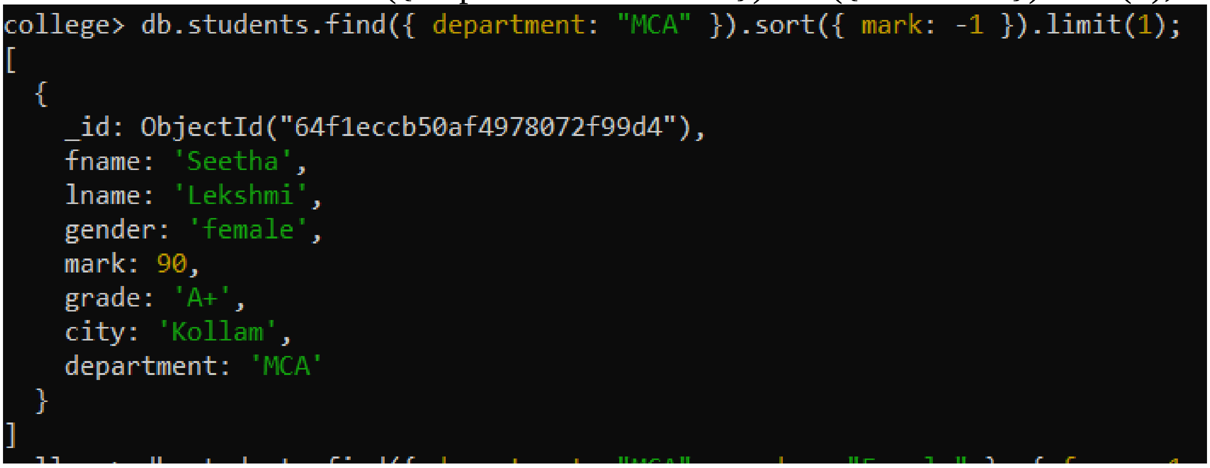
Answer: db.students.find({ department: "MCA", gender: "Female" }, { fname: 1, lname: 1, mark: 1 });



```
college> db.students.find({ department: "MCA", gender: "female" }, { fname: 1, lname: 1, mark: 1 });
[
  {
    _id: ObjectId("64f1eccb50af4978072f99d4"),
    fname: 'Seetha',
    lname: 'Lekshmi',
    mark: 90
  },
  {
    _id: ObjectId("64f1ed7050af4978072f99d8"),
    fname: 'Vidhya',
    lname: 'Sajeev',
    mark: 75
  }
]
```

2. Display the details of the student who secured the highest mark in the course MCA.

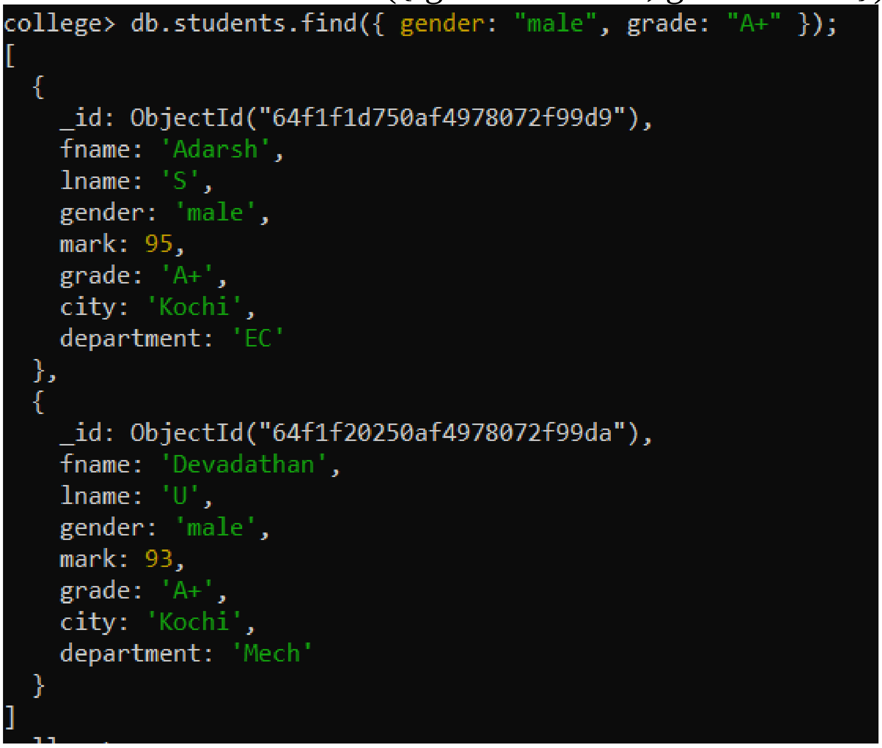
Answer: db.students.find({ department: "MCA" }).sort({ mark: -1 }).limit(1);



```
college> db.students.find({ department: "MCA" }).sort({ mark: -1 }).limit(1);
[
  {
    _id: ObjectId("64f1eccb50af4978072f99d4"),
    fname: 'Seetha',
    lname: 'Lekshmi',
    gender: 'female',
    mark: 90,
    grade: 'A+',
    city: 'Kollam',
    department: 'MCA'
  }
]
```

3. Display all male students who secured an A+ grade.

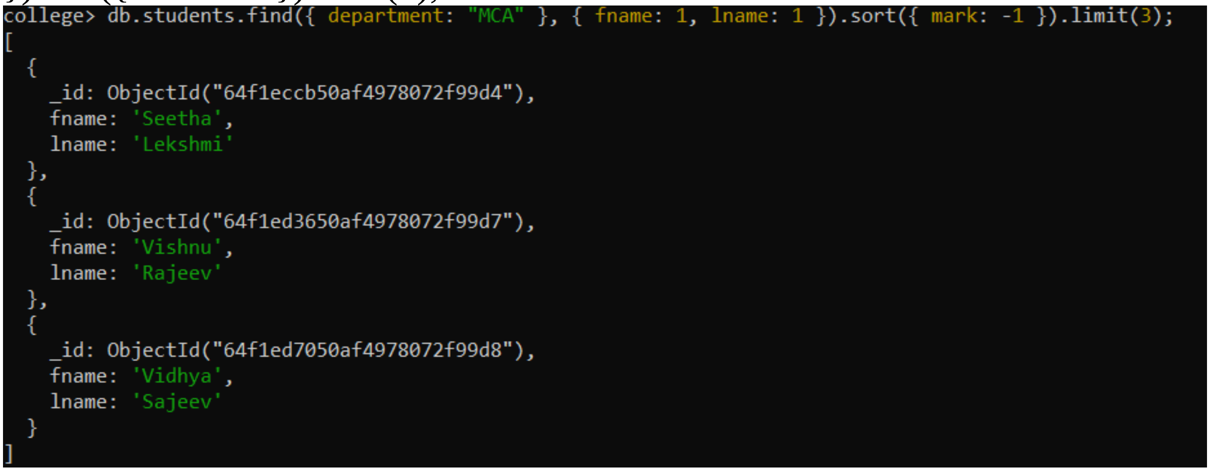
Answer: db.students.find({ gender: "male", grade: "A+" });



```
college> db.students.find({ gender: "male", grade: "A+" });
[
  {
    _id: ObjectId("64f1f1d750af4978072f99d9"),
    fname: 'Adarsh',
    lname: 'S',
    gender: 'male',
    mark: 95,
    grade: 'A+',
    city: 'Kochi',
    department: 'EC'
  },
  {
    _id: ObjectId("64f1f20250af4978072f99da"),
    fname: 'Devadathan',
    lname: 'U',
    gender: 'male',
    mark: 93,
    grade: 'A+',
    city: 'Kochi',
    department: 'Mech'
  }
]
```

4.Display the names of the top two students in the Mechanical department.

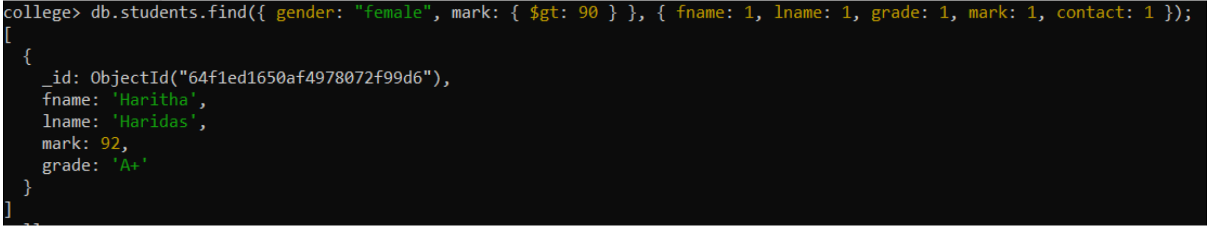
Answer: db.students.find({ department: "MCA" }, { fname: 1, lname: 1 }).sort({ mark: -1 }).limit(3);



```
college> db.students.find({ department: "MCA" }, { fname: 1, lname: 1 }).sort({ mark: -1 }).limit(3);
[
  {
    _id: ObjectId("64f1eccb50af4978072f99d4"),
    fname: 'Seetha',
    lname: 'Lekshmi'
  },
  {
    _id: ObjectId("64f1ed3650af4978072f99d7"),
    fname: 'Vishnu',
    lname: 'Rajeev'
  },
  {
    _id: ObjectId("64f1ed7050af4978072f99d8"),
    fname: 'Vidhya',
    lname: 'Sajeev'
  }
]
```

5.Display the details of female students [fname, lname, grade, mark, contact] who achieved a mark more than 90.

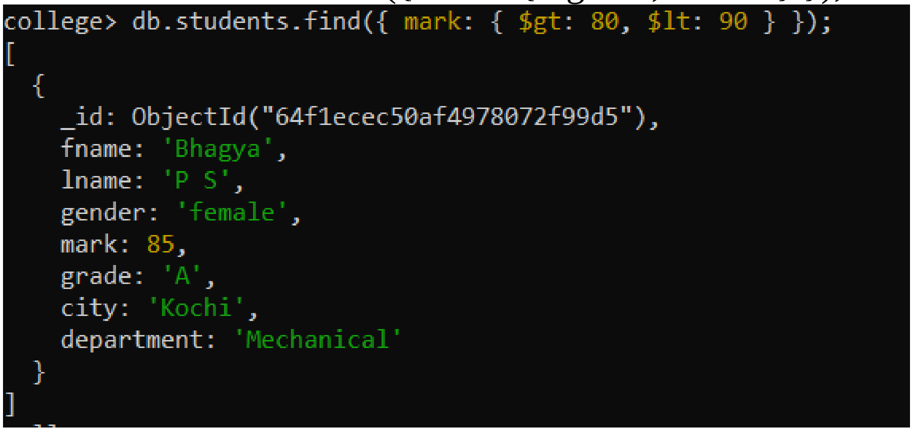
Answer: db.students.find({ gender: "female", mark: { $gt: 90 } }, { fname: 1, lname: 1, grade: 1, mark: 1, contact: 1 });



```
college> db.students.find({ gender: "female", mark: { $gt: 90 } }, { fname: 1, lname: 1, grade: 1, mark: 1, contact: 1 });
[
  {
    _id: ObjectId("64f1ed1650af4978072f99d6"),
    fname: 'Haritha',
    lname: 'Haridas',
    mark: 92,
    grade: 'A+'
  }
]
```

6.Display the details of students who secured a mark more than 80 but less than 90.

