PRATICLE QUESTION BANK

SQL & PLSQL

1) **For the given Library database**
BOOK (Book_ISBN [PK], Title[Not Null], Publisher_ Name, price[Check Price>0], Date_Of_Publication,Book_Copy),
BOOK_AUTHORS (Book_ISBN [PK,FK]Author_Name [PK], Author_City) Solve the following
a) Create view BOOK_AUTHOR_INFO consisting Book_ISBN, Title from BOOK Table and Author_Name from BOOK_AUTHORS Table in ascending order of ISBN no.
b) Create an index on Book_Author on table on attribute "Author_Name".
c) Create table Book_Auto_Increment (BookID int Auto_increament=100, Book Name) insert five records in table.
d) Delete the book from Book table written by Author 'Korth'.
e) Select Book Names from table Book whose copies are in between 10 to 15.

OUTPUT:

```
mysql> create database Library;
Query OK, 1 row affected (0.04 sec)

mysql> use Library
Database changed

mysql> CREATE TABLE BOOK (Book_ISBN VARCHAR(20) PRIMARY KEY,Title VARCHAR(255) NOT
NULL,Publisher_Name VARCHAR(255),Price DECIMAL(10, 2) CHECK (Price > 0),Date_Of_Publication
DATE,Book_Copy INT);
Query OK, 0 rows affected (0.06 sec)

mysql> CREATE TABLE BOOK_AUTHORS (Book_ISBN VARCHAR(20),Author_Name
VARCHAR(255),Author_City VARCHAR(255),PRIMARY KEY (Book_ISBN, Author_Name),FOREIGN KEY
(Book_ISBN) REFERENCES BOOK(Book_ISBN));
Query OK, 0 rows affected (0.04 sec)

mysql> INSERT INTO BOOK (Book_ISBN, Title, Publisher_Name, Price, Date_Of_Publication,
Book_Copy) VALUES('978-3-16-148410-0', 'Book Title 1', 'Publisher A', 100, '2021-01-15', 12),
    -> ('978-1-23-456789-7', 'Book Title 2', 'Korth', 150, '2020-03-22', 10),
    -> ('978-0-12-345678-9', 'Book Title 3', 'Publisher C', 150, '2022-06-30', 20),
    -> ('978-3-16-148410-1', 'Book Title 4', 'Publisher A', 220, '2019-07-18', 8),
    -> ('978-1-23-456789-8', 'Book Title 5', 'Publisher B', 250, '2014-09-10', 5),
    -> ('978-0-12-345678-0', 'Book Title 6', 'Publisher C', 185, '2013-05-20', 14),
    -> ('978-3-16-148410-2', 'Book Title 7', 'Publisher A', 300, '2012-08-12', 9),
    -> ('978-1-23-456789-9', 'Book Title 8', 'Publisher B', 200, '2020-11-01', 11),
    -> ('978-0-12-345678-1', 'Book Title 9', 'Publisher C', 129, '2022-12-15', 15),
    -> ('978-3-16-148410-3', 'Book Title 10', 'Publisher A', 240, '2018-04-25', 7);
Query OK, 10 rows affected (0.02 sec)
Records: 10  Duplicates: 0  Warnings: 0

mysql> INSERT INTO BOOK_AUTHORS (Book_ISBN, Author_Name, Author_City) VALUES
```

```
    -> ('978-3-16-148410-0', 'Author A', 'Pune'),
    -> ('978-1-23-456789-7', 'Korth', 'Nashik'),
    -> ('978-0-12-345678-9', 'Author C', 'Mumbai'),
    -> ('978-3-16-148410-1', 'Author D', 'Pune'),
    -> ('978-1-23-456789-8', 'Author E', 'Bhiwandi'),
    -> ('978-0-12-345678-0', 'Author F', 'Shirdi'),
    -> ('978-3-16-148410-2', 'Author G', 'Nashik'),
    -> ('978-1-23-456789-9', 'Author H', 'Pune'),
    -> ('978-0-12-345678-1', 'Author I', 'Mumbai'),
    -> ('978-3-16-148410-3', 'Author J', 'Baramati');
Query OK, 10 rows affected (0.01 sec)
Records: 10  Duplicates: 0  Warnings: 0
```

a)
```
mysql> CREATE VIEW BOOK_AUTHOR_INFO AS SELECT b.Book_ISBN, b.Title, a.Author_Name
FROM BOOK b JOIN BOOK_AUTHORS a ON b.Book_ISBN = a.Book_ISBN ORDER BY b.Book_ISBN
ASC;
Query OK, 0 rows affected (0.02 sec)

mysql> SELECT * FROM BOOK_AUTHOR_INFO;
+-------------------+---------------+-------------+
| Book_ISBN         | Title         | Author_Name |
+-------------------+---------------+-------------+
| 978-0-12-345678-0 | Book Title 6  | Author F    |
| 978-0-12-345678-1 | Book Title 9  | Author I    |
| 978-0-12-345678-9 | Book Title 3  | Author C    |
| 978-1-23-456789-7 | Book Title 2  | Korth       |
| 978-1-23-456789-8 | Book Title 5  | Author E    |
| 978-1-23-456789-9 | Book Title 8  | Author H    |
| 978-3-16-148410-0 | Book Title 1  | Author A    |
| 978-3-16-148410-1 | Book Title 4  | Author D    |
| 978-3-16-148410-2 | Book Title 7  | Author G    |
| 978-3-16-148410-3 | Book Title 10 | Author J    |
+-------------------+---------------+-------------+
10 rows in set (0.02 sec)
```

b)
```
mysql> CREATE INDEX idx_author_name ON BOOK_AUTHORS(Author_Name);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0
mysql> SHOW INDEX FROM BOOK_AUTHORS;
+--------------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------
+------+------------+---------+---------------+---------+------------+
| Table        | Non_unique | Key_name        | Seq_in_index | Column_name | Collation |
Cardinality | Sub_part | Packed | Null | Index_type | Comment | Index_comment | Visible |
Expression |
+--------------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------
+------+------------+---------+---------------+---------+------------+
| book_authors |          0 | PRIMARY         |            1 | Book_ISBN   | A         |          10 |     NULL |
NULL |      | BTREE      |         |               | YES     | NULL       |
| book_authors |          0 | PRIMARY        |            2 | Author_Name | A         |          10 |     NULL |
NULL |      | BTREE      |         |               | YES     | NULL       |
| book_authors |          1 | idx_author_name |            1 | Author_Name | A         |          10 |
NULL |   NULL |      | BTREE      |         |               | YES     | NULL       |
+--------------+------------+-----------------+--------------+-------------+-----------+-------------+----------+--------
+------+------------+---------+---------------+---------+------------+
3 rows in set (0.03 sec)
```

```
c) mysql> CREATE TABLE Book_Auto_Increment (BookID INT AUTO_INCREMENT PRIMARY
KEY,BookName VARCHAR(255)) AUTO_INCREMENT=100;
Query OK, 0 rows affected (0.03 sec)

mysql> INSERT INTO Book_Auto_Increment (BookName) VALUES
    -> ('Book 1'),
    -> ('Book 2'),
    -> ('Book 3'),
    -> ('Book 4'),
    -> ('Book 5');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0

d) mysql> DELETE FROM BOOK_AUTHORS WHERE Author_Name = 'Korth';
Query OK, 1 row affected (0.01 sec)

mysql> DELETE FROM BOOK WHERE Book_ISBN IN (SELECT Book_ISBN FROM BOOK_AUTHORS
WHERE Author_Name = 'Korth');
Query OK, 0 rows affected (0.00 sec)

e) mysql> SELECT Title FROM BOOK WHERE Book_Copy BETWEEN 10 AND 15;
+--------------+
| Title        |
+--------------+
| Book Title 6 |
| Book Title 9 |
| Book Title 2 |
| Book Title 8 |
| Book Title 1 |
+--------------+
5 rows in set (0.00 sec)
```

2) a) Select Book_ISBN, Title, Author_Name from relations Book and Book_Authors INNER JOIN on attribute Book_ISBN.

b) Select Book_ISBN, Title, Publisher, Author_Name from relations Book and Book_Authors LEFT OUTER JOIN on attribute Book_ISBN.

c) Select Book_ISBN, Title, Publisher, Author_Name from relations Book and Book_Authors RIGHT OUTER JOIN on attribute Book_ISBN.

d) Select Book_ISBN, Title from relation Book whose author is living in City ='Pune'.

e) Select Book_ISBN, Title from relation Book, which written by more than 2 Authors.

OUTPUT:

a) mysql> SELECT b.Book_ISBN, b.Title, a.Author_Name FROM BOOK b INNER JOIN BOOK_AUTHORS a ON b.Book_ISBN = a.Book_ISBN;

```
+-------------------+---------------+-------------+
| Book_ISBN         | Title         | Author_Name |
+-------------------+---------------+-------------+
| 978-3-16-148410-0 | Book Title 1  | Author A    |
| 978-0-12-345678-9 | Book Title 3  | Author C    |
| 978-3-16-148410-1 | Book Title 4  | Author D    |
| 978-1-23-456789-8 | Book Title 5  | Author E    |
| 978-0-12-345678-0 | Book Title 6  | Author F    |
| 978-3-16-148410-2 | Book Title 7  | Author G    |
| 978-1-23-456789-9 | Book Title 8  | Author H    |
| 978-0-12-345678-1 | Book Title 9  | Author I    |
| 978-3-16-148410-3 | Book Title 10 | Author J    |
+-------------------+---------------+-------------+
9 rows in set (0.00 sec)
```

b) mysql> SELECT b.Book_ISBN, b.Title, b.Publisher_Name, a.Author_Name FROM BOOK b LEFT OUTER JOIN BOOK_AUTHORS a ON b.Book_ISBN = a.Book_ISBN;

```
+-------------------+---------------+----------------+-------------+
| Book_ISBN         | Title         | Publisher_Name | Author_Name |
+-------------------+---------------+----------------+-------------+
| 978-0-12-345678-0 | Book Title 6  | Publisher C    | Author F    |
| 978-0-12-345678-1 | Book Title 9  | Publisher C    | Author I    |
| 978-0-12-345678-9 | Book Title 3  | Publisher C    | Author C    |
| 978-1-23-456789-7 | Book Title 2  | Korth          | NULL        |
| 978-1-23-456789-8 | Book Title 5  | Publisher B    | Author E    |
| 978-1-23-456789-9 | Book Title 8  | Publisher B    | Author H    |
| 978-3-16-148410-0 | Book Title 1  | Publisher A    | Author A    |
| 978-3-16-148410-1 | Book Title 4  | Publisher A    | Author D    |
| 978-3-16-148410-2 | Book Title 7  | Publisher A    | Author G    |
| 978-3-16-148410-3 | Book Title 10 | Publisher A    | Author J    |
+-------------------+---------------+----------------+-------------+
10 rows in set (0.00 sec)
```

c) mysql> SELECT b.Book_ISBN, b.Title, b.Publisher_Name, a.Author_Name FROM BOOK b RIGHT OUTER JOIN BOOK_AUTHORS a ON b.Book_ISBN = a.Book_ISBN;

```
+-------------------+---------------+----------------+-------------+
| Book_ISBN         | Title         | Publisher_Name | Author_Name |
+-------------------+---------------+----------------+-------------+
| 978-3-16-148410-0 | Book Title 1  | Publisher A    | Author A    |
| 978-0-12-345678-9 | Book Title 3  | Publisher C    | Author C    |
| 978-3-16-148410-1 | Book Title 4  | Publisher A    | Author D    |
| 978-1-23-456789-8 | Book Title 5  | Publisher B    | Author E    |
| 978-0-12-345678-0 | Book Title 6  | Publisher C    | Author F    |
| 978-3-16-148410-2 | Book Title 7  | Publisher A    | Author G    |
| 978-1-23-456789-9 | Book Title 8  | Publisher B    | Author H    |
| 978-0-12-345678-1 | Book Title 9  | Publisher C    | Author I    |
| 978-3-16-148410-3 | Book Title 10 | Publisher A    | Author J    |
+-------------------+---------------+----------------+-------------+
9 rows in set (0.00 sec)
```

d) mysql> SELECT b.Book_ISBN, b.Title FROM BOOK b JOIN BOOK_AUTHORS a ON b.Book_ISBN = a.Book_ISBN WHERE a.Author_City = 'Pune';

```
+-------------------+--------------+
| Book_ISBN         | Title        |
+-------------------+--------------+
| 978-1-23-456789-9 | Book Title 8 |
| 978-3-16-148410-0 | Book Title 1 |
| 978-3-16-148410-1 | Book Title 4 |
+-------------------+--------------+
3 rows in set (0.00 sec)
```

e) mysql> SELECT b.Book_ISBN, b.Title FROM BOOK b JOIN BOOK_AUTHORS a ON b.Book_ISBN = a.Book_ISBN GROUP BY b.Book_ISBN, b.Title HAVING COUNT(a.Author_Name) > 2;

```
Empty set (0.00 sec)
```

3)i) Display name of publishers as per no of books published by them in ascending order.

ii) Get publisher names who published at least one book written by author name like 'K%'.
iii) Get book name and Authors names where book written by maximum authors.
iv) Get publisher names accordingly books published alphabetically
v) Find the no of books published in 01 Jan 2014 to till date.

OUTPUT:

```
i) mysql> SELECT Publisher_Name, COUNT(*) AS NumberOfBooks FROM BOOK GROUP BY
Publisher_Name ORDER BY NumberOfBooks ASC;
+----------------+---------------+
| Publisher_Name | NumberOfBooks |
+----------------+---------------+
| Korth          |             1 |
| Publisher B    |             2 |
| Publisher C    |             3 |
| Publisher A    |             4 |
+----------------+---------------+
4 rows in set (0.00 sec)

ii) mysql> SELECT DISTINCT b.Publisher_Name FROM BOOK b JOIN BOOK_AUTHORS a ON
b.Book_ISBN = a.Book_ISBN WHERE a.Author_Name LIKE 'K%';
Empty set (0.01 sec)

iii) mysql> SELECT b.Title, a.Author_Name FROM BOOK b JOIN BOOK_AUTHORS a ON b.Book_ISBN =
a.Book_ISBN WHERE b.Book_ISBN IN ( SELECT Book_ISBN FROM BOOK_AUTHORS GROUP BY
Book_ISBN HAVING COUNT(Author_Name) = ( SELECT MAX(AuthorCount) FROM ( SELECT
COUNT(Author_Name) AS AuthorCount FROM BOOK_AUTHORS GROUP BY Book_ISBN ) AS
AuthorCounts ) );
+---------------+-------------+
| Title         | Author_Name |
+---------------+-------------+
| Book Title 1  | Author A    |
| Book Title 3  | Author C    |
| Book Title 4  | Author D    |
| Book Title 5  | Author E    |
| Book Title 6  | Author F    |
| Book Title 7  | Author G    |
| Book Title 8  | Author H    |
| Book Title 9  | Author I    |
| Book Title 10 | Author J    |
+---------------+-------------+
9 rows in set (0.01 sec)

iv) mysql> SELECT DISTINCT Publisher_Name FROM BOOK ORDER BY Publisher_Name ASC;
+----------------+
| Publisher_Name |
+----------------+
| Korth          |
| Publisher A    |
| Publisher B    |
| Publisher C    |
+----------------+
4 rows in set (0.00 sec)

v) mysql> SELECT COUNT(*) AS NumberOfBooks FROM BOOK WHERE Date_Of_Publication >=
'2014-01-01';
+---------------+
| NumberOfBooks |
+---------------+
|             8 |
+---------------+
1 row in set (0.01 sec)
```

4)Consider INSURANCE database with following schema : person(driver-id, name, address) car(license, model, year) accident (report - no, date, location) owns(driver-id,license) participated(driver-id,car,report-no,damage-amount) Write a query in SQL for following requirements :

i) Find the total no. of people who owned cars that were involved in accidents in 2016.

ii) Retrieve the name of person whose address contains Pune.

iii) Find the name of persons having more than two cars.

OUTPUT:

```
mysql> CREATE DATABASE INSURANCE;

Query OK, 1 row affected (0.00 sec)


mysql> USE INSURANCE;

Database changed


mysql> CREATE TABLE person ( driver_id INT PRIMARY KEY, name VARCHAR(255) NOT NULL, address
VARCHAR(255) );

Query OK, 0 rows affected (0.02 sec)


mysql> CREATE TABLE car ( license VARCHAR(20) PRIMARY KEY, model VARCHAR(100) NOT NULL, year INT
CHECK (year > 1885));

Query OK, 0 rows affected (0.01 sec)
```

```
mysql> CREATE TABLE accident ( report_no INT PRIMARY KEY, date DATE NOT NULL, location VARCHAR(255)
);

Query OK, 0 rows affected (0.01 sec)


mysql> CREATE TABLE owns ( driver_id INT, license VARCHAR(20), PRIMARY KEY (driver_id, license),
FOREIGN KEY (driver_id) REFERENCES person(driver_id), FOREIGN KEY (license) REFERENCES car(license) );

Query OK, 0 rows affected (0.03 sec)


mysql> CREATE TABLE participated ( driver_id INT, car VARCHAR(20), report_no INT, damage_amount
DECIMAL(10, 2), PRIMARY KEY (driver_id, car, report_no), FOREIGN KEY (driver_id) REFERENCES
person(driver_id), FOREIGN KEY (car) REFERENCES car(license), FOREIGN KEY (report_no) REFERENCES
accident(report_no) );

Query OK, 0 rows affected (0.04 sec)


mysql> INSERT INTO person (driver_id, name, address) VALUES(1, 'Alice Smith', '123 Elm St, Springfield'),(2,
'Bob Johnson', '456 Oak St, Springfield'),(3, 'Charlie Brown', '789 Pine St, Springfield'),(4, 'David Wilson',
'321 Maple St, Springfield'),(5, 'Eva Green', '654 Cedar St, Springfield');

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> INSERT INTO car (license, model, year) VALUES('ABC123', 'Toyota Camry', 2020),('XYZ456', 'Honda
Accord', 2019),('LMN789', 'Ford Focus', 2018),('JKL012', 'Chevrolet Malibu', 2021),('QRS345', 'Nissan
Altima', 2022);

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> INSERT INTO accident (report_no, date, location) VALUES(1, '2023-01-15', '1st Ave & Main St'),(2,
'2023-02-20', '2nd Ave & Oak St'),(3, '2023-03-05', '3rd Ave & Pine St'),(4, '2016-04-10', '4th Ave & Maple
St'),(5, '2016-05-12', '5th Ave & Cedar St');

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> INSERT INTO owns (driver_id, license) VALUES(1, 'ABC123'),(2, 'XYZ456'),(3, 'LMN789'),(4,
'JKL012'),(5, 'QRS345');

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO participated (driver_id, car, report_no, damage_amount) VALUES(1, 'ABC123', 1, 1500.00),(2, 'XYZ456', 2, 2300.50),(3, 'LMN789', 3, 1200.75),(4, 'JKL012', 4, 3000.00),(5, 'QRS345', 5, 1750.25);

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0
```

```
i) mysql> SELECT COUNT(DISTINCT o.driver_id) AS TotalPeople FROM owns o JOIN participated p ON o.driver_id = p.driver_id JOIN accident a ON p.report_no = a.report_no WHERE YEAR(a.date) = 2016;

+-------------+

| TotalPeople |

+-------------+

|           2 |

+-------------+

1 row in set (0.01 sec)

ii) mysql> SELECT name FROM person WHERE address LIKE '%Pune%';

Empty set (0.00 sec)

iii) mysql> SELECT p.name FROM person p JOIN owns o ON p.driver_id = o.driver_id GROUP BY p.driver_id, p.name HAVING COUNT(o.license) > 2;

Empty set (0.00 sec)
```

5) Implement MySQL database connectivity with Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.

OUTPUT:

```
package JavaSQL;


import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

import java.util.Scanner;


public class JavaSQL {

   String dbname;

   String Tbname;
   String attributes;
   String insert;
   String upd;
   String del;
   Scanner sc = new Scanner(System.in);

   public static void main(String[] args) {
     JavaSQL s1 = new JavaSQL();
     Scanner sc = new Scanner(System.in);
     do {
       System.out.println("MENU");
       System.out.println("Create Database");
       System.out.println("Create Table");
       System.out.println("Insert");
       System.out.println("Update");
       System.out.println("Delete");
       System.out.println("Exit");
       int n = sc.nextInt();
       sc.nextLine();  // Clear the buffer

       switch (n) {
         case 1:
           s1.createDatabase();
           break;
         case 2:
           s1.createTable();
           break;
         case 3:
           s1.insertData();
           break;
         case 4:
           s1.updateData();
```

```
            break;
        case 5:
            s1.deleteData();
            break;
        case 6:
            System.out.println("Exiting...");
            System.exit(6);
        default:
            System.out.println("Invalid input");
        }
    } while (true);
}

public void createDatabase() {
    try {
        String url = "jdbc:mysql://localhost:3306/";
        String username = "root";
        String password = "Vishal@4983";
        Connection con = DriverManager.getConnection(url, username, password);
        Statement st = con.createStatement();

        System.out.println("Enter the database name:");
        dbname = sc.nextLine();

        String Query = "CREATE database " + dbname;
        st.execute(Query);
        System.out.println("Database created successfully");
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

```java
public void createTable() {
    try {
        String url = "jdbc:mysql://localhost:3306/" + dbname;
        String username = "root";
        String password = "Vishal@4983";
        Connection con = DriverManager.getConnection(url, username, password);
        Statement st = con.createStatement();

        System.out.println("Enter the Table name:");
        Tbname = sc.nextLine();

        System.out.println("Enter the attributes and the datatype ");
        attributes = sc.nextLine();

        String Query = "CREATE Table " + Tbname + "(" + attributes + ")";
        st.execute(Query);
        System.out.println("Table created successfully");
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void insertData() {
    try {
        String url = "jdbc:mysql://localhost:3306/" + dbname;
        String username = "root";
        String password = "Vishal@4983";
        Connection con = DriverManager.getConnection(url, username, password);
        Statement st = con.createStatement();
        System.out.println("Enter the number of records you wish to insert:");
        int n = sc.nextInt();
        sc.nextLine();  // Clear buffer
        for (int i = 1; i <= n; i++) {
            System.out.println("Enter record " + i + " inside table " + Tbname);
            insert = sc.nextLine();
            String Query = "INSERT INTO " + Tbname + " VALUES(" + insert + ")";
            st.execute(Query);
            System.out.println("Record inserted successfully");
        }
        con.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}

public void updateData() {
    try {
        String url = "jdbc:mysql://localhost:3306/" + dbname;
        String username = "root";
        String password = "Vishal@4983";
        Connection con = DriverManager.getConnection(url, username, password);
        Statement st = con.createStatement();

        System.out.println("Enter the attribute and its new value (e.g., name = 'John')");
        upd = sc.nextLine();
        System.out.println("Enter the condition (e.g., id = 1)");
        String condition = sc.nextLine();

        String Query = "UPDATE " + Tbname + " SET " + upd + " WHERE " + condition;
        st.executeUpdate(Query);
        System.out.println("Record updated successfully");
```

```
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public void deleteData() {
        try {
            String url = "jdbc:mysql://localhost:3306/" + dbname;
            String username = "root";
            String password = "Vishal@4983";
            Connection con = DriverManager.getConnection(url, username, password);
            Statement st = con.createStatement();