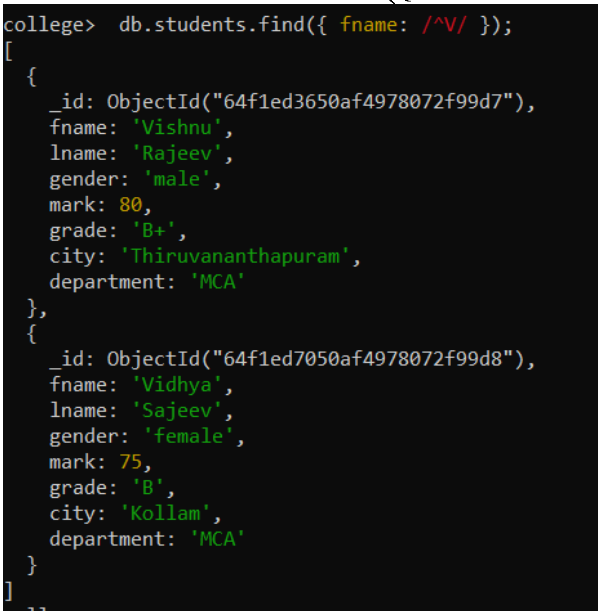
Answer: db.students.find({ mark: { $gt: 80, $lt: 90 } });



```
college> db.students.find({ mark: { $gt: 80, $lt: 90 } });
[
  {
    _id: ObjectId("64f1ecec50af4978072f99d5"),
    fname: 'Bhagya',
    lname: 'P S',
    gender: 'female',
    mark: 85,
    grade: 'A',
    city: 'Kochi',
    department: 'Mechanical'
  }
]
```

7.Display the details of students whose name starts with 'V':

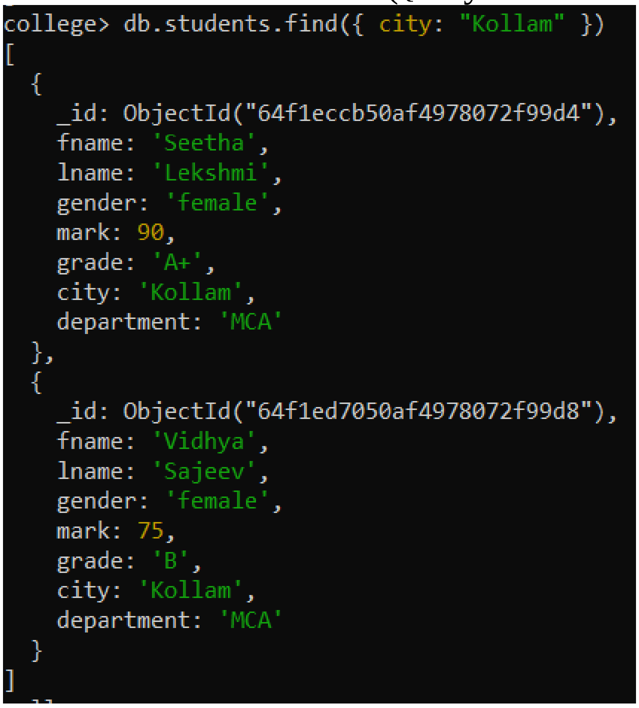
Answer: db.students.find({ fname: /^V/ })



```
college>  db.students.find({ fname: /^V/ });
[
  {
    _id: ObjectId("64f1ed3650af4978072f99d7"),
    fname: 'Vishnu',
    lname: 'Rajeev',
    gender: 'male',
    mark: 80,
    grade: 'B+',
    city: 'Thiruvananthapuram',
    department: 'MCA'
  },
  {
    _id: ObjectId("64f1ed7050af4978072f99d8"),
    fname: 'Vidhya',
    lname: 'Sajeev',
    gender: 'female',
    mark: 75,
    grade: 'B',
    city: 'Kollam',
    department: 'MCA'
  }
]
```

8.Display all students from Kollam:

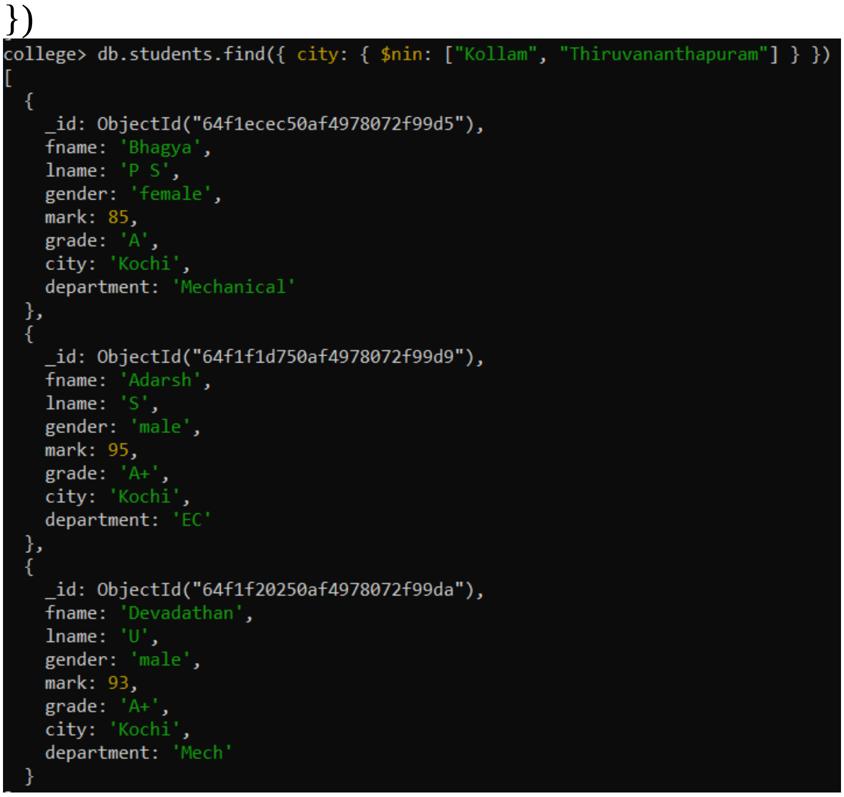
Answer: db.students.find({ city: "Kollam" })



```
college> db.students.find({ city: "Kollam" })
[
  {
    _id: ObjectId("64f1eccb50af4978072f99d4"),
    fname: 'Seetha',
    lname: 'Lekshmi',
    gender: 'female',
    mark: 90,
    grade: 'A+',
    city: 'Kollam',
    department: 'MCA'
  },
  {
    _id: ObjectId("64f1ed7050af4978072f99d8"),
    fname: 'Vidhya',
    lname: 'Sajeev',
    gender: 'female',
    mark: 75,
    grade: 'B',
    city: 'Kollam',
    department: 'MCA'
  }
]
```

9.Display all students who do not belong to either Kollam or Thiruvananthapuram:

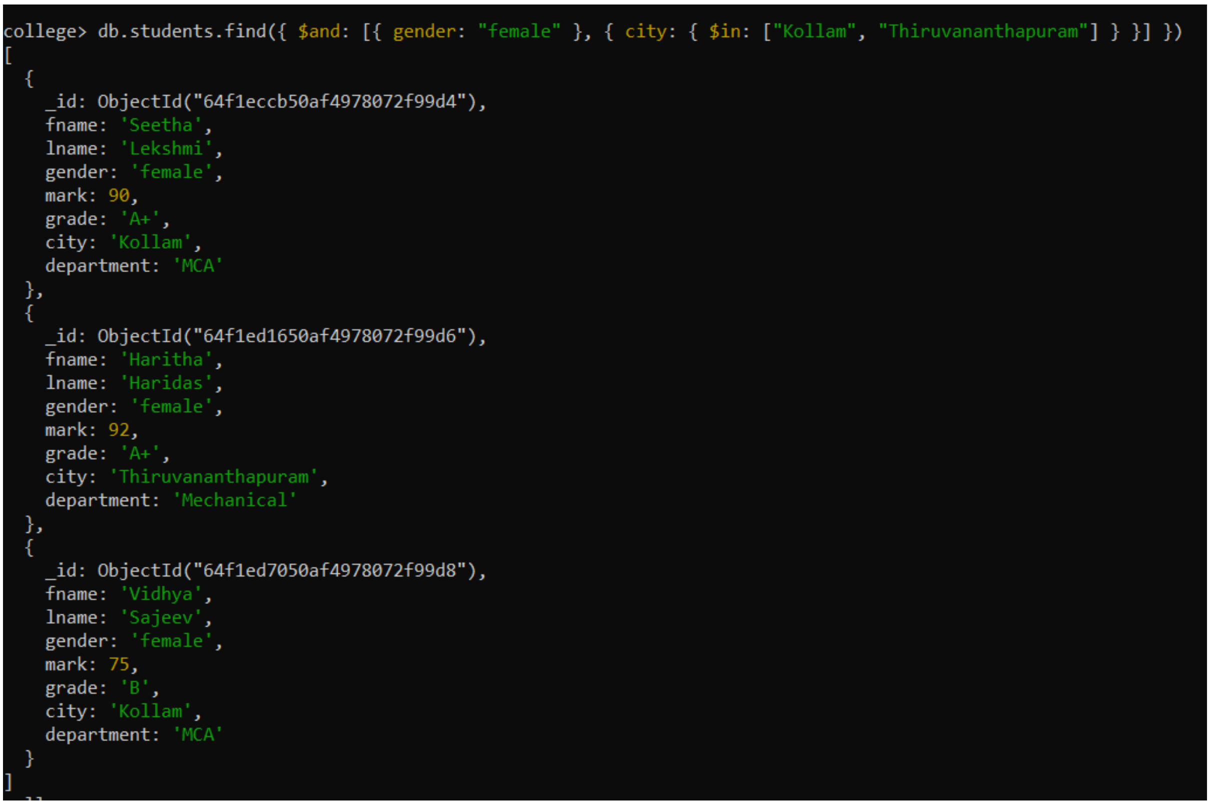
Answer: db.students.find({ city: { $nin: ["Kollam", "Thiruvananthapuram"] } })



```
college> db.students.find({ city: { $nin: ["Kollam", "Thiruvananthapuram"] } })
[
  {
    _id: ObjectId("64f1ecec50af4978072f99d5"),
    fname: 'Bhagya',
    lname: 'P S',
    gender: 'female',
    mark: 85,
    grade: 'A',
    city: 'Kochi',
    department: 'Mechanical'
  },
  {
    _id: ObjectId("64f1f1d750af4978072f99d9"),
    fname: 'Adarsh',
    lname: 'S',
    gender: 'male',
    mark: 95,
    grade: 'A+',
    city: 'Kochi',
    department: 'EC'
  },
  {
    _id: ObjectId("64f1f20250af4978072f99da"),
    fname: 'Devadathan',
    lname: 'U',
    gender: 'male',
    mark: 93,
    grade: 'A+',
    city: 'Kochi',
    department: 'Mech'
  }
```

10.Display all female students who belong to either Kollam or Thiruvananthapuram:

Answer: db.students.find({ $and: [{ gender: "female" }, { city: { $in: ["Kollam", "Thiruvananthapuram"] } }] })