            System.out.println("Enter the condition for deletion (e.g., id = 1):");
            del = sc.nextLine();

            String Query = "DELETE FROM " + Tbname + " WHERE " + del;
            st.execute(Query);
            System.out.println("Record deleted successfully");
            con.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Output:

```
MENU
Create Database
Create Table
Insert
```

```
Update
Delete
Exit
1
Enter the database name:
Connectivity
Database created successfully
MENU
Create Database
Create Table
Insert
Update
Delete
Exit
2
Enter the Table name:
Students
Enter the attributes and the datatype
Student_ID int,Name varchar(20),Phone_no int,Address varchar(20)
Table created successfully
MENU
Create Database
Create Table
Insert
Update
Delete
Exit
3
Enter the number of records you wish to insert:
4
Enter record 1 inside table Students
12,'Vishal ',321654987,'Pune'
```

```
Record inserted successfully

Enter record 2 inside table Students

13,'Raj,123456789,'Pune'

Record inserted successfully

Enter record 3 inside table Students

14,'Ankit’,456123789,'Nashik'

Record inserted successfully

Enter record 4 inside table Students

15,'Sam,789654123,'Baramati'

Record inserted successfully

MENU

Create Database

Create Table

Insert

Update

Delete

Exit

4

Enter the attribute and its new value (e.g., name = 'John')

Address = 'Pune'

Enter the condition (e.g., id = 1)

Student_ID = 15

Record updated successfully

MENU

Create Database

Create Table

Insert

Update

Delete

Exit

5

Enter the condition for deletion (e.g., id = 1):
```

```
Student_ID = 13

Record deleted successfully

MENU

Create Database

Create Table

Insert

Update

Delete

Exit

6

Invalid input

MENU

Create Database

Create Table

Insert

Update

Delete

Exit

0

Exiting...
```

6) For the given Employee database
EmployeeInfo(EmpID[PK],EmpFname,EmpLname,Department,Project,Address,DOB,Ge nder)
EmployeePosition(EmpID[FK],EmpPosition,DateOfJoining,Salary)

i. Write a query to fetch the EmpFname from the EmployeeInfo table in the upper case and use the ALIAS name as EmpName.

ii. Write a query to fetch the number of employees working in the department 'HR'.

iii.Write q query to find all the employees whose salary is between 50000 to 100000

iv.Write a query to find the names of employees that begin with 'S'

v. Write a query to fetch top N records

OUTPUT:

```
mysql> CREATE DATABASE Employee;

Query OK, 1 row affected (0.00 sec)
```

```
mysql> USE Employee

Database changed

mysql> CREATE TABLE EmployeeInfo ( EmpID INT PRIMARY KEY, EmpFname VARCHAR(50) NOT NULL,
EmpLname VARCHAR(50) NOT NULL, Department VARCHAR(50), Project VARCHAR(50), Address
VARCHAR(255), DOB DATE, Gender VARCHAR(10) );

Query OK, 0 rows affected (0.01 sec)


mysql> CREATE TABLE EmployeePosition ( EmpID INT, EmpPosition VARCHAR(50), DateOfJoining DATE,
Salary DECIMAL(10, 2), PRIMARY KEY (EmpID, EmpPosition), FOREIGN KEY (EmpID) REFERENCES
EmployeeInfo(EmpID) );

Query OK, 0 rows affected (0.02 sec)


mysql> INSERT INTO EmployeeInfo (EmpID, EmpFname, EmpLname, Department, Project, Address, DOB,
Gender) VALUES(1, 'Alice', 'Smith', 'HR', 'Recruitment', '123 Main St, Springfield', '1990-05-15', 'Female'),(2,
'Bob', 'Johnson', 'IT', 'Web Development', '456 Oak St, Springfield', '1985-03-22', 'Male'),(3, 'Charlie',
'Brown', 'Finance', 'Accounting', '789 Pine St, Springfield', '1992-07-30', 'Male'),(4, 'David', 'Wilson',
'Marketing', 'Advertising', '321 Maple St, Springfield', '1988-12-05', 'Male'),(5, 'Eva', 'Green', 'IT', 'Mobile
Development', '654 Cedar St, Springfield', '1995-09-18', 'Female');

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> INSERT INTO EmployeePosition (EmpID, EmpPosition, DateOfJoining, Salary) VALUES(1, 'HR
Manager', '2020-01-10', 60000.00),(2, 'Software Engineer', '2019-02-15', 75000.00),(3, 'Accountant', '2018-
03-20', 55000.00),(4, 'Marketing Specialist', '2021-04-25', 50000.00),(5, 'Mobile Developer', '2022-05-30',
70000.00);

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0


i)mysql> SELECT UPPER (EmpFname) AS EmpName FROM EmployeeInfo;

+---------+

| EmpName |

+---------+

| ALICE   |

| BOB     |

| CHARLIE |
| DAVID   |
| EVA     |
+---------+
5 rows in set (0.01 sec)
```

```
ii)mysql> SELECT COUNT(*) AS NumberOfEmployees FROM EmployeeInfo WHERE Department = 'HR';
+-------------------+
| NumberOfEmployees |
+-------------------+
|                 1 |
+-------------------+
1 row in set (0.00 sec)
```

```
iii)mysql> SELECT e.EmpFname, e.EmpLname, p.Salary FROM EmployeeInfo e JOIN EmployeePosition p ON
e.EmpID = p.EmpID WHERE p.Salary BETWEEN 50000 AND 100000;
+----------+----------+----------+
| EmpFname | EmpLname | Salary   |
+----------+----------+----------+
| Alice    | Smith    | 60000.00 |
| Bob      | Johnson  | 75000.00 |
| Charlie  | Brown    | 55000.00 |
| David    | Wilson   | 50000.00 |
| Eva      | Green    | 70000.00 |
+----------+----------+----------+
5 rows in set (0.00 sec)
```

```
iv)mysql> SELECT EmpFname, EmpLname FROM EmployeeInfo WHERE EmpFname LIKE 'S%';
Empty set (0.00 sec)
```

```
v)mysql> SELECT *FROM EmployeeInfo ORDER BY EmpID LIMIT 3;
+-------+----------+----------+------------+-----------------+-------------------------+------------+--------+
| EmpID | EmpFname | EmpLname | Department | Project         | Address                 | DOB        | Gender |
+-------+----------+----------+------------+-----------------+-------------------------+------------+--------+
|    1 | Alice    | Smith    | HR         | Recruitment     | 123 Main St, Springfield | 1990-05-15 | Female |
|    2 | Bob      | Johnson  | IT         | Web Development | 456 Oak St, Springfield  | 1985-03-22 | Male   |
|    3 | Charlie  | Brown    | Finance    | Accounting      | 789 Pine St, Springfield | 1992-07-30 | Male   |
+-------+----------+----------+------------+-----------------+-------------------------+------------+--------+
3 rows in set (0.00 sec)
```

7) Write a PL/SQL block of code using parameterized Cursor Merge the data available in the newly created table N_RollCall with the data available in the table O_RollCall. If the data in the first table already exist in the second table then that data should be skipped.2.

OUTPUT:

```
SQL> CREATE TABLE O_RollCall (Roll_no INT,Name VARCHAR(20));

Table created.


SQL> CREATE TABLE N_RollCall (Roll_no INT,Name VARCHAR(20));

Table created.


SQL> INSERT INTO O_RollCall VALUES (1, 'Vishal');

1 row created.


SQL> INSERT INTO O_RollCall VALUES (2, 'Vivek');

1 row created.


SQL> INSERT INTO O_RollCall VALUES (3, 'Sam');

1 row created.


SQL> INSERT INTO O_RollCall VALUES (4, 'Raj');

1 row created.


SQL> INSERT INTO O_RollCall VALUES (5, 'Avi');

1 row created.
```

```
SQL> INSERT INTO O_RollCall VALUES (6, 'Anu');
1 row created.


SQL> INSERT INTO O_RollCall VALUES (7, 'Vidhi');
1 row created.


SQL> INSERT INTO O_RollCall VALUES (8, 'Samu');
1 row created.


SQL> select * from O_RollCall;
  ROLL_NO NAME
---------- --------------------
        1 Vishal
        2 Vivek
        3 Sam
        4 Raj
        5 Avi
        6 Anu
        7 Vidhi
        8 Samu

8 rows selected.
SQL> INSERT INTO N_RollCall VALUES (1, 'Vishal Updated');
1 row created.


SQL> INSERT INTO N_RollCall VALUES (9, 'New Student');
1 row created.


SQL> select * from N_RollCall;
  ROLL_NO NAME
```

```
---------- --------------------
         1 Vishal Updated
         9 New Student


SQL> ed
Wrote file afiedt.buf
  1  DECLARE
  2     -- Parameterized cursor to fetch records from N_RollCall
  3     CURSOR roll_call_cursor IS
  4        SELECT Roll_no, Name FROM N_RollCall;
  5  BEGIN
  6     -- Open the cursor and loop through the records
  7     FOR rec IN roll_call_cursor LOOP
  8        -- Merge the data into O_RollCall
  9        MERGE INTO O_RollCall o
 10        USING (SELECT rec.Roll_no AS Roll_no, rec.Name AS Name FROM dual) n
 11        ON (o.Roll_no = n.Roll_no)
 12        WHEN MATCHED THEN
 13           UPDATE SET o.Name = n.Name  -- Update the Name if Roll_no matches
 14        WHEN NOT MATCHED THEN
 15           INSERT (Roll_no, Name) VALUES (n.Roll_no, n.Name);  -- Insert new record
 16     END LOOP;
 17     -- Commit the transaction
 18     COMMIT;
 19     DBMS_OUTPUT.PUT_LINE('Data merged successfully from N_RollCall to O_RollCall.');
 20  EXCEPTION
 21     WHEN OTHERS THEN
 22        ROLLBACK;  -- Rollback if any error occurs
 23        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
 24* END;
 25  /
Data merged successfully from N_RollCall to O_RollCall.
PL/SQL procedure successfully completed.


SQL> select * from O_RollCall;
   ROLL_NO NAME
---------- --------------------
         1 Vishal Updated
         2 Vivek
         3 Sam
         4 Raj
         5 Avi
         6 Anu
         7 Vidhi
         8 Samu
         9 New Student
9 rows selected.
```

8) Write a PL/SQL block of Stored Procedure and Stored Function proc_Grade for following problem statement. Write a Stored Procedure namely proc_Grade for the categorization of student. If marks scored by students in examination is <=1500 and marks>=990 then student will be placed in distinction category if marks scored are between 989 and 900 category is first class, if marks 899 and 825 category is Higher Second Class.

OUTPUT:

```
SQL> CREATE TABLE Stu_Marks (Roll_no NUMBER,Name VARCHAR(20), Total_Marks NUMBER);
Table created.

SQL> CREATE TABLE Results(Roll_no NUMBER,Name VARCHAR(20),Class VARCHAR(10));
Table created.
```

```
SQL> INSERT INTO Stu_Marks VALUES(101, 'Aviansh', 1000);

1 row created.


SQL> INSERT INTO Stu_Marks VALUES(102, 'Raj', 500);

1 row created.


SQL> INSERT INTO Stu_Marks VALUES(103, 'Ankit', 1500);

1 row created.


SQL> INSERT INTO Stu_Marks VALUES(104, 'Anu', 1200);

1 row created.


SQL> Select * from Stu_Marks;


   ROLL_NO NAME                 TOTAL_MARKS
---------- -------------------- -----------
       101 Aviansh                     1000
       102 Raj                          500
       103 Ankit                       1500
       104 Anu                         1200



SQL> ed
Wrote file afiedt.buf


  1  CREATE OR REPLACE PROCEDURE proc_Grade AS
  2  BEGIN
  3     FOR rec IN (SELECT Roll_no, Name, Total_Marks FROM Stu_Marks) LOOP
  4        DECLARE
  5           category VARCHAR2(10);  -- Adjusted size to fit existing limit
  6        BEGIN
  7           -- Determine the category based on Total_Marks
  8            IF rec.Total_Marks >= 990 AND rec.Total_Marks <= 1500 THEN
  9               category := 'Distnct';  -- Shortened
 10             ELSIF rec.Total_Marks BETWEEN 900 AND 989 THEN
 11                category := '1st Class';  -- Shortened
 12             ELSIF rec.Total_Marks BETWEEN 825 AND 899 THEN
 13                category := 'H. Sec';  -- Shortened
 14             ELSE
 15                category := 'Not Cat';  -- Shortened
 16             END IF;
 17             -- Insert the result into the Result table
 18             INSERT INTO Result (Roll_no, Name, Class)
 19             VALUES (rec.Roll_no, rec.Name, category);
 20          EXCEPTION
 21             WHEN OTHERS THEN
 22                DBMS_OUTPUT.PUT_LINE('Error inserting result for Roll_no: ' || rec.Roll_no || ' - ' ||
SQLERRM);
 23          END;
 24       END LOOP;
 25       COMMIT;
 26* END;
 27  /

Procedure created.

SQL> ed
Wrote file afiedt.buf

  1  DECLARE
  2  BEGIN
  3     -- Call the procedure to categorize students
  4     proc_Grade;
  5     -- Loop through the results in the Result table
  6     FOR rec IN (SELECT Roll_no, Name, Class FROM Result) LOOP
  7        DBMS_OUTPUT.PUT_LINE('Roll: ' || rec.Roll_no || ', Name: ' || rec.Name || ', Class: ' || rec.Class);
  8     END LOOP;
  9  EXCEPTION
 10     WHEN NO_DATA_FOUND THEN
 11        DBMS_OUTPUT.PUT_LINE('No records found in the Result table.');
 12     WHEN OTHERS THEN
 13        DBMS_OUTPUT.PUT_LINE('An unexpected error occurred: ' || SQLERRM);
 14* END;
 15  /
Roll: 103, Name: Ankit, Class: Distnct
Roll: 104, Name: Anu, Class: Distnct
Roll: 102, Name: Raj, Class: Not Cat
Roll: 101, Name: Aviansh, Class: Distnct

PL/SQL procedure successfully completed.
```

9) Write a database trigger: Row level and Statement level triggers, Before and After delete or update of database. Write a database trigger on Library table. The System should keep track of the records that are being updated or deleted. The old value of updated or deleted records should be added in Library_Audit table.

OUTPUT:

```
SQL> CREATE SEQUENCE Library_Audit_Sequence START WITH 1 INCREMENT BY 1 NOCACHE;

Sequence created.

SQL> CREATE TABLE Library (BookID INT PRIMARY KEY,Title VARCHAR(25),Author
VARCHAR(25),PublishedYear INT);

Table created.

SQL> CREATE TABLE Library_Audit (AuditID INT PRIMARY KEY,BookID INT,Title VARCHAR(25),Author
VARCHAR(25),PublishedYear INT,Action VARCHAR(10),ChangeDate DATE);

Table created.
```

```
SQL> INSERT INTO Library VALUES (1, 'Sample Book 1', 'Author 1', '2020');

1 row created.

SQL> INSERT INTO Library VALUES (2, 'Sample Book 2', 'Author 2', 2018);

1 row created.

SQL> select * from Library;

    BOOKID TITLE                    AUTHOR                   PUBLISHEDYEAR
---------- ------------------------ ------------------------ -------------
         1 Sample Book 1            Author 1                          2020
         2 Sample Book 2            Author 2                          2018

SQL> ed
Wrote file afiedt.buf

  1  CREATE OR REPLACE TRIGGER Library_Audit_Trigger
  2  AFTER DELETE OR UPDATE ON Library
  3  FOR EACH ROW
  4  BEGIN
  5      -- Insert audit record for DELETE operation
  6      IF DELETING THEN
  7          INSERT INTO Library_Audit (AuditID, BookID, Title, Author, PublishedYear, Action, ChangeDate)
  8          VALUES (Library_Audit_Sequence.NEXTVAL, :OLD.BookID, :OLD.Title, :OLD.Author,
:OLD.PublishedYear, 'DELETE', SYSDATE);
  9      END IF;
 10      -- Insert audit record for UPDATE operation
 11      IF UPDATING THEN
 12          INSERT INTO Library_Audit (AuditID, BookID, Title, Author, PublishedYear, Action, ChangeDate)
 13         VALUES (Library_Audit_Sequence.NEXTVAL, :OLD.BookID, :OLD.Title, :OLD.Author,
:OLD.PublishedYear, 'UPDATE', SYSDATE);
 14     END IF;
 15* END;
 16  /

Trigger created.

SQL> UPDATE Library SET Title = 'Updated Book 1' WHERE BookID = 1;

1 row updated.

SQL> DELETE FROM Library WHERE BookID = 2;

1 row deleted.

SQL>
SQL> SELECT * FROM Library_Audit;

   AUDITID     BOOKID TITLE                    AUTHOR
---------- ---------- ------------------------ ------------------------
PUBLISHEDYEAR ACTION     CHANGEDAT
------------- ---------- ---------
         1          1 Sample Book 1            Author 1
         2020 UPDATE     02-NOV-24

         2          2 Sample Book 2            Author 2
         2018 DELETE     02-NOV-24
```

10) Write a PL/SQL block of code for the following requirements:- Schema:

Borrower(Rollin, Name, DateofIssue, NameofBook, Status)

Fine(Roll_no, Date, Amt)

a. Accept roll_no & name of book from user.

b. Check the number of days (from date of issue), if days are between 15 to 30 then fine amount will be Rs 5per day.

c. If no. of days>30, per day fine will be Rs 50 per day & for days less than 30, Rs. 5 per day.

d. After submitting the book, status will change from I to R.

e. If condition of fine is true, then details will be stored into fine table.

OUTPUT:

```
SQL> create table B_1(roll_no int,name varchar(50),dateofissue date,nameofbook varchar(100),status
varchar(1));


Table created.


SQL> create table F_1(roll_no int,F_1date date,amt int);


Table created.


SQL> insert into B_1 values(101,'Raj',TO_DATE('2022-10-01','YYYY-MM-DD'),'dbms','I');


1 row created.


SQL> insert into B_1 values(101,'Prathamesh', TO_DATE('2022-10-15','YYYY-MM-DD'),'oop','I');

1 row created.


SQL> insert into B_1 values(103,'Harshada', TO_DATE('2022-09-24','YYYY-MM-DD'),'dsa','I');

1 row created.


SQL> insert into B_1 values(104,'Neha', TO_DATE('2022-08-26','YYYY-MM-DD'),'cns','I');

1 row created.


SQL> select*from B_1;


ed
SQL> select*from B_1;

+---------+------------+-------------+------------+--------+

| roll_no | name       | dateofissue | nameofbook | status |

+---------+------------+-------------+------------+--------+

|     101 | Raj        | 2022-10-01  | dbms       | I      |
|     102 | Prathamesh | 2022-10-15  | oop        | I      |
|     103 | Harshada   | 2022-09-24  | dsa        | I      |
|     104 | Neha       | 2022-08-26  | cns        | I      |
+---------+------------+-------------+------------+--------+
4 rows in set (0.01 sec)


SQL> ed
Wrote file afiedt.buf

DECLARE
   roll_no NUMBER;
   name_of_book VARCHAR2(100);
   no_of_days NUMBER;
   return_date DATE := SYSDATE;
   doi DATE;
   fine_amount NUMBER := 0;
BEGIN
   -- Accepting input from the user (simulated for this example)
   roll_no := &roll_no;               -- User is prompted to enter roll_no
   name_of_book := '&name_of_book';    -- User is prompted to enter name_of_book
   -- Fetch the date of issue based on roll_no and name_of_book
   SELECT dateofissue INTO doi
   FROM B_1
   WHERE roll_no = roll_no AND nameofbook = name_of_book AND status = 'I'; -- Only consider issued books
   -- Calculate the number of days late
   no_of_days := TRUNC(return_date) - TRUNC(doi);
   -- Determine the fine based on the number of days
   IF no_of_days > 30 THEN
      fine_amount := 150 + (no_of_days - 30) * 50; -- 150 for the first 30 days
   ELSIF no_of_days > 15 THEN
      fine_amount := (no_of_days * 5); -- 5 per day for days between 15 and 30
   END IF;
   -- Output the calculated fine
   dbms_output.put_line('Number of days: ' || no_of_days);
   dbms_output.put_line('Fine amount: ' || fine_amount);
   -- Insert the fine details into the Fine table
   INSERT INTO F_1 (roll_no, F_1date, amt) VALUES (roll_no, return_date, fine_amount);
   -- Update the status of the book to 'R' (returned)
   UPDATE B_1
   SET status = 'R'
   WHERE roll_no = roll_no AND nameofbook = name_of_book;
   dbms_output.put_line('Book status updated to Returned.');
EXCEPTION
   WHEN NO_DATA_FOUND THEN
      dbms_output.put_line('No record found for the given roll number and book name.');
```

```
 37      WHEN OTHERS THEN
 38         dbms_output.put_line('An error occurred: ' || SQLERRM);
 39* END;
 40  /
Enter value for roll_no: 101
old  10:     roll_no := &roll_no;               -- User is prompted to enter roll_no
new  10:     roll_no := 101;               -- User is prompted to enter roll_no
Enter value for name_of_book: dbms
old  11:     name_of_book := '&name_of_book';    -- User is prompted to enter name_of_book
new  11:     name_of_book := 'dbms';    -- User is prompted to enter name_of_book
Number of days: 763
Fine amount: 36800
Book status updated to Returned.

PL/SQL procedure successfully completed.

SQL> select * from F_1;

   ROLL_NO F_1DATE          AMT
---------- --------- ----------
       101 02-NOV-24      36800
```

11) The organization has decided to increase the salary of employees by 10% of existing salary, whose existing salary is less than Rs. 10000/- Write a PL/SQ block to update the salary as per above requirement, display an appropriate message based on the no. of rows affected by this update (using implicit cursor status variables).