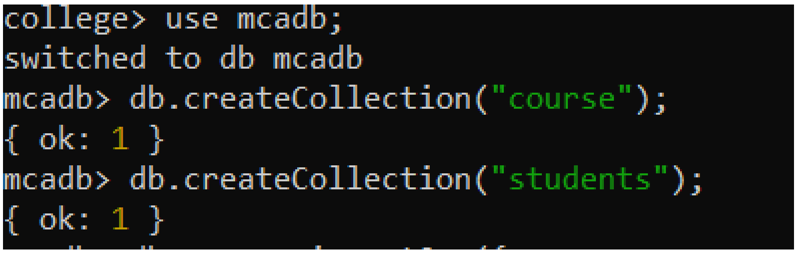
```
college> db.students.find({ $and: [{ gender: "female" }, { city: { $in: ["Kollam", "Thiruvananthapuram"] } }] })
[
  {
    _id: ObjectId("64f1eccb50af4978072f99d4"),
    fname: 'Seetha',
    lname: 'Lekshmi',
    gender: 'female',
    mark: 90,
    grade: 'A+',
    city: 'Kollam',
    department: 'MCA'
  },
  {
    _id: ObjectId("64f1ed1650af4978072f99d6"),
    fname: 'Haritha',
    lname: 'Haridas',
    gender: 'female',
    mark: 92,
    grade: 'A+',
    city: 'Thiruvananthapuram',
    department: 'Mechanical'
  },
  {
    _id: ObjectId("64f1ed7050af4978072f99d8"),
    fname: 'Vidhya',
    lname: 'Sajeev',
    gender: 'female',
    mark: 75,
    grade: 'B',
    city: 'Kollam',
    department: 'MCA'
  }
]
```

PROGRAM 13

AIM: Create a database in MongoDB named MCADB with collections named COURSE and STUDENTS and perform CRUD operations on it. Perform the CRUD operations using mongoshell and pymongo.

COMMANDS

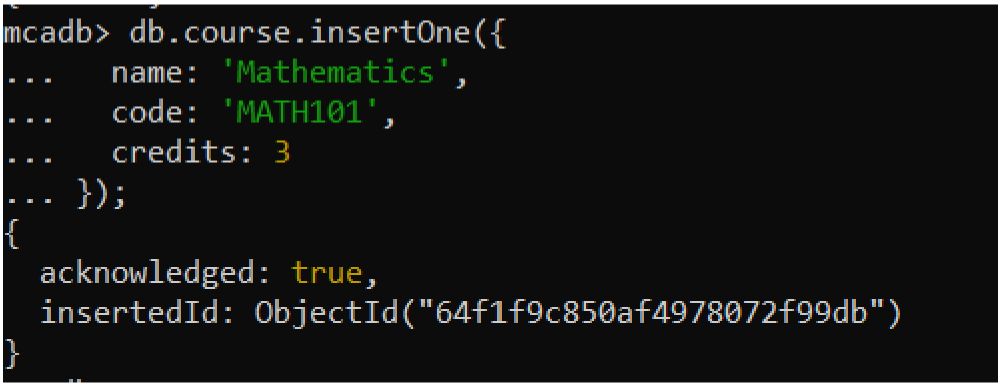
Using MongoDB shell (mongosh) commands for the CRUD operations



```
college> use mcadb;
switched to db mcadb
mcadb> db.createCollection("course");
{ ok: 1 }
mcadb> db.createCollection("students");
{ ok: 1 }
```

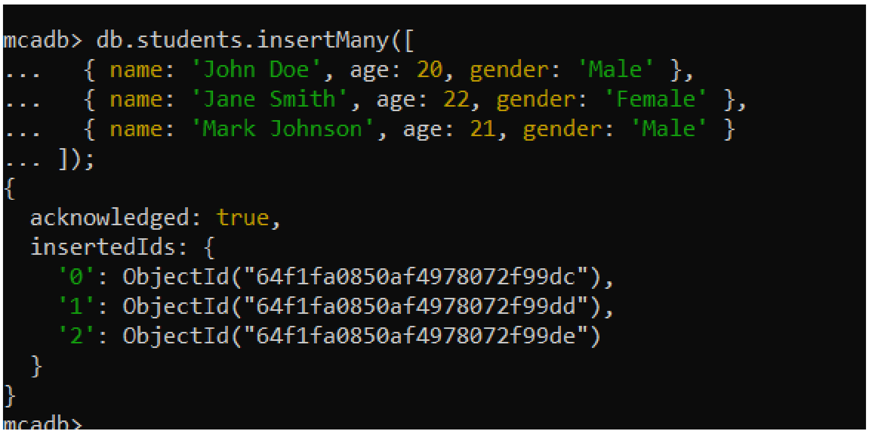
1. **Create/Insert Documents**:

```
// Insert a document into the "course" collection
db.course.insertOne({
  name: 'Mathematics',
  code: 'MATH101',
  credits: 3
});
```



```
mcadb> db.course.insertOne({
...    name: 'Mathematics',
...    code: 'MATH101',
...    credits: 3
... });
{
  acknowledged: true,
  insertedId: ObjectId("64f1f9c850af4978072f99db")
}
```

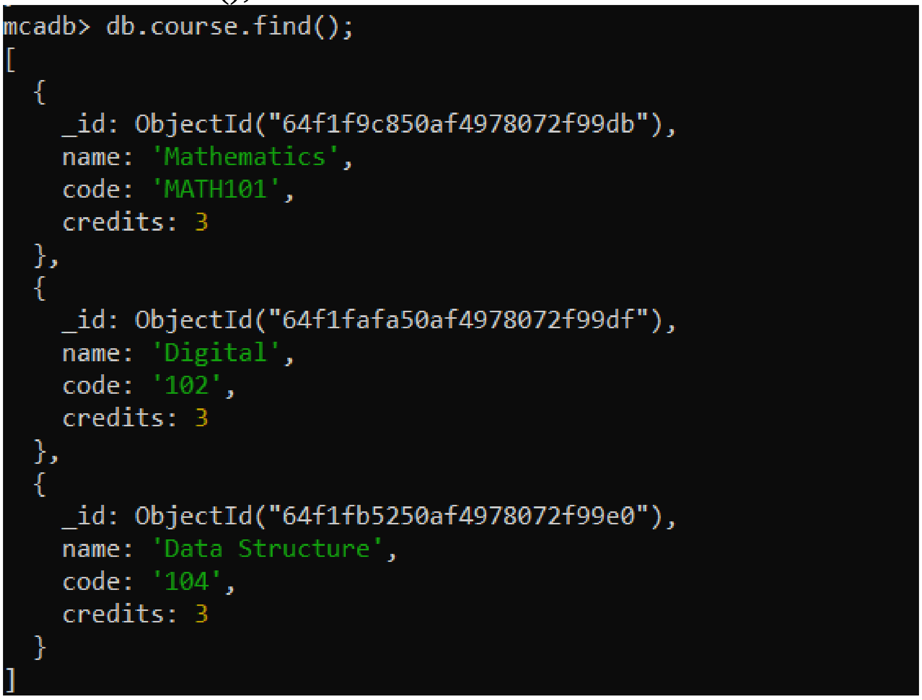
```
// Insert multiple documents into the "students" collection
db.students.insertMany([
  { name: 'John Doe', age: 20, gender: 'Male' },
  { name: 'Jane Smith', age: 22, gender: 'Female' },
  { name: 'Mark Johnson', age: 21, gender: 'Male' }
]);
```



```
mcadb> db.students.insertMany([
...    { name: 'John Doe', age: 20, gender: 'Male' },
...    { name: 'Jane Smith', age: 22, gender: 'Female' },
...    { name: 'Mark Johnson', age: 21, gender: 'Male' }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("64f1fa0850af4978072f99dc"),
    '1': ObjectId("64f1fa0850af4978072f99dd"),
    '2': ObjectId("64f1fa0850af4978072f99de")
  }
}
mcadb>
```

2. **Read/Retrieve Documents**:

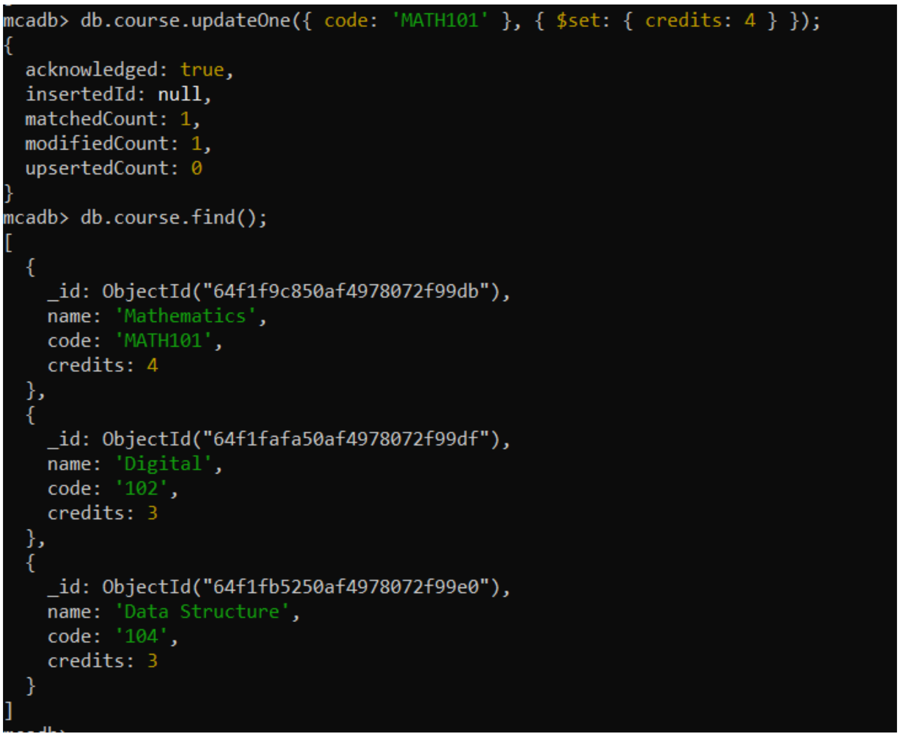
// Retrieve all documents from the "course" collection
db.course.find();



```
mcadb> db.course.find();
[
  {
    _id: ObjectId("64f1f9c850af4978072f99db"),
    name: 'Mathematics',
    code: 'MATH101',
    credits: 3
  },
  {
    _id: ObjectId("64f1fafa50af4978072f99df"),
    name: 'Digital',
    code: '102',
    credits: 3
  },
  {
    _id: ObjectId("64f1fb5250af4978072f99e0"),
    name: 'Data Structure',
    code: '104',
    credits: 3
  }
]
```