OUTPUT:

```
DECLARE

   -- Variable to hold the number of rows affected

   v_rows_updated NUMBER;

BEGIN

   -- Update the salary for employees with salary less than 10,000

   UPDATE EmployeePosition

   SET Salary = Salary * 1.10

   WHERE Salary < 10000;



   -- Get the number of rows updated

   v_rows_updated := SQL%ROWCOUNT;



   -- Display message based on the number of rows updated
```

```
    IF v_rows_updated > 0 THEN
        DBMS_OUTPUT.PUT_LINE(v_rows_updated || ' employee(s) salary updated successfully.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('No employee salaries were updated.');
    END IF;

    -- Optionally, commit the changes if needed
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        -- Handle any exceptions that occur
        DBMS_OUTPUT.PUT_LINE('An error occurred: ' || SQLERRM);
        ROLLBACK; -- Rollback changes in case of error
END;
/
```

12) Create The following two tables :

College-info Faculty-info College-info consists of fields : college-code, college-name, address Faculty-info consists of fields : college-code, faculty-code, faculty-name, qualification, experience-in-no-of-years, address. The field college-code is foreign key. Generate queries to do the following :

a) Retrieve all records from the College_info table

b) Retrieve all records from the Faculty_info table

c) List the faculty members along with the college name they are associated with

d) List all those faculty members whose experience is greater than or equal to 10 years and have M. Tech degree.

e) List all those faculty members, who have at least 10 years of experience but do not have M. Tech degree

OUTPUT:

```
mysql> CREATE DATABASE COLLEGE;

Query OK, 1 row affected (0.01 sec)

mysql> USE COLLEGE;

Database changed
```

```
mysql> CREATE TABLE College_info ( college_code INT PRIMARY KEY, college_name VARCHAR(100) NOT NULL, address VARCHAR(255) );

Query OK, 0 rows affected (0.02 sec)


mysql> CREATE TABLE Faculty_info ( college_code INT, faculty_code INT PRIMARY KEY, faculty_name VARCHAR(100) NOT NULL, qualification VARCHAR(50), experience_in_no_of_years INT, address VARCHAR(255), FOREIGN KEY (college_code) REFERENCES College_info(college_code) );

Query OK, 0 rows affected (0.03 sec)


mysql> INSERT INTO College_info (college_code, college_name, address) VALUES(1, 'Springfield College', '123 College Ave, Springfield'),(2, 'Riverdale University', '456 River Rd, Riverdale'),(3, 'Hilltop Institute', '789 Hill St, Hilltown'),(4, 'Lakeside Academy', '321 Lakeview Dr, Lakeside'),(5, 'Mountainview College', '654 Mountain Rd, Mountainview');

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> INSERT INTO Faculty_info (college_code, faculty_code, faculty_name, qualification, experience_in_no_of_years, address) VALUES(1, 101, 'Alice Smith', 'B.Tech', 5, '101 Faculty St, Springfield'),(1, 102, 'Bob Johnson', 'M.Tech', 3, '102 Faculty St, Springfield'),(2, 201, 'Charlie Brown', 'MBA', 8, '201 Faculty St, Riverdale'),(3, 301, 'David Wilson', 'PhD in Physics', 10, '301 Faculty St, Hilltown'),(4, 401, 'Eva Green', 'M.Tech', 4, '401 Faculty St, Lakeside');

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0


a)mysql> SELECT * FROM Faculty_info;

+--------------+--------------+---------------+----------------+---------------------------+-----------------------------+

| college_code | faculty_code | faculty_name  | qualification  | experience_in_no_of_years | address
|

+--------------+--------------+---------------+----------------+---------------------------+-----------------------------+

|            1 |          101 | Alice Smith   | B.Tech         |                         5 | 101 Faculty St, Springfield |

|            1 |          102 | Bob Johnson   | M.Tech         |                         3 | 102 Faculty St, Springfield |

|            2 |          201 | Charlie Brown | MBA            |                         8 | 201 Faculty St, Riverdale   |

|            3 |          301 | David Wilson  | PhD in Physics |                        10 | 301 Faculty St, Hilltown    |

|            4 |          401 | Eva Green     | M.Tech         |                         4 | 401 Faculty St, Lakeside    |

+--------------+--------------+---------------+----------------+---------------------------+-----------------------------+

5 rows in set (0.00 sec)
```

b)mysql> SELECT * FROM College_info;

```
+--------------+---------------------+------------------------------+
| college_code | college_name        | address                      |
+--------------+---------------------+------------------------------+
|            1 | Springfield College  | 123 College Ave, Springfield  |
|            2 | Riverdale University | 456 River Rd, Riverdale       |
|            3 | Hilltop Institute    | 789 Hill St, Hilltown         |
|            4 | Lakeside Academy     | 321 Lakeview Dr, Lakeside     |
|            5 | Mountainview College | 654 Mountain Rd, Mountainview |
+--------------+---------------------+------------------------------+
5 rows in set (0.00 sec)
```

c)mysql> SELECT f.faculty_name, c.college_name FROM Faculty_info f JOIN College_info c ON f.college_code = c.college_code;

```
+---------------+----------------------+
| faculty_name  | college_name         |
+---------------+----------------------+
| Alice Smith   | Springfield College  |
| Bob Johnson   | Springfield College  |
| Charlie Brown | Riverdale University |
| David Wilson  | Hilltop Institute    |
| Eva Green     | Lakeside Academy     |
+---------------+----------------------+
5 rows in set (0.00 sec)
```

d)mysql> SELECT faculty_name FROM Faculty_info WHERE experience_in_no_of_years >= 10 AND qualification = 'M. Tech';

```
Empty set (0.00 sec)
```

e)mysql> SELECT faculty_name FROM Faculty_info WHERE experience_in_no_of_years >= 10 AND qualification <> 'M. Tech';

```
+--------------+
| faculty_name |
+--------------+
| David Wilson |
+--------------+
1 row in set (0.00 sec)
```

13) Create the following table : Student (roll-no, name, subject-name, subject-opted) Subject(faculty-code, faculty-name, specialization) Generate queries to do the following :

i) Retrieve all records from the Student table

ii)Retrieve all records from the Subject table

iii) Find the number of students who have enrolled for the subject "DBMS".

iv) Find all those faculty members who have not offered any subject.

v) Find all subjects enrolled by Roll-no 1101.

OUTPUT:

```
mysql> CREATE DATABASE STUDENTS;
Query OK, 1 row affected (0.00 sec)

mysql> USE STUDENTS;
Database changed
mysql> CREATE TABLE Student ( roll_no INT PRIMARY KEY, name VARCHAR(100) NOT NULL, subject_name VARCHAR(100), subject_opted VARCHAR(100) );
Query OK, 0 rows affected (0.02 sec)

mysql> CREATE TABLE Subject ( faculty_code INT PRIMARY KEY, faculty_name VARCHAR(100) NOT NULL, specialization VARCHAR(100) );
Query OK, 0 rows affected (0.02 sec)

mysql> INSERT INTO Student (roll_no, name, subject_name, subject_opted) VALUES(1101, 'Alice Smith', 'DBMS', 'Yes'),(1102, 'Bob Johnson', 'Algorithms', 'Yes'),(1103, 'Charlie Brown', 'DBMS', 'Yes'),(1104, 'David Wilson', 'Operating Systems', 'No'),(1105, 'Eva Green', 'Networking', 'Yes');
Query OK, 5 rows affected (0.01 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> INSERT INTO Subject (faculty_code, faculty_name, specialization) VALUES(101, 'Prof. Alan Turing', 'Computer Science'),(102, 'Dr. Ada Lovelace', 'Mathematics'),(103, 'Dr. Grace Hopper', 'Software Engineering'),(104, 'Mr. John von Neumann', 'Theoretical Computer Science'),(105, 'Ms. Barbara Liskov', 'Distributed Systems');

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0


i)mysql> SELECT *FROM Student;

+---------+---------------+-------------------+---------------+

| roll_no | name          | subject_name      | subject_opted |

+---------+---------------+-------------------+---------------+

|    1101 | Alice Smith   | DBMS              | Yes           |

|    1102 | Bob Johnson   | Algorithms        | Yes           |

|    1103 | Charlie Brown | DBMS              | Yes           |

|    1104 | David Wilson  | Operating Systems | No            |

|    1105 | Eva Green     | Networking        | Yes           |

+---------+---------------+-------------------+---------------+

5 rows in set (0.00 sec)


ii)mysql> SELECT *FROM Subject;

+--------------+----------------------+------------------------------+

| faculty_code | faculty_name         | specialization               |

+--------------+----------------------+------------------------------+

|          101 | Prof. Alan Turing    | Computer Science             |

|          102 | Dr. Ada Lovelace     | Mathematics                  |

|          103 | Dr. Grace Hopper     | Software Engineering         |

|          104 | Mr. John von Neumann | Theoretical Computer Science |

|          105 | Ms. Barbara Liskov   | Distributed Systems          |

+--------------+----------------------+------------------------------+

5 rows in set (0.00 sec)


iii)mysql> SELECT COUNT(*) AS NumberOfStudents FROM Student WHERE subject_name = 'DBMS';

+------------------+
| NumberOfStudents |
+------------------+
|                2 |
+------------------+
1 row in set (0.00 sec)
```

```
iv)mysql> SELECT faculty_name FROM Subject WHERE faculty_code NOT IN (SELECT DISTINCT faculty_code FROM Student);
Empty set (0.01 sec)
```

```
v)mysql> SELECT subject_name FROM Student WHERE roll_no = 1101;
+--------------+
| subject_name |
+--------------+
| DBMS         |
+--------------+
1 row in set (0.00 sec)
```

14) Create the following table :

Item (item-code, item-name, qty-in-stock, reorder-level)

Supplier (supplier-code, supplier-name, address)

Can-supply(supplier-code, item-code) Generate queries to do the following :

- **(i)** **Retrieve all records from the Item table**
- **(ii)** **Retrieve all records from the Supplier table**
- **(iii)** **Display all Items supplied by all suppliers.**
- **(iv)** **Retrieve items where the quantity in stock is below the reorder level**
- **(v)** **List all those suppliers who can supply the given item.**
- **(vi)** **List all those items which cannot be supplied by given company**

```
mysql> CREATE DATABASE HOTEL;
Query OK, 1 row affected (0.01 sec)

mysql> USE HOTEL;
```

```
Database changed

mysql> CREATE TABLE Item ( item_code INT PRIMARY KEY, item_name VARCHAR(100) NOT NULL,
qty_in_stock INT, reorder_level INT );

Query OK, 0 rows affected (0.02 sec)


mysql> CREATE TABLE Supplier ( supplier_code INT PRIMARY KEY, supplier_name VARCHAR(100) NOT NULL,
address VARCHAR(255) );

Query OK, 0 rows affected (0.02 sec)


mysql> CREATE TABLE Can_supply ( supplier_code INT, item_code INT, PRIMARY KEY (supplier_code,
item_code), FOREIGN KEY (supplier_code) REFERENCES Supplier(supplier_code), FOREIGN KEY (item_code)
REFERENCES Item(item_code) );

Query OK, 0 rows affected (0.04 sec)


mysql> INSERT INTO Item (item_code, item_name, qty_in_stock, reorder_level) VALUES(1, 'Widget A', 50,
20),(2, 'Widget B', 30, 15),(3, 'Gadget C', 10, 25),(4, 'Gadget D', 5, 10),(5, 'Thingamajig E', 100, 50);

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> INSERT INTO Supplier (supplier_code, supplier_name, address) VALUES(101, 'Supplier One', '123
Main St, Springfield'),(102, 'Supplier Two', '456 Elm St, Springfield'),(103, 'Supplier Three', '789 Oak St,
Springfield'),(104, 'Supplier Four', '321 Pine St, Springfield'),(105, 'Supplier Five', '654 Cedar St, Springfield');

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0


mysql> INSERT INTO Can_supply (supplier_code, item_code) VALUES(101, 1),(101, 2),(102, 1),(103, 3),(104,
4),(105, 5),(105, 2);

Query OK, 7 rows affected (0.00 sec)

Records: 7  Duplicates: 0  Warnings: 0


i)mysql> SELECT *FROM Item;

+-----------+---------------+--------------+---------------+

| item_code | item_name     | qty_in_stock | reorder_level |

+-----------+---------------+--------------+---------------+

|         1 | Widget A      |           50 |            20 |
|        2 | Widget B      |           30 |            15 |
|        3 | Gadget C      |           10 |            25 |
|        4 | Gadget D      |            5 |            10 |
|        5 | Thingamajig E |          100 |            50 |
+-----------+---------------+--------------+---------------+
5 rows in set (0.00 sec)
```

ii)mysql> SELECT *FROM Supplier;

```
+---------------+----------------+---------------------------+
| supplier_code | supplier_name  | address                   |
+---------------+----------------+---------------------------+
|           101 | Supplier One   | 123 Main St, Springfield  |
|           102 | Supplier Two   | 456 Elm St, Springfield   |
|           103 | Supplier Three | 789 Oak St, Springfield   |
|           104 | Supplier Four  | 321 Pine St, Springfield  |
|           105 | Supplier Five  | 654 Cedar St, Springfield |
+---------------+----------------+---------------------------+
5 rows in set (0.00 sec)
```

iii)mysql> SELECT s.supplier_name, i.item_name FROM Supplier s JOIN Can_supply cs ON s.supplier_code = cs.supplier_code JOIN Item i ON cs.item_code = i.item_code;

```
+----------------+---------------+
| supplier_name  | item_name     |
+----------------+---------------+
| Supplier One   | Widget A      |
| Supplier One   | Widget B      |
| Supplier Two   | Widget A      |
| Supplier Three | Gadget C      |
| Supplier Four  | Gadget D      |
| Supplier Five  | Widget B      |
| Supplier Five  | Thingamajig E |
+----------------+---------------+
7 rows in set (0.00 sec)
```

```
iv)mysql> SELECT *FROM Item WHERE qty_in_stock < reorder_level;

+-----------+-----------+--------------+---------------+

| item_code | item_name | qty_in_stock | reorder_level |

+-----------+-----------+--------------+---------------+

|         3 | Gadget C  |           10 |            25 |

|         4 | Gadget D  |            5 |            10 |

+-----------+-----------+--------------+---------------+

2 rows in set (0.00 sec)
```

```
v)mysql> SELECT s.supplier_name FROM Supplier s JOIN Can_supply cs ON s.supplier_code =
cs.supplier_code WHERE cs.item_code = 2;

+---------------+

| supplier_name |

+---------------+

| Supplier One  |

| Supplier Five |

+---------------+

2 rows in set (0.00 sec)
```

```
vi)mysql> SELECT i.item_name FROM Item i WHERE i.item_code NOT IN ( SELECT cs.item_code FROM
Can_supply cs WHERE cs.supplier_code = 101 );

+---------------+

| item_name     |

+---------------+

| Gadget C      |

| Gadget D      |

| Thingamajig E |

+---------------+

3 rows in set (0.00 sec)
```