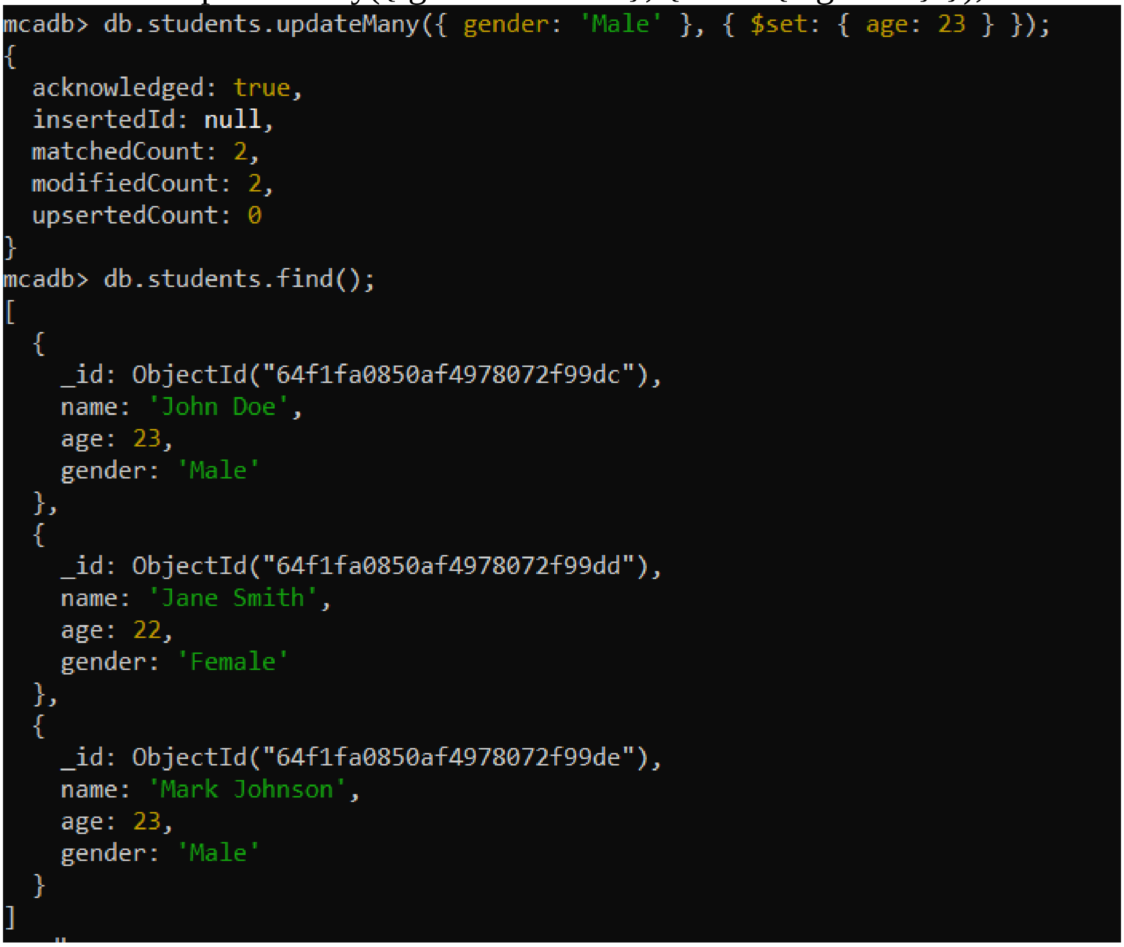
3. **Update Documents**:

// Update a document in the "course" collection
db.course.updateOne({ code: 'MATH101' }, { $set: { credits: 4 } });



```
mcadb> db.course.updateOne({ code: 'MATH101' }, { $set: { credits: 4 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
mcadb> db.course.find();
[
  {
    _id: ObjectId("64f1f9c850af4978072f99db"),
    name: 'Mathematics',
    code: 'MATH101',
    credits: 4
  },
  {
    _id: ObjectId("64f1fafa50af4978072f99df"),
    name: 'Digital',
    code: '102',
    credits: 3
  },
  {
    _id: ObjectId("64f1fb5250af4978072f99e0"),
    name: 'Data Structure',
    code: '104',
    credits: 3
  }
]
```

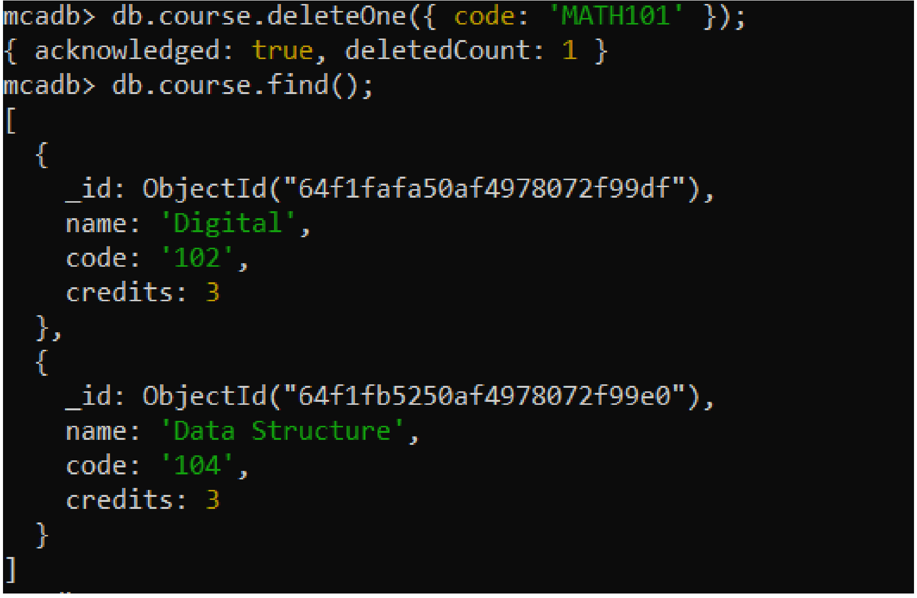
// Update multiple documents in the "students" collection
db.students.updateMany({ gender: 'Male' }, { $set: { age: 23 } });



```
mcadb> db.students.updateMany({ gender: 'Male' }, { $set: { age: 23 } });
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
mcadb> db.students.find();
[
  {
    _id: ObjectId("64f1fa0850af4978072f99dc"),
    name: 'John Doe',
    age: 23,
    gender: 'Male'
  },
  {
    _id: ObjectId("64f1fa0850af4978072f99dd"),
    name: 'Jane Smith',
    age: 22,
    gender: 'Female'
  },
  {
    _id: ObjectId("64f1fa0850af4978072f99de"),
    name: 'Mark Johnson',
    age: 23,
    gender: 'Male'
  }
]
```

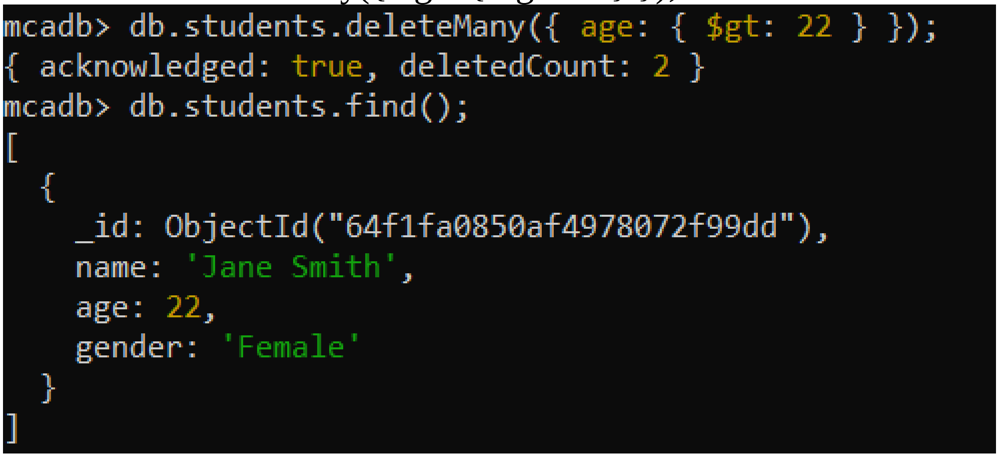
4. **Delete Documents**:

// Delete a document from the "course" collection
db.course.deleteOne({ code: 'MATH101' });



```
mcadb> db.course.deleteOne({ code: 'MATH101' });
{ acknowledged: true, deletedCount: 1 }
mcadb> db.course.find();
[
  {
    _id: ObjectId("64f1fafa50af4978072f99df"),
    name: 'Digital',
    code: '102',
    credits: 3
  },
  {
    _id: ObjectId("64f1fb5250af4978072f99e0"),
    name: 'Data Structure',
    code: '104',
    credits: 3
  }
]
```

// Delete multiple documents from the "students" collection
db.students.deleteMany({ age: { $gt: 22 } });



```
mcadb> db.students.deleteMany({ age: { $gt: 22 } });
{ acknowledged: true, deletedCount: 2 }
mcadb> db.students.find();
[
  {
    _id: ObjectId("64f1fa0850af4978072f99dd"),
    name: 'Jane Smith',
    age: 22,
    gender: 'Female'
  }
]
```

Using Pymongo for the CRUD operations

1. **Create/Insert Documents**:

from pymongo import MongoClient

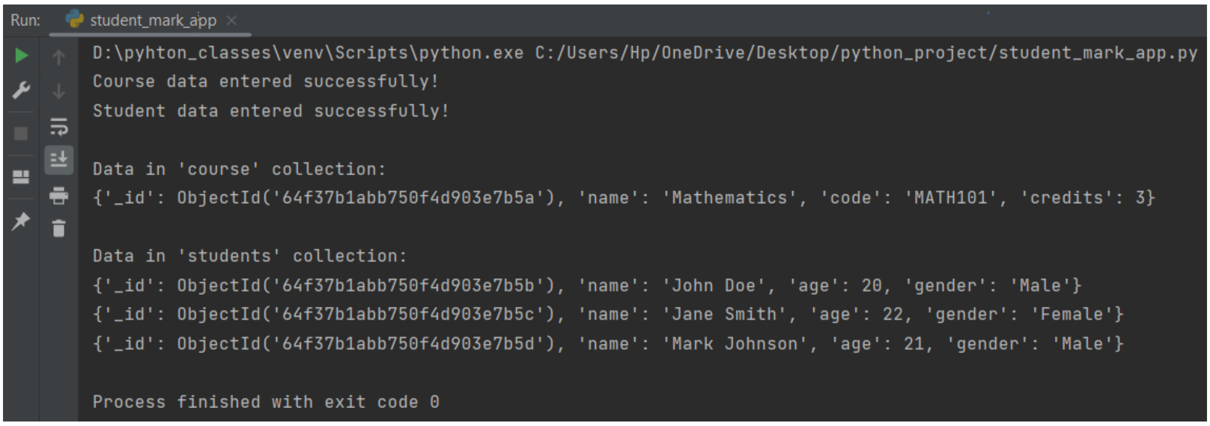
# Connect to the MongoDB server
client = MongoClient('mongodb://localhost:27017/')

# Select the database
db = client.mcadb

# Insert a document into the "course" collection
course = {

```
    'name': 'Mathematics',
    'code': 'MATH101',
    'credits': 3
}
db.course.insert_one(course)
print("Course data entered successfully!")

# Insert multiple documents into the "students" collection
students = [
    {'name': 'John Doe', 'age': 20, 'gender': 'Male'},
    {'name': 'Jane Smith', 'age': 22, 'gender': 'Female'},
    {'name': 'Mark Johnson', 'age': 21, 'gender': 'Male'}
]
db.students.insert_many(students)
print("Student data entered successfully!")

# Print the inserted data from the "course" collection
print("\nData in 'course' collection:")
for doc in db.course.find():
    print(doc)

# Print the inserted data from the "students" collection
print("\nData in 'students' collection:")
for doc in db.students.find():
    print(doc)
```