15) Create the following tables:

Student (roll-no, marks, category, district, state)

Student-rank(roll-no, marks, rank) Generate queries to do the following :

i) Retrieve all records from the Student table

ii) Retrieve all records from the Student-rank table

iii)display each student's details along with their rank

vi) List all those students who have come from Tamilnadu state and secured a rank above 100.

v) List all those students who come from Andhra Pradesh state and belong to given category who have secured a rank above 100

OUTPUT:

```
mysql> CREATE DATABASE STU;

Query OK, 1 row affected (0.01 sec)


mysql> USE STU;

Database changed

mysql> CREATE TABLE Student ( roll_no INT PRIMARY KEY, marks INT, category VARCHAR(50), district
VARCHAR(100), state VARCHAR(100) );

Query OK, 0 rows affected (0.02 sec)


mysql> CREATE TABLE Student_rank ( roll_no INT PRIMARY KEY, marks INT, student_rank INT, FOREIGN KEY
(roll_no) REFERENCES Student(roll_no) );

Query OK, 0 rows affected (0.02 sec)


mysql> INSERT INTO Student (roll_no, marks, category, district, state) VALUES(1, 85, 'General', 'Chennai',
'Tamil Nadu'),(2, 78, 'OBC', 'Hyderabad', 'Telangana'),(3, 90, 'SC', 'Visakhapatnam', 'Andhra Pradesh'),(4, 65,
'General', 'Coimbatore', 'Tamil Nadu'),(5, 72, 'OBC', 'Amaravati', 'Andhra Pradesh');

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0


i) mysql> SELECT *FROM Student;

+---------+-------+----------+---------------+----------------+

| roll_no | marks | category | district      | state          |

+---------+-------+----------+---------------+----------------+

|       1 |    85 | General  | Chennai       | Tamil Nadu     |

|       2 |    78 | OBC      | Hyderabad     | Telangana      |

|       3 |    90 | SC       | Visakhapatnam | Andhra Pradesh |
|       4 |    65 | General  | Coimbatore     | Tamil Nadu     |
|       5 |    72 | OBC      | Amaravati      | Andhra Pradesh |
+---------+-------+----------+---------------+----------------+
5 rows in set (0.00 sec)
```

ii)mysql> SELECT *FROM Student_rank;

```
+---------+-------+--------------+
| roll_no | marks | student_rank |
+---------+-------+--------------+
|       1 |    85 |           50 |
|       2 |    78 |          150 |
|       3 |    90 |           30 |
|       4 |    65 |          200 |
|       5 |    72 |          120 |
+---------+-------+--------------+
5 rows in set (0.00 sec)
```

iii) mysql> SELECT s.roll_no, s.marks, s.category, s.district, s.state, r.student_rank FROM Student s JOIN Student_rank r ON s.roll_no = r.roll_no;

```
+---------+-------+----------+---------------+----------------+--------------+
| roll_no | marks | category | district      | state          | student_rank |
+---------+-------+----------+---------------+----------------+--------------+
|       1 |    85 | General  | Chennai       | Tamil Nadu     |           50 |
|       2 |    78 | OBC      | Hyderabad     | Telangana      |          150 |
|       3 |    90 | SC       | Visakhapatnam | Andhra Pradesh |           30 |
|       4 |    65 | General  | Coimbatore    | Tamil Nadu     |          200 |
|       5 |    72 | OBC      | Amaravati     | Andhra Pradesh |          120 |
+---------+-------+----------+---------------+----------------+--------------+
5 rows in set (0.00 sec)
```

iv) mysql> SELECT s.roll_no, s.marks, s.category, s.district, s.state, r.student_rank FROM Student s JOIN Student_rank r ON s.roll_no = r.roll_no WHERE s.state = 'Tamil Nadu' AND r.student_rank > 100;

```
+---------+-------+----------+------------+------------+--------------+
| roll_no | marks | category | district   | state      | student_rank |
+---------+-------+----------+------------+------------+--------------+
|       4 |    65 | General  | Coimbatore | Tamil Nadu |          200 |
+---------+-------+----------+------------+------------+--------------+
1 row in set (0.00 sec)
```

v)mysql> SELECT s.roll_no, s.marks, s.category, s.district, s.state, r.student_rank FROM Student s JOIN Student_rank r ON s.roll_no = r.roll_no WHERE s.state = 'Andhra Pradesh' AND s.category = 'OBC' AND r.student_rank > 100;

```
+---------+-------+----------+-----------+----------------+--------------+
| roll_no | marks | category | district  | state          | student_rank |
+---------+-------+----------+-----------+----------------+--------------+
|       5 |    72 | OBC      | Amaravati | Andhra Pradesh |          120 |
+---------+-------+----------+-----------+----------------+--------------+
1 row in set (0.00 sec)
```

MONGODB

16) Create a collection named Book. (book_isbn,title,punlisher_name,author(Name, Address, Phone No[landline, mobile]), publisher_city, price,copies) i.

a. Add 5 documents in the collection with keys

b. Give details of Books whose Publisher lives in “Pune”.

c. Delete name Book from Book whose name start with “D”

d. Change the city of publisher “A.Nagar” to “New York”.

e. Find the details of publisher named “Data Analytics”.

OUTPUT:

```
a) test> show dbs
Book     8.00 KiB
admin   40.00 KiB
config  60.00 KiB
local   40.00 KiB
test> use Book
switched to db Book
Book>
db.Book.insert({Book_isbn:1001,Title:"DBMS",P_name:"Technical",Author:{Name:"ABC",Address:"Pune",Phone_no:[{Landline:1234567989,Mobile:365897413}]},P_city:"Baramati",Price:250,Copies:150})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb477634a595b13c73bf8') }
}
Book> db.Book.find().pretty()
[
  {
    _id: ObjectId('66ffb477634a595b13c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  }
]
Book> db.Book.find()
[
  {
    _id: ObjectId('66ffb477634a595b13c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  }
]
Book>
db.Book.insert({Book_isbn:1002,Title:"JAVA",P_name:"Technical",Author:{Name:"PQR",Address:"Ba
ramati",Phone_no:[{Landline:24367989,Mobile:986897413}]},P_city:"Nashik",Price:350,Copies:50})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb57c634a595b13c73bf9') }
}
Book>
db.Book.insert({Book_isbn:1002,Title:"MYSQL",P_name:"OracleCorporation",Author:{Name:"XYZ",A
ddress:"Pune",Phone_no:[{Landline:2154359,Mobile:986541233}]},P_city:"Pune",Price:150,Copies:
70})
```

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb631634a595b13c73bfa') }
}
Book> db.Book.insert({Book_isbn:1004,Title:"Data Analytics Made Accesible",P_name:"Data
Analytics",Author:{Name:"Anil
Maheshwari",Address:"Mumbai",Phone_no:[{Landline:21354359,Mobile:78941233}]},P_city:"Pune
",Price:250,Copies:70})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb729634a595b13c73bfb') }
}
Book> db.Book.insert({Book_isbn:1005,Title:"Python for Data Analytics",P_name:"Data Analytics
PY",Author:{Name:"William
McKinney",Address:"Pune",Phone_no:[{Landline:291354359,Mobile:978941233}]},P_city:"A.Nagar"
,Price:540,Copies:55})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb7d5634a595b13c73bfc') }
}
Book> db.Book.find().pretty()
[
  {
    _id: ObjectId('66ffb477634a595b13c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  },
  {
    _id: ObjectId('66ffb57c634a595b13c73bf9'),
    Book_isbn: 1002,
    Title: 'JAVA',
    P_name: 'Technical',
    Author: {
      Name: 'PQR',
      Address: 'Baramati',
      Phone_no: [ { Landline: 24367989, Mobile: 986897413 } ]
    },
    P_city: 'Nashik',
    Price: 350,
    Copies: 50
  },
  {
    _id: ObjectId('66ffb631634a595b13c73bfa'),
    Book_isbn: 1002,
    Title: 'MYSQL',
    P_name: 'OracleCorporation',
    Author: {
      Name: 'XYZ',
      Address: 'Pune',
      Phone_no: [ { Landline: 2154359, Mobile: 986541233 } ]
    },
    P_city: 'Pune',
    Price: 150,
    Copies: 70
  },
  {
    _id: ObjectId('66ffb729634a595b13c73bfb'),
    Book_isbn: 1004,
    Title: 'Data Analytics Made Accesible',
    P_name: 'Data Analytics',
    Author: {
      Name: 'Anil Maheshwari',
      Address: 'Mumbai',
      Phone_no: [ { Landline: 21354359, Mobile: 78941233 } ]
    },
    P_city: 'Pune',
    Price: 250,
    Copies: 70
  },
  {
    _id: ObjectId('66ffb7d5634a595b13c73bfc'),
    Book_isbn: 1005,
    Title: 'Python for Data Analytics',
    P_name: 'Data Analytics PY',
    Author: {
      Name: 'William McKinney',
      Address: 'Pune',
      Phone_no: [ { Landline: 291354359, Mobile: 978941233 } ]
    },
    P_city: 'A.Nagar',
    Price: 540,
    Copies: 55
  }
]
b) Book> db.Book.find({P_city:"Pune"})
[
  {
    _id: ObjectId('66ffb631634a595b13c73bfa'),
    Book_isbn: 1002,
    Title: 'MYSQL',
    P_name: 'OracleCorporation',
    Author: {
      Name: 'XYZ',
      Address: 'Pune',
      Phone_no: [ { Landline: 2154359, Mobile: 986541233 } ]
    },
    P_city: 'Pune',
    Price: 150,
    Copies: 70
  },
  {
    _id: ObjectId('66ffb729634a595b13c73bfb'),
    Book_isbn: 1004,
    Title: 'Data Analytics Made Accesible',
    P_name: 'Data Analytics',
    Author: {
      Name: 'Anil Maheshwari',
      Address: 'Mumbai',
      Phone_no: [ { Landline: 21354359, Mobile: 78941233 } ]
    },
    P_city: 'Pune',
    Price: 250,
    Copies: 70
  }
]
```

a. c) **Delete name Book from Book whose name start with “D”**