OUTPUT

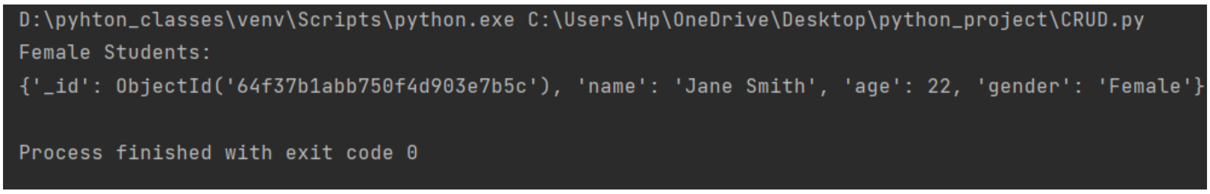


```
D:\pyhton_classes\venv\Scripts\python.exe C:/Users/Hp/OneDrive/Desktop/python_project/student_mark_app.py
Course data entered successfully!
Student data entered successfully!

Data in 'course' collection:
{'_id': ObjectId('64f37b1abb750f4d903e7b5a'), 'name': 'Mathematics', 'code': 'MATH101', 'credits': 3}

Data in 'students' collection:
{'_id': ObjectId('64f37b1abb750f4d903e7b5b'), 'name': 'John Doe', 'age': 20, 'gender': 'Male'}
{'_id': ObjectId('64f37b1abb750f4d903e7b5c'), 'name': 'Jane Smith', 'age': 22, 'gender': 'Female'}
{'_id': ObjectId('64f37b1abb750f4d903e7b5d'), 'name': 'Mark Johnson', 'age': 21, 'gender': 'Male'}

Process finished with exit code 0
```

2. **Read/Retrieve Documents**:

```
from pymongo import MongoClient
client = MongoClient('mongodb://localhost:27017/')
db = client.mcadb
female_students = db.students.find({'gender': 'Female'})
print("Female Students:")
for student in female_students:
    print(student)
```

OUTPUT

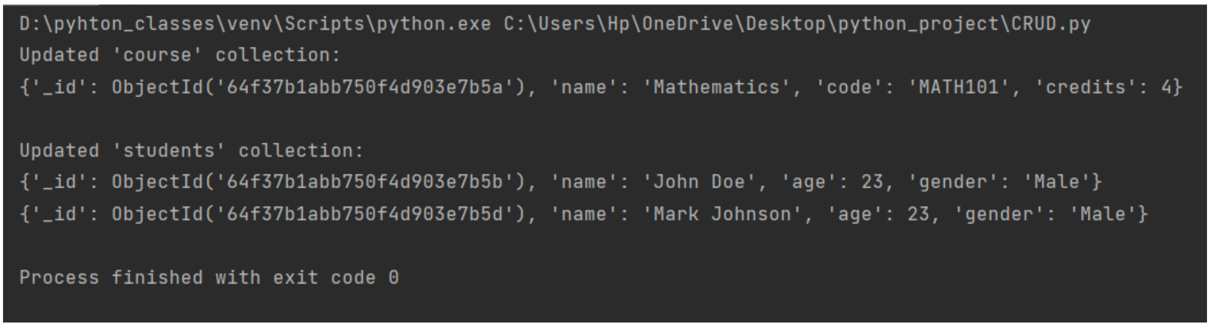


```
D:\pyhton_classes\venv\Scripts\python.exe C:\Users\Hp\OneDrive\Desktop\python_project\CRUD.py
Female Students:
{'_id': ObjectId('64f37b1abb750f4d903e7b5c'), 'name': 'Jane Smith', 'age': 22, 'gender': 'Female'}

Process finished with exit code 0
```

3. **Update Documents**:

```
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')
db = client.mcadb

db.course.update_one({'code': 'MATH101'}, {'$set': {'credits': 4}})
print("Updated 'course' collection:")

updated_course = db.course.find_one({'code': 'MATH101'})
print(updated_course)
db.students.update_many({'gender': 'Male'}, {'$set': {'age': 23}})
print("\nUpdated 'students' collection:")
updated_students = db.students.find({'gender': 'Male'})
for student in updated_students:
    print(student)
```

OUTPUT

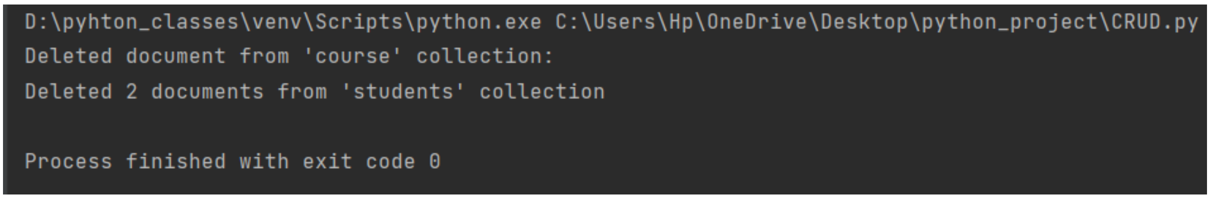


```
D:\pyhton_classes\venv\Scripts\python.exe C:\Users\Hp\OneDrive\Desktop\python_project\CRUD.py
Updated 'course' collection:
{'_id': ObjectId('64f37b1abb750f4d903e7b5a'), 'name': 'Mathematics', 'code': 'MATH101', 'credits': 4}

Updated 'students' collection:
{'_id': ObjectId('64f37b1abb750f4d903e7b5b'), 'name': 'John Doe', 'age': 23, 'gender': 'Male'}
{'_id': ObjectId('64f37b1abb750f4d903e7b5d'), 'name': 'Mark Johnson', 'age': 23, 'gender': 'Male'}

Process finished with exit code 0
```

4. **Delete Documents**:

```
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')
db = client.mcadb

# Delete a document from the "course" collection
db.course.delete_one({'code': 'MATH101'})
print("Deleted document from 'course' collection:")

deleted_students = db.students.delete_many({'age': {'$gt': 22}})
print(f"Deleted {deleted_students.deleted_count} documents from 'students'
collection")
```

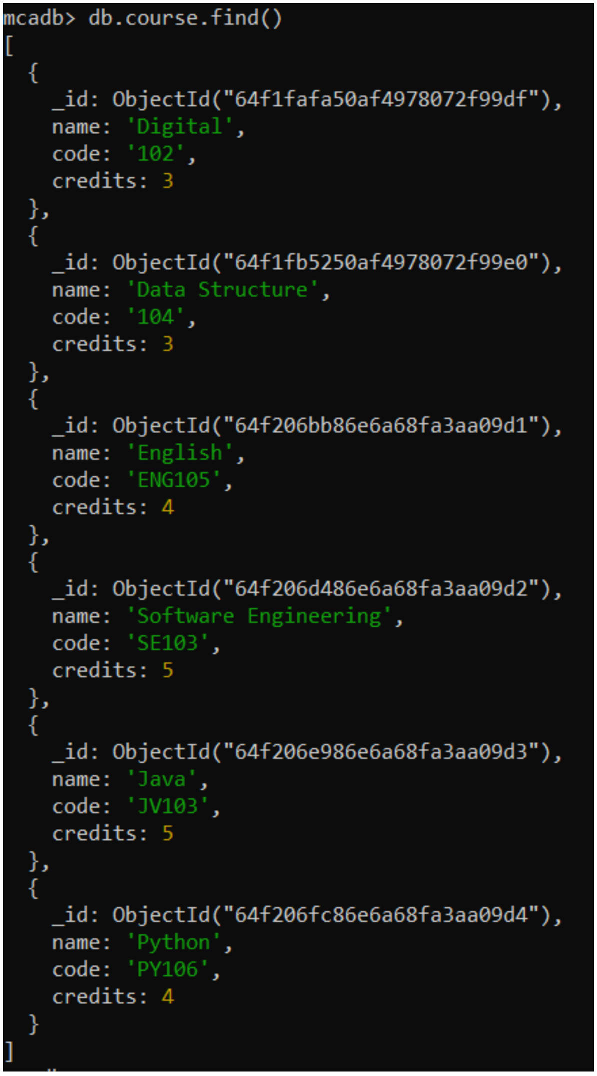
OUTPUT



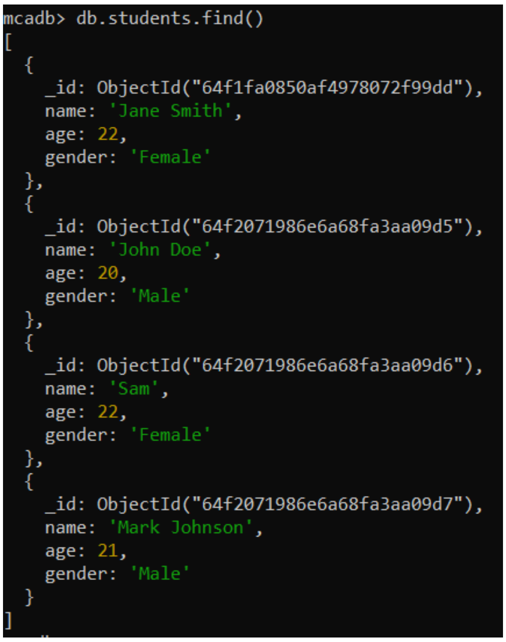
```
D:\pyhton_classes\venv\Scripts\python.exe C:\Users\Hp\OneDrive\Desktop\python_project\CRUD.py
Deleted document from 'course' collection:
Deleted 2 documents from 'students' collection

Process finished with exit code 0
```

PROGRAM 14

AIM: Create a database in MongoDB named "mcadb" with collections named "course" and "students" and perform aggregate functions, and regular expressions on it.



```
mcadb> db.course.find()
[
  {
    _id: ObjectId("64f1fafa50af4978072f99df"),
    name: 'Digital',
    code: '102',
    credits: 3
  },
  {
    _id: ObjectId("64f1fb5250af4978072f99e0"),
    name: 'Data Structure',
    code: '104',
    credits: 3
  },
  {
    _id: ObjectId("64f206bb86e6a68fa3aa09d1"),
    name: 'English',
    code: 'ENG105',
    credits: 4
  },
  {
    _id: ObjectId("64f206d486e6a68fa3aa09d2"),
    name: 'Software Engineering',
    code: 'SE103',
    credits: 5
  },
  {
    _id: ObjectId("64f206e986e6a68fa3aa09d3"),
    name: 'Java',
    code: 'JV103',
    credits: 5
  },
  {
    _id: ObjectId("64f206fc86e6a68fa3aa09d4"),
    name: 'Python',
    code: 'PY106',
    credits: 4
  }
]
```

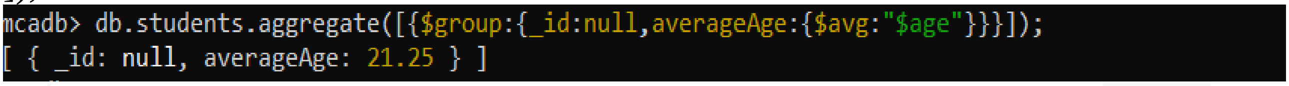


```
mcadb> db.students.find()
[
  {
    _id: ObjectId("64f1fa0850af4978072f99dd"),
    name: 'Jane Smith',
    age: 22,
    gender: 'Female'
  },
  {
    _id: ObjectId("64f2071986e6a68fa3aa09d5"),
    name: 'John Doe',
    age: 20,
    gender: 'Male'
  },
  {
    _id: ObjectId("64f2071986e6a68fa3aa09d6"),
    name: 'Sam',
    age: 22,
    gender: 'Female'
  },
  {
    _id: ObjectId("64f2071986e6a68fa3aa09d7"),
    name: 'Mark Johnson',
    age: 21,
    gender: 'Male'
  }
]
```

1. **Aggregate Functions**:

// Calculate the average age of students

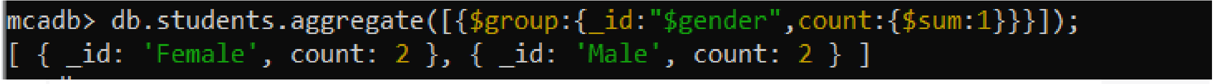
db.students.aggregate([{ $group: { _id: null, averageAge: { $avg: "$age" } } }]);



```
mcadb> db.students.aggregate([{$group:{_id:null,averageAge:{$avg:"$age"}}}]);
[ { _id: null, averageAge: 21.25 } ]
```

- **$group**: This stage groups documents based on a specified criteria.
- **_id:** null: It groups all documents into a single group because _id: null means there's no specific field used for grouping, and all documents are treated as a single group.
- **averageAge: { $avg: "$age" }**: Inside the group, it calculates the average age of all the documents within that group. $avg is an aggregation operator that calculates the average of the specified field, in this case, the "age" field.

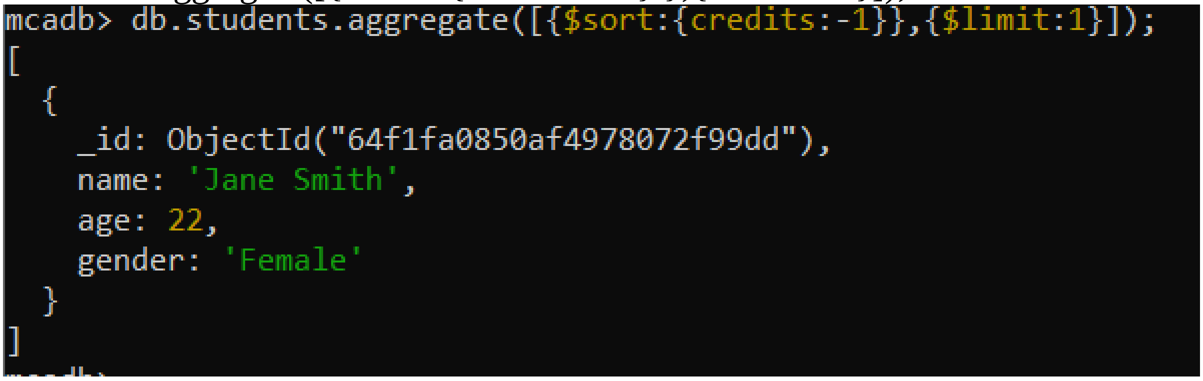
```
// Count the number of male and female students
db.students.aggregate([
  { $group: { _id: "$gender", count: { $sum: 1 } } }
]);
```



```
mcadb> db.students.aggregate([{$group:{_id:"$gender",count:{$sum:1}}}]);
[ { _id: 'Female', count: 2 }, { _id: 'Male', count: 2 } ]
```

- **$group**: This stage groups documents based on a specified criteria.
- **_id**: "$gender": It groups documents by the "gender" field. This means that it will create separate groups for each unique gender value found in the collection.
- **count: { $sum: 1 }:** Within each group, it calculates a count of documents using the $sum operator. It increments the count by 1 for each document within the group.

```
// Find the courses with the highest number of credits
db.course.aggregate([{ $sort: { credits: -1 } },{ $limit: 1 }]);
```

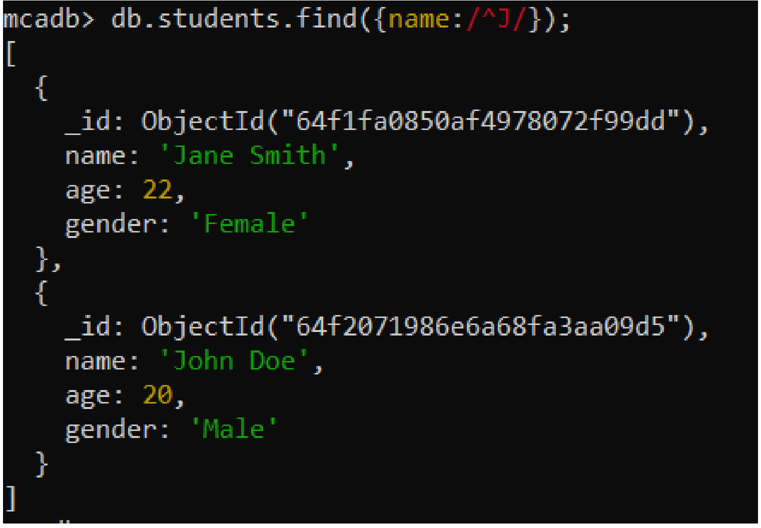


```
mcadb> db.students.aggregate([{$sort:{credits:-1}},{$limit:1}]);
[
  {
    _id: ObjectId("64f1fa0850af4978072f99dd"),
    name: 'Jane Smith',
    age: 22,
    gender: 'Female'
  }
]
```

- **{ $sort: { credits: -1 } }:** This is the first stage in the aggregation pipeline, and it uses the $sort operator to sort the documents in the collection based on the "credits" field in descending order (-1 indicates descending order). In other words, it arranges the courses from highest to lowest credits.
- **{ $limit: 1 }**: This is the second stage in the aggregation pipeline, and it uses the $limit operator to limit the number of documents passed to the next stage to just one document. In this case, it ensures that only the first (highest credits) course document is passed on to the next stage.

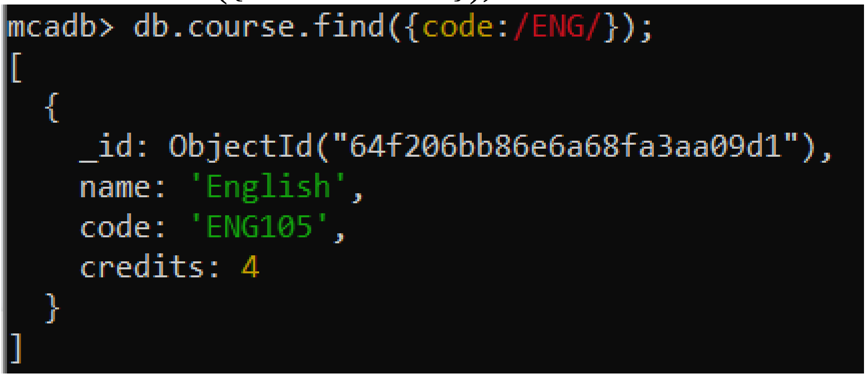
2. **Regular Expressions**:

// Find students whose names start with "J" db.students.find({ name: /^J/ });



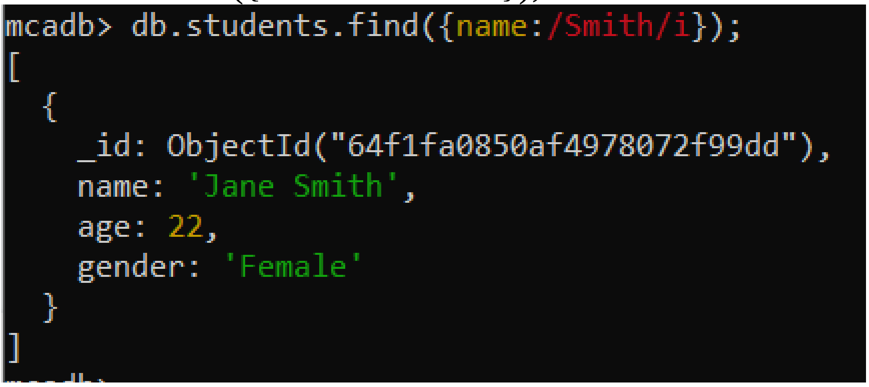
```
mcadb> db.students.find({name:/^J/});
[
  {
    _id: ObjectId("64f1fa0850af4978072f99dd"),
    name: 'Jane Smith',
    age: 22,
    gender: 'Female'
  },
  {
    _id: ObjectId("64f2071986e6a68fa3aa09d5"),
    name: 'John Doe',
    age: 20,
    gender: 'Male'
  }
]
```

// Find courses with codes containing "ENG"
db.course.find({ code: /ENG/ });



```
mcadb> db.course.find({code:/ENG/});
[
  {
    _id: ObjectId("64f206bb86e6a68fa3aa09d1"),
    name: 'English',
    code: 'ENG105',
    credits: 4
  }
]
```

// Case-insensitive search for students with "Smith" in their name
db.students.find({ name: /Smith/i });



```
mcadb> db.students.find({name:/Smith/i});
[
  {
    _id: ObjectId("64f1fa0850af4978072f99dd"),
    name: 'Jane Smith',
    age: 22,
    gender: 'Female'
  }
]
```

PROGRAM 15

AIM: Create a collection employee. Design and run 5 aggregate pipeline functions on employee collection. Use combinations of $match, $group, $sort and $project functions.

Aggregate PipeLine

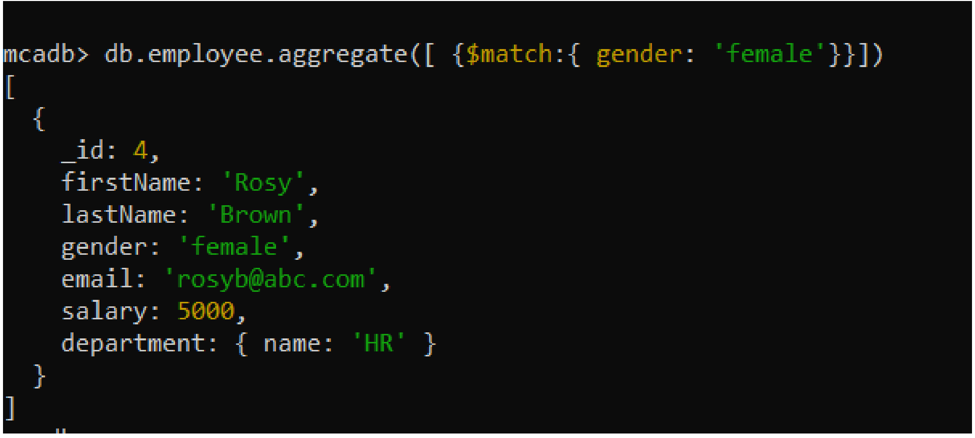
Different stages on employee collection:

Employees collection:

```
db.employee.insertMany([
   {
      _id:1,
      firstName: "John",
      lastName: "King",
      gender:'male',
      email: "john.king@abc.com",
      salary: 5000,
      department: {
                 "name":"HR" }},
   {
      _id:2,
      firstName: "Sachin",
      lastName: "T",
      gender:'male',
      email: "sachin.t@abc.com",
      salary: 8000,
      department: {
                 "name":"Finance" }},
   {
      _id:3,
      firstName: "James",
      lastName: "Bond",
      gender:'male',
      email: "jamesb@abc.com",
      salary: 7500,
      department: {
                 "name":"Marketing" }},
   {
      _id:4,
      firstName: "Rosy",
      lastName: "Brown",
      gender:'female',
      email: "rosyb@abc.com",
      salary: 5000,
      department: {
                 "name":"HR" }},
   {
      _id:5,
      firstName: "Kapil",
      lastName: "D",
      gender:'male',
      email: "kapil.d@abc.com",
      salary: 4500,
      department: {
                 "name":"Finance" }},
   {
      _id:6,
      firstName: "Amitabh",
      lastName: "B",
      gender:'male',
      email: "amitabh.b@abc.com",
      salary: 7000,
      department: {
                 "name":"Marketing" }}])
```