```
Book> db.Book.find({Title:/^D/})
[
  {
    _id: ObjectId('66ffb477634a595b13c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  },
  {
    _id: ObjectId('66ffb729634a595b13c73bfb'),
    Book_isbn: 1004,
    Title: 'Data Analytics Made Accesible',
    P_name: 'Data Analytics',
    Author: {
      Name: 'Anil Maheshwari',
      Address: 'Mumbai',
      Phone_no: [ { Landline: 21354359, Mobile: 78941233 } ]
    },
    P_city: 'Pune',
    Price: 250,
    Copies: 70
  }
]
Book> db.Book.remove({Title:/^D/})
DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany,
findOneAndDelete, or bulkWrite.
{ acknowledged: true, deletedCount: 2 }
d) Book> db.Book.find({P_city: 'A.Nagar'})
[
  {
    _id: ObjectId('6713f1749e77ca7f76c73bfc'),
    Book_isbn: 1005,
    Title: 'Python for Data Analytics',
    P_name: 'Data Analytics PY',
    Author: {
      Name: 'William McKinney',
      Address: 'Pune',
      Phone_no: [ { Landline: 291354359, Mobile: 978941233 } ]
    },
    P_city: 'A.Nagar',
    Price: 540,
    Copies: 55
  }
]
Book> db.Book.update({P_city:'A.Nagar'},{$set:{P_City:'New York'}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
e) Book> db.Book.find({P_name:'Data Analytics'})
[
  {
    _id: ObjectId('6713f1669e77ca7f76c73bfb'),
    Book_isbn: 1004,
    Title: 'Data Analytics Made Accesible',
    P_name: 'Data Analytics',
    Author: {
      Name: 'Anil Maheshwari',
      Address: 'Mumbai',
      Phone_no: [ { Landline: 21354359, Mobile: 78941233 } ]
    },
    P_city: 'Pune',
    Price: 250,
    Copies: 70
  }
]
```

17) Create a collection named Book. (book_isbn,title,punlisher_name,author(Name, Address, Phone No[landline, mobile]), publisher_city, price,copies)

a. Count the number of documents in the collection.

b. Arrange the documents in descending order of book_isbn.

c. Select Book Names whose title is ”DBMS” .

d. Update Book Copies as “120” whose Book Publisher is “OracleCorporation”.

e.Display name of publishers as per no of books published by them in ascending order.

OUTPUT:

a.
```
Book> db.Book.countDocuments()

5
```
b. **Arrange the documents in descending order of book_isbn**
```
Book> db.Book.find({}).sort({ book_isbn: -1 })
[
  {
    _id: ObjectId('6713f1149e77ca7f76c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  },
  {
    _id: ObjectId('6713f13e9e77ca7f76c73bf9'),
    Book_isbn: 1002,
  Title: 'JAVA',
  P_name: 'Technical',
  Author: {
    Name: 'PQR',
    Address: 'Baramati',
    Phone_no: [ { Landline: 24367989, Mobile: 986897413 } ]
  },
  P_city: 'Nashik',
  Price: 350,
  Copies: 50
},
{
  _id: ObjectId('6713f15e9e77ca7f76c73bfa'),
  Book_isbn: 1002,
  Title: 'MYSQL',
  P_name: 'OracleCorporation',
  Author: {
    Name: 'XYZ',
    Address: 'Pune',
    Phone_no: [ { Landline: 2154359, Mobile: 986541233 } ]
  },
  P_city: 'Pune',
  Price: 150,
  Copies: 70
},
{
  _id: ObjectId('6713f1669e77ca7f76c73bfb'),
  Book_isbn: 1004,
  Title: 'Data Analytics Made Accesible',
  P_name: 'Data Analytics',
  Author: {
    Name: 'Anil Maheshwari',
    Address: 'Mumbai',
    Phone_no: [ { Landline: 21354359, Mobile: 78941233 } ]
  },
  P_city: 'Pune',
  Price: 250,
  Copies: 70
},
{
  _id: ObjectId('6713f1749e77ca7f76c73bfc'),
  Book_isbn: 1005,
  Title: 'Python for Data Analytics',
  P_name: 'Data Analytics PY',
  Author: {
    Name: 'William McKinney',
    Address: 'Pune',
    Phone_no: [ { Landline: 291354359, Mobile: 978941233 } ]
    },
    P_city: 'A.Nagar',
    Price: 540,
    Copies: 55,
    P_City: 'New York'
  }
]
```

c. **Select Book Names whose title is "Python for Data Analytics"**

```
Book> db.Book.find({Title:"Python for Data Analytics"})
[
  {
    _id: ObjectId('6713f1749e77ca7f76c73bfc'),
    Book_isbn: 1005,
    Title: 'Python for Data Analytics',
    P_name: 'Data Analytics PY',
    Author: {
      Name: 'William McKinney',
      Address: 'Pune',
      Phone_no: [ { Landline: 291354359, Mobile: 978941233 } ]
    },
    P_city: 'A.Nagar',
    Price: 540,
    Copies: 55,
    P_City: 'New York'
  }
]
```

d. **Update Book Copies as "120" whose Book Publisher is "OracleCorporation"**

```
Book> db.Book.update({P_name:"OracleCorporation"},{$set:{Copies:120}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

e. **Display name of publishers as per no of books published by them in ascending order.**

```
Book> db.Book.aggregate([{$group:{_id:"$P_name", totalBooks:{$sum:1}}},{$sort:{totalBooks:1}}])
[
  { _id: 'Data Analytics PY', totalBooks: 1 },
  { _id: 'Data Analytics', totalBooks: 1 },
  { _id: 'OracleCorporation', totalBooks: 1 },
  { _id: 'Technical', totalBooks: 2 }
]
```

18) Create a collection named “ORDERS” that contain documents of the following prototype and solve the following queries: { cust_id: "abc123", ord_date: new Date("Oct 04, 2012"), status: 'A', price: 50, items: [{ sku: "xxx", qty: 25, price: 1 }, { sku: "yyy", qty: 25, price: 1 }]}

OUTPUT:

```
Book> use Orders

switched to db Orders

Orders> db.Orders.find()

[

  {

    _id: ObjectId('67153519506ef8b41cc73bf8'),

    cust_id: 'ABC123',

    ord_date: ISODate('2024-10-03T18:30:00.000Z'),

    Status: 'A',

    Price: 150,

    items: [

      { sku: 'XXX', qty: 20, Price: 100 },

      { sku: 'YYY', qty: 25, Price: 120 }

    ]

  },

  {

    _id: ObjectId('67153568506ef8b41cc73bf9'),

    cust_id: 'PQR456',

    ord_date: ISODate('2023-11-03T18:30:00.000Z'),

    Status: 'O',

    Price: 220,

    items: [

      { sku: 'AAA', qty: 10, Price: 250 },

      { sku: 'ZZZ', qty: 15, Price: 220 }

    ]

  },

  {

    _id: ObjectId('671535a8506ef8b41cc73bfa'),

    cust_id: 'PQR456',
   ord_date: ISODate('2024-09-09T18:30:00.000Z'),
   Status: 'A',
   Price: 450,
   items: [
     { sku: 'QQQ', qty: 30, Price: 150 },
     { sku: 'PPP', qty: 5, Price: 320 }
   ]
  },
  {
   _id: ObjectId('671535e6506ef8b41cc73bfb'),
   cust_id: 'XYZ789',
   ord_date: ISODate('2024-12-09T18:30:00.000Z'),
   Status: 'O',
   Price: 200,
   items: [
     { sku: 'TTT', qty: 30, Price: 150 },
     { sku: 'SSS', qty: 5, Price: 320 }
   ]
  },
  {
   _id: ObjectId('67153641506ef8b41cc73bfc'),
   cust_id: 'LMN123',
   ord_date: ISODate('2024-01-19T18:30:00.000Z'),
   Status: 'A',
   Price: 200,
   items: [
     { sku: 'NNN', qty: 30, Price: 150 },
     { sku: 'MMM', qty: 5, Price: 320 }
   ]
  }
]
```

a. **Count all records from orders**

```
Orders> db.Orders.countDocuments()
```

```
5
```

b. **Sum of the price field from orders**

```
Orders> db.Orders.aggregate([{ $group:{ _id: null, totalPrice: {$sum:"$Price"}}}])
[ { _id: null, totalPrice: 1220 } ]
```

c. **For each unique cust_id, sum the price field.**

```
Orders> db.Orders.aggregate([{$group:{ _id:"$cust_id", totalPrice: { $sum: "$Price"}}}])
[
  { _id: 'XYZ789', totalPrice: 200 },
  { _id: 'LMN123', totalPrice: 200 },
  { _id: 'ABC123', totalPrice: 150 },
  { _id: 'PQR456', totalPrice: 670 }
]
```

d. **For each unique cust_id, sum the price field, results sorted by sum**

```
Orders> db.Orders.aggregate([{$group:{ _id:"$cust_id", totalPrice: { $sum: "$Price"}}},
{$sort:{TotalPrice: -1}}])
[
  { _id: 'XYZ789', totalPrice: 200 },
  { _id: 'LMN123', totalPrice: 200 },
  { _id: 'PQR456', totalPrice: 670 },
  { _id: 'ABC123', totalPrice: 150 }
]
```

19) Create a collection named rating that contain 5 documents of the following prototype and solve the following Queries. { movie_id: 123, user_id: 12, title: Toy Story(1995), status: 'A' }

OUTPUT:

```
Orders> use Rating

switched to db Rating

Rating> db.Rating.insert({ Movie_id:123, User_id:12, Title:"Avatar(2009)", Status:'A'})

{

  acknowledged: true,

  insertedIds: { '0': ObjectId('671542ac506ef8b41cc73bfd') }

}

Rating> db.Rating.insert({ Movie_id:423, User_id:13, Title:"Titanic(1997)", Status:'UA'})

{

  acknowledged: true,

  insertedIds: { '0': ObjectId('671542df506ef8b41cc73bfe') }

}

Rating> db.Rating.insert({ Movie_id:234, User_id:14, Title:"The Dark Knight(2007)", Status:'A'})

{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6715430e506ef8b41cc73bff') }
}
Rating> db.Rating.insert({ Movie_id:1234, User_id:15, Title:"The Gun(1986)", Status:'A'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67154337506ef8b41cc73c00') }
}
Rating> db.Rating.insert({ Movie_id:254, User_id:16, Title:"Star Wars(1983)", Status:'UA'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6715436b506ef8b41cc73c01') }
}
Rating> db.Rating.find().pretty()
[{
    _id: ObjectId('671542ac506ef8b41cc73bfd'),
    Movie_id: 123,
    User_id: 12,
    Title: 'Avatar(2009)',
    Status: 'A'
  },
  {
    _id: ObjectId('671542df506ef8b41cc73bfe'),
    Movie_id: 423,
    User_id: 13,
    Title: 'Titanic(1997)',
    Status: 'UA'
  },{
    _id: ObjectId('6715430e506ef8b41cc73bff'),
    Movie_id: 234,
    User_id: 14,
    Title: 'The Dark Knight(2007)',
    Status: 'A'
  },
  {
    _id: ObjectId('67154337506ef8b41cc73c00'),
    Movie_id: 1234,
    User_id: 15,
    Title: 'The Gun(1986)',
    Status: 'A'
  },
  {
    _id: ObjectId('6715436b506ef8b41cc73c01'),
    Movie_id: 254,
    User_id: 16,
    Title: 'Star Wars(1983)',
    Status: 'UA'
  }]
```

a. **Creating an index on Movie_id and sorts the keys in the index in ascending order. Verify the query plan**

```
Rating> db.Rating.createIndex({ Movie_id: 1 })
Movie_id_1
```

b. **Showvarious indexes created on movie collection.**

```
Rating> db.Rating.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { Movie_id: 1 }, name: 'Movie_id_1' }
]
```