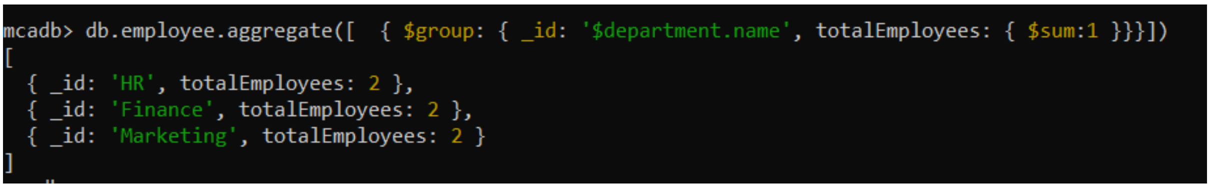
AGGREGATE PIPELINE COMMANDS:

1. db.employee.aggregate([{$match:{ gender: 'female'}}])



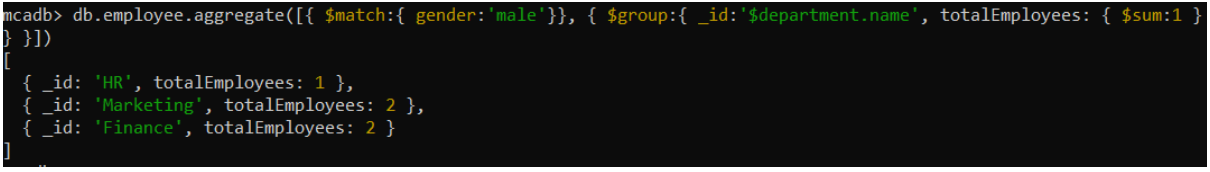
```
mcadb> db.employee.aggregate([ {$match:{ gender: 'female'}}])
[
  {
    _id: 4,
    firstName: 'Rosy',
    lastName: 'Brown',
    gender: 'female',
    email: 'rosyb@abc.com',
    salary: 5000,
    department: { name: 'HR' }
  }
]
```

2. db.employee.aggregate([{ $group: { _id: '$department.name', totalEmployees: { $sum:1 }}}])



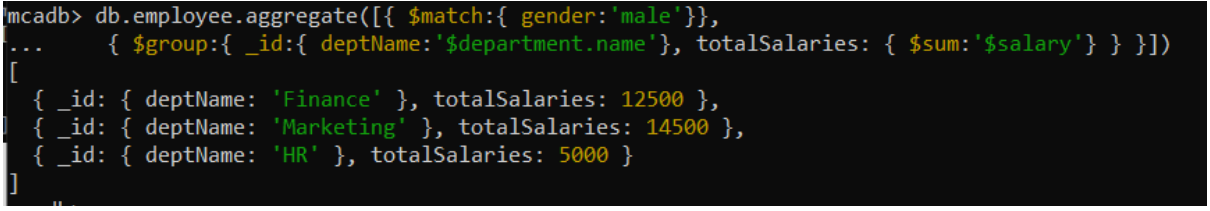
```
mcadb> db.employee.aggregate([  { $group: { _id: '$department.name', totalEmployees: { $sum:1 }}}])
[
  { _id: 'HR', totalEmployees: 2 },
  { _id: 'Finance', totalEmployees: 2 },
  { _id: 'Marketing', totalEmployees: 2 }
]
```

3. db.employee.aggregate([{ $match:{ gender:'male'}}, { $group:{ _id:'$department.name', totalEmployees: { $sum:1 } } }])



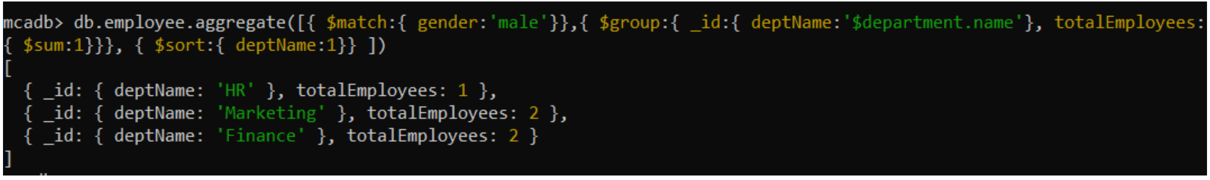
```
mcadb> db.employee.aggregate([{ $match:{ gender:'male'}}, { $group:{ _id:'$department.name', totalEmployees: { $sum:1 }
} }])
[
  { _id: 'HR', totalEmployees: 1 },
  { _id: 'Marketing', totalEmployees: 2 },
  { _id: 'Finance', totalEmployees: 2 }
]
```

4. db.employee.aggregate([{ $match:{ gender:'male'}}, { $group:{ _id:{ deptName:'$department.name'}, totalSalaries: { $sum:'$salary'} } }])



```
mcadb> db.employee.aggregate([{ $match:{ gender:'male'}},
...     { $group:{ _id:{ deptName:'$department.name'}, totalSalaries: { $sum:'$salary'} } }])
[
  { _id: { deptName: 'Finance' }, totalSalaries: 12500 },
  { _id: { deptName: 'Marketing' }, totalSalaries: 14500 },
  { _id: { deptName: 'HR' }, totalSalaries: 5000 }
]
```

5. db.employee.aggregate([{ $match:{ gender:'male'}},{ $group:{ _id:{ deptName:'$department.name'}, totalEmployees: { $sum:1}}}, { $sort:{ deptName:1}}])



```
mcadb> db.employee.aggregate([{ $match:{ gender:'male'}},{ $group:{ _id:{ deptName:'$department.name'}, totalEmployees:
{ $sum:1}}}, { $sort:{ deptName:1}} ])
[
  { _id: { deptName: 'HR' }, totalEmployees: 1 },
  { _id: { deptName: 'Marketing' }, totalEmployees: 2 },
  { _id: { deptName: 'Finance' }, totalEmployees: 2 }
]
```

CO 5

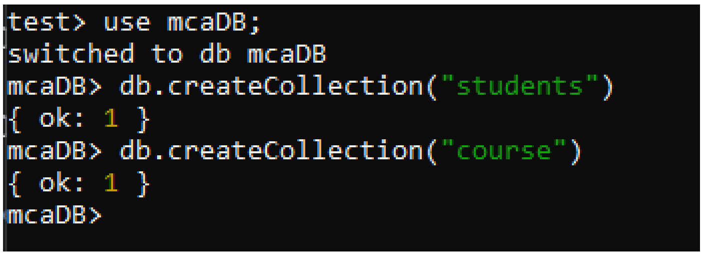
Understand the basic storage architecture of distributed file systems

PROGRAM 16

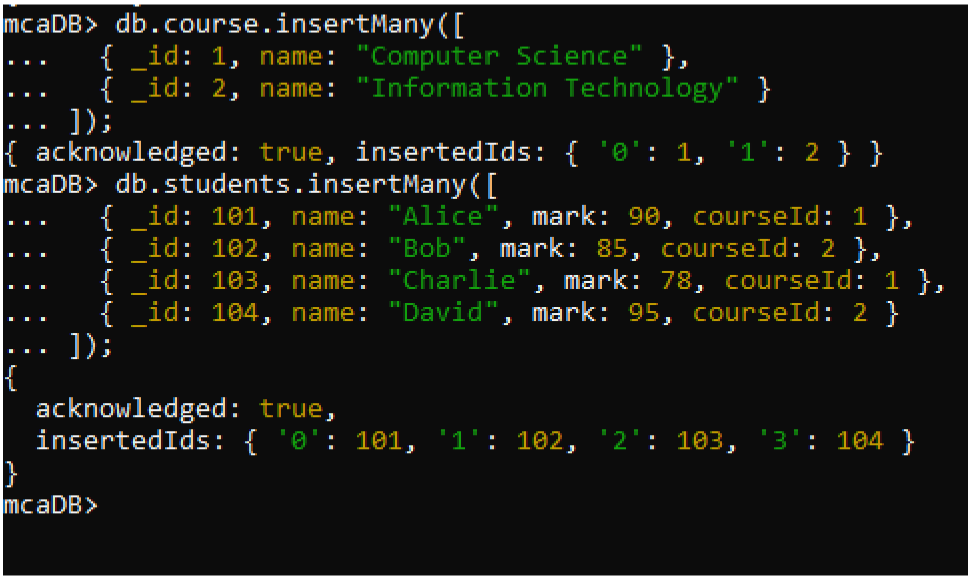
AIM

Build collections mcaDB documents students, course and perform shell commands to create indexing

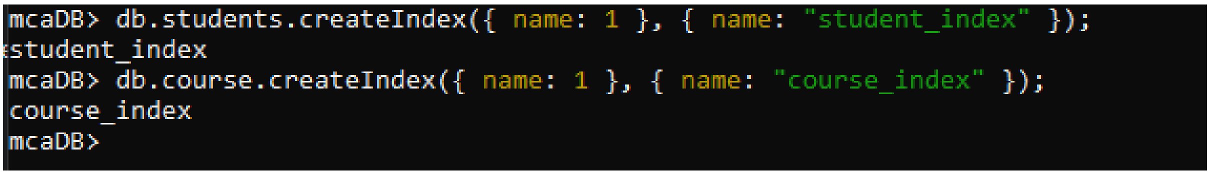
create a database



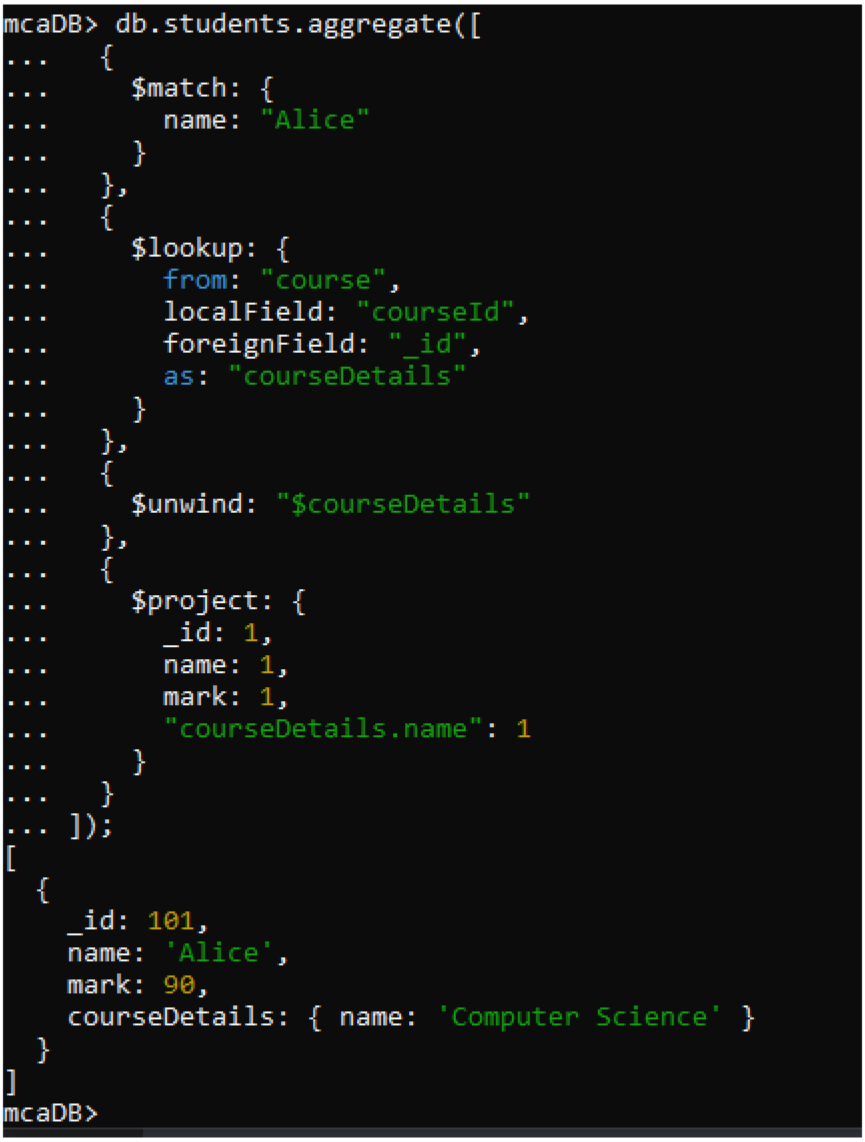
```
test> use mcaDB;
switched to db mcaDB
mcaDB> db.createCollection("students")
{ ok: 1 }
mcaDB> db.createCollection("course")
{ ok: 1 }
mcaDB>
```



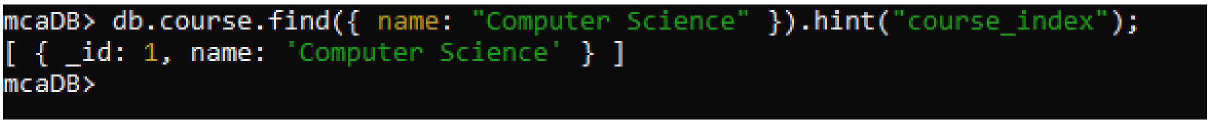
```
mcaDB> db.course.insertMany([
...    { _id: 1, name: "Computer Science" },
...    { _id: 2, name: "Information Technology" }
... ]);
{ acknowledged: true, insertedIds: { '0': 1, '1': 2 } }
mcaDB> db.students.insertMany([
...    { _id: 101, name: "Alice", mark: 90, courseId: 1 },
...    { _id: 102, name: "Bob", mark: 85, courseId: 2 },
...    { _id: 103, name: "Charlie", mark: 78, courseId: 1 },
...    { _id: 104, name: "David", mark: 95, courseId: 2 }
... ]);
{
  acknowledged: true,
  insertedIds: { '0': 101, '1': 102, '2': 103, '3': 104 }
}
mcaDB>
```



```
mcaDB> db.students.createIndex({ name: 1 }, { name: "student_index" });
student_index
mcaDB> db.course.createIndex({ name: 1 }, { name: "course_index" });
course_index
mcaDB>
```



```
mcaDB> db.students.aggregate([
...   {
...     $match: {
...       name: "Alice"
...     }
...   },
...   {
...     $lookup: {
...       from: "course",
...       localField: "courseId",
...       foreignField: "_id",
...       as: "courseDetails"
...     }
...   },
...   {
...     $unwind: "$courseDetails"
...   },
...   {
...     $project: {
...       _id: 1,
...       name: 1,
...       mark: 1,
...       "courseDetails.name": 1
...     }
...   }
... ]);
[
  {
    _id: 101,
    name: 'Alice',
    mark: 90,
    courseDetails: { name: 'Computer Science' }
  }
]
mcaDB>
```



```
mcaDB> db.course.find({ name: "Computer Science" }).hint("course_index");
[ { _id: 1, name: 'Computer Science' } ]
mcaDB>
```

CO 6

Design and deployment of NoSQL databases with real time requirements.

PROGRAM 17

AIM

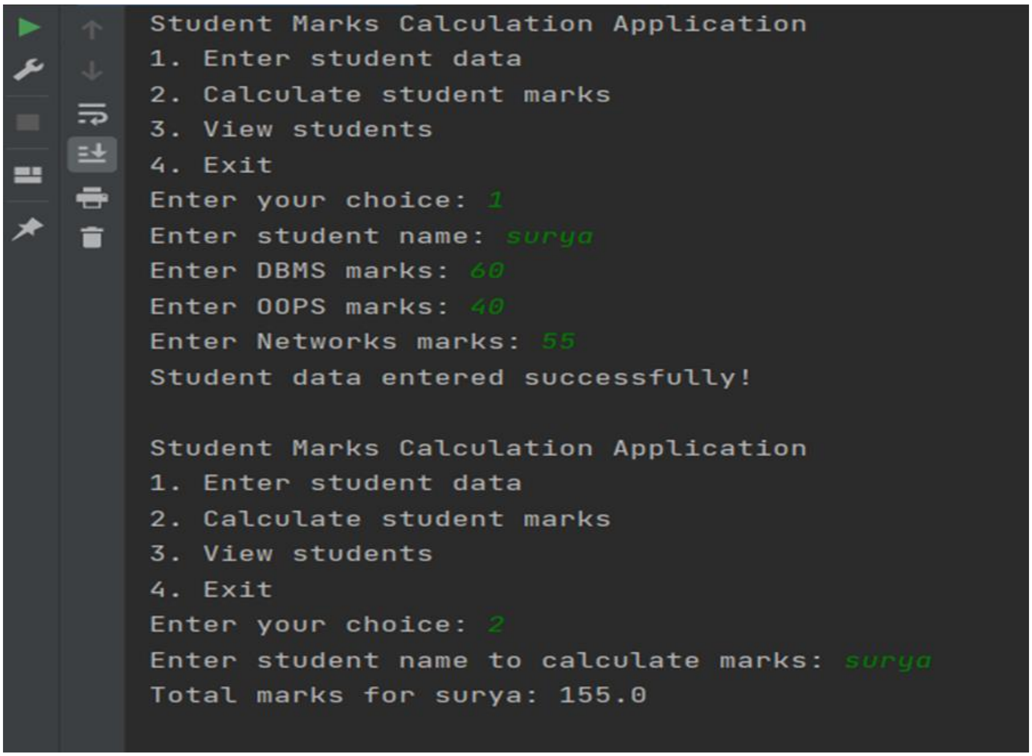
Develop students’ marks calculation applications using Python and MongoDB

CODE

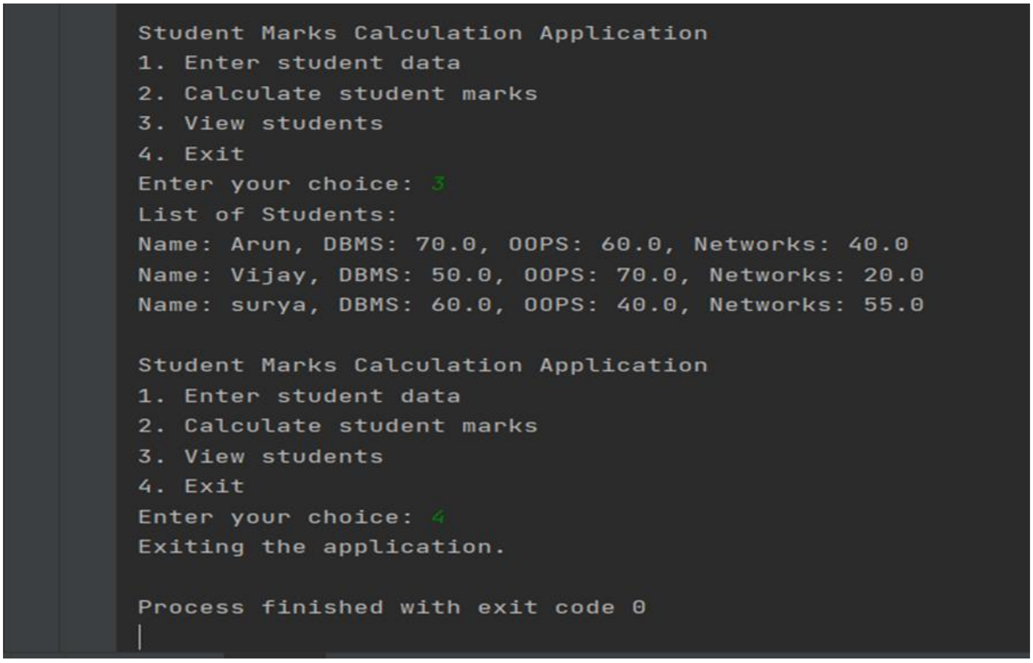
```
import pymongo


# Connect to MongoDB

client = pymongo.MongoClient("mongodb://localhost:27017/")

db = client["student_marks"]

collection = db["students"]


def enter_student_data():

    name = input("Enter student name: ")

    dbms = float(input("Enter DBMS marks: "))

    oops = float(input("Enter OOPS marks: "))

    networks = float(input("Enter Networks marks: "))


    student_data = {

    "name": name,

   "dbms": dbms,

    "oops": oops,
    "networks": networks
    }
    collection.insert_one(student_data)
    print("Student data entered successfully!")

def calculate_student_marks():
    name = input("Enter student name to calculate marks: ")
    student = collection.find_one({"name": name})


    if student:
    total_marks = student["dbms"] + student["oops"] + student["networks"]
    print(f"Total marks for {name}: {total_marks}")
    else:
    print(f"Student '{name}' not found!")

def view_students():
    print("List of Students:")
    for student in collection.find():
    print(
    f"Name: {student['name']}, DBMS: {student['dbms']}, OOPS: {student['oops']}, Networks: {student['networks']}")


def main():
    while True:
```

```
    print("\nStudent Marks Calculation Application")
    print("1. Enter student data")
    print("2. Calculate student marks")
    print("3. View students")
    print("4. Exit")

    choice = input("Enter your choice: ")

    if choice == "1":
    enter_student_data()
    elif choice == "2":
    calculate_student_marks()
    elif choice == "3":
    view_students()
    elif choice == "4":
    print("Exiting the application.")
      break
    else:
    print("Invalid choice. Please try again.")
if __name__ == "__main__":
    main()
```

OUTPUT



```
Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice: 1
Enter student name: surya
Enter DBMS marks: 60
Enter OOPS marks: 40
Enter Networks marks: 55
Student data entered successfully!

Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice: 2
Enter student name to calculate marks: surya
Total marks for surya: 155.0
```



```
Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice: 3
List of Students:
Name: Arun, DBMS: 70.0, OOPS: 60.0, Networks: 40.0
Name: Vijay, DBMS: 50.0, OOPS: 70.0, Networks: 20.0
Name: surya, DBMS: 60.0, OOPS: 40.0, Networks: 55.0

Student Marks Calculation Application
1. Enter student data
2. Calculate student marks
3. View students
4. Exit
Enter your choice: 4
Exiting the application.

Process finished with exit code 0
```