c. **Sort movie_id in descending order.**

```
Rating> db.Rating.find({}).sort({ Movie_id: -1 })
[
  {
    _id: ObjectId('67154337506ef8b41cc73c00'),
    Movie_id: 1234,
    User_id: 15,
    Title: 'The Gun(1986)',
    Status: 'A'
  },
  {
    _id: ObjectId('671542df506ef8b41cc73bfe'),
    Movie_id: 423,
    User_id: 13,
    Title: 'Titanic(1997)',
    Status: 'UA'
  },
  {
    _id: ObjectId('6715436b506ef8b41cc73c01'),
    Movie_id: 254,
    User_id: 16,
    Title: 'Star Wars(1983)',
    Status: 'UA'
  },
  {
    _id: ObjectId('6715430e506ef8b41cc73bff'),
    Movie_id: 234,
    User_id: 14,
    Title: 'The Dark Knight(2007)',
    Status: 'A'
  },
  {
    _id: ObjectId('671542ac506ef8b41cc73bfd'),
    Movie_id: 123,
    User_id: 12,
    Title: 'Avatar(2009)',
    Status: 'A'
  }
]
```

d. **Create a descending order index on Movie_id to get ratings related to “Avatar(2009)” verify the query plan.**

```
Rating> db.Rating.createIndex({ Movie_id: -1 })

Movie_id_-1

Rating> db.Rating.find({ Title: "Avatar(2009)" })

[

  {

    _id: ObjectId('671542ac506ef8b41cc73bfd'),

    Movie_id: 123,

    User_id: 12,

    Title: 'Avatar(2009)',

    Status: 'A'

  }

]
```

e. **Limit the number of items in the result of above query.**

```
Rating> db.Rating.find({}).limit(5);

[

  {

    _id: ObjectId('671542ac506ef8b41cc73bfd'),

    Movie_id: 123,

    User_id: 12,

    Title: 'Avatar(2009)',

    Status: 'A'

  },

  {

    _id: ObjectId('671542df506ef8b41cc73bfe'),

    Movie_id: 423,

    User_id: 13,

    Title: 'Titanic(1997)',

    Status: 'UA'

  },

  {

    _id: ObjectId('6715430e506ef8b41cc73bff'),
    Movie_id: 234,
    User_id: 14,
    Title: 'The Dark Knight(2007)',
    Status: 'A'
  },
  {
    _id: ObjectId('67154337506ef8b41cc73c00'),
    Movie_id: 1234,
    User_id: 15,
    Title: 'The Gun(1986)',
    Status: 'A'
  },
  {
    _id: ObjectId('6715436b506ef8b41cc73c01'),
    Movie_id: 254,
    User_id: 16,
    Title: 'Star Wars(1983)',
    Status: 'UA'
  }
]
```

20) Design a map-reduce operations on a collection "orders" that contains documents of the following prototype. Solve the following . { cust_id: "abc123", ord_date: new Date("Oct 04, 2012"), status: 'A', price: 25, gender :'F', rating: 1 }

a) Count the number of female (F) and male (M) respondents in the orders collection

b) Count the number of each type of rating (1, 2, 3, 4 or 5) for each orders

OUTPUT:

```
Orders> db.orders.insertMany([
...    {
...        cust_id: "abc123",
...        ord_date: new Date("Oct 04, 2012"),
...        status: 'A',
...        price: 25,
...        gender: 'F',
...       rating: 1
...    },
...    {
...       cust_id: "def456",
...       ord_date: new Date("Nov 10, 2012"),
...       status: 'A',
...       price: 30,
...       gender: 'M',
...       rating: 3
...    },
...    {
...       cust_id: "ghi789",
...       ord_date: new Date("Dec 15, 2012"),
...       status: 'A',
...       price: 45,
...       gender: 'F',
...       rating: 5
...    },
...    {
...       cust_id: "jkl012",
...       ord_date: new Date("Jan 22, 2013"),
...       status: 'A',
...       price: 20,
...       gender: 'M',
...       rating: 2
...    },
...    {
...       cust_id: "mno345",
...       ord_date: new Date("Feb 28, 2013"),
...       status: 'A',
...       price: 50,
...       gender: 'F',
...         rating: 4
...     },
...     {
...         cust_id: "pqr678",
...         ord_date: new Date("Mar 12, 2013"),
...         status: 'A',
...         price: 15,
...         gender: 'M',
...         rating: 1
...     }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67274614620f1f5f8cc73bf8'),
    '1': ObjectId('67274614620f1f5f8cc73bf9'),
    '2': ObjectId('67274614620f1f5f8cc73bfa'),
    '3': ObjectId('67274614620f1f5f8cc73bfb'),
    '4': ObjectId('67274614620f1f5f8cc73bfc'),
    '5': ObjectId('67274614620f1f5f8cc73bfd')
  }
}
```

a) Orders> db.Orders.mapReduce(function () {emit (this.gender, 1);}, function (key, values) {return Array.sum(values);},{out:'gender_count'});
{ result: 'gender_count', ok: 1 }

```
Orders> db.orders.aggregate([ { $group: { _id: "$gender", } } ]);
[ { _id: 'F' }, { _id: 'M' }
```

b) Orders> db.Orders.mapReduce(function () {emit (this.rating, 1);}, function (key, values) {return Array.sum(values);},{out:'rating_count'});
{ result: 'rating_count', ok: 1 }

```
Orders> db.orders.aggregate([ { $group: { _id: "$rating", count: { $sum: 1 } } } ]);
[
  { _id: 3, count: 1 },
  { _id: 5, count: 1 },
  { _id: 2, count: 1 },
  { _id: 1, count: 2 },
  { _id: 4, count: 1 }
]
```

21) Create a collection named rating that contain 5 documents of the following prototype and solve the following Queries. { movie_id: 123, user_id: 12, title: Toy Story(1995), status: 'A' }

a) Get ratings for the movie "Star Wars(1993)" using the descending ordered index on movie_id and explain.

b) Rebuild all indexes for the ratings collection.

c) Drop index on rating collection.

d) Create an index on movie_id and ratings fields together with movie_id (ascending order sorted) and rating (descending order sorted).

e) Create a descending order index on movie_id to get ratings related to "Avatar(2009)" verify the query plan.

OUTPUT:

```
Orders> use Rating

switched to db Rating

Rating> db.Rating.insert({ Movie_id:123, User_id:12, Title:"Avatar(2009)", Status:'A'})

{

  acknowledged: true,

  insertedIds: { '0': ObjectId('671542ac506ef8b41cc73bfd') }

}

Rating> db.Rating.insert({ Movie_id:423, User_id:13, Title:"Titanic(1997)", Status:'UA'})

{

  acknowledged: true,

  insertedIds: { '0': ObjectId('671542df506ef8b41cc73bfe') }

}

Rating> db.Rating.insert({ Movie_id:234, User_id:14, Title:"The Dark Knight(2007)", Status:'A'})

{

  acknowledged: true,

  insertedIds: { '0': ObjectId('6715430e506ef8b41cc73bff') }

}

Rating> db.Rating.insert({ Movie_id:1234, User_id:15, Title:"The Gun(1986)", Status:'A'})

{

  acknowledged: true,
  insertedIds: { '0': ObjectId('67154337506ef8b41cc73c00') }
}
Rating> db.Rating.insert({ Movie_id:254, User_id:16, Title:"Star Wars(1983)", Status:'UA'})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6715436b506ef8b41cc73c01') }
}
Rating> db.Rating.find().pretty()
[{
    _id: ObjectId('671542ac506ef8b41cc73bfd'),
    Movie_id: 123,
    User_id: 12,
    Title: 'Avatar(2009)',
    Status: 'A'
  },
  {
    _id: ObjectId('671542df506ef8b41cc73bfe'),
    Movie_id: 423,
    User_id: 13,
    Title: 'Titanic(1997)',
    Status: 'UA'
  },{
    _id: ObjectId('6715430e506ef8b41cc73bff'),
    Movie_id: 234,
    User_id: 14,
    Title: 'The Dark Knight(2007)',
    Status: 'A'
  },
  {
    _id: ObjectId('67154337506ef8b41cc73c00'),
    Movie_id: 1234,
    User_id: 15,
    Title: 'The Gun(1986)',

    Status: 'A'

  },

  {

    _id: ObjectId('6715436b506ef8b41cc73c01'),

    Movie_id: 254,

    User_id: 16,

    Title: 'Star Wars(1983)',

    Status: 'UA'

  }]
```

a. **Get ratings for the movie "Star Wars(1993)" using the descending ordered index on movie_id and explain**

```
Rating> db.Rating.find({ Title: "Star Wars(1983)" }).sort({ Movie_id: -1 });
[
  {
    _id: ObjectId('6715436b506ef8b41cc73c01'),
    Movie_id: 254,
    User_id: 16,
    Title: 'Star Wars(1983)',
    Status: 'UA'
  }
]
```

b. **Rebuild all indexes for the ratings collection**

```
Rating> db.Rating.reIndex();
{
  nIndexesWas: 4,
  nIndexes: 4,
  indexes: [
    { v: 2, key: { _id: 1 }, name: '_id_' },
    { v: 2, key: { movie_id: 1 }, name: 'movie_id_1' },
    { v: 2, key: { Movie_id: 1 }, name: 'Movie_id_1' },
    { v: 2, key: { Movie_id: -1 }, name: 'Movie_id_-1' }
  ],
  ok: 1
}
```

c. **Drop index on rating collection.**

```
Rating> db.Rating.dropIndex("Movie_id_1");
{ nIndexesWas: 4, ok: 1 }
```

d. **Create an index on Movie_id and ratings fields together with Movie_id (ascending order sorted) and rating (descending order sorted).**

```
Rating> db.Rating.dropIndex("Movie_id_1");
{ nIndexesWas: 4, ok: 1 }
Rating> db.Rating.getIndexes();
```

```
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { movie_id: 1 }, name: 'movie_id_1' },
  { v: 2, key: { Movie_id: -1 }, name: 'Movie_id_-1' },
  {
    v: 2,
    key: { Movie_id: 1, Status: -1 },
    name: 'Movie_id_1_Status_-1'
  }
]
```

e. **Create a descending order index on Movie_id to get ratings related to "Avatar(2009)" verify the query plan.**

```
Rating> db.Rating.createIndex({ Movie_id: -1 })
Movie_id_-1
Rating> db.Rating.find({ Title: "Avatar(2009)" })
[
  {
    _id: ObjectId('671542ac506ef8b41cc73bfd'),
    Movie_id: 123,
    User_id: 12,
    Title: 'Avatar(2009)',
    Status: 'A'
  }
]
```

22) Implement MongoDb database connectivity with Java Implement Database navigation operations (add, delete, edit,) using ODBC/JDBC.

OUTPUT:

```
package JavaMongo;

import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
import com.mongodb.client.model.Filters;
import org.bson.Document;
import org.bson.conversions.Bson;

import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class JavaMongo {

    public static void main(String[] args) {
        // Create a MongoDB client connection
        MongoClient mongoClient = MongoClients.create("mongodb://localhost:27017");
```

```
System.out.println("Created Mongo Connection successfully");

// Access the database
MongoDatabase db = mongoClient.getDatabase("ConnectivityJavaMongo");
System.out.println("Get database is successful");

// Access the collection
MongoCollection<Document> collection = db.getCollection("EXAMPLE1");

Scanner scanner = new Scanner(System.in);
int choice;

do {
  // Display the menu options
  System.out.println("\nChoose an operation:");
  System.out.println("1. Create");
  System.out.println("2. Read");
  System.out.println("3. Update");
  System.out.println("4. Delete");
  System.out.println("5. Exit");
  System.out.print("Enter your choice: ");
  choice = scanner.nextInt();
  scanner.nextLine(); // Consume the newline

  switch (choice) {
    case 1: // Create
      System.out.print("Enter name: ");
      String name = scanner.nextLine();
      System.out.print("Enter age: ");
      int age = scanner.nextInt();
      Document newDoc = new Document("name", name).append("age", age);
      collection.insertOne(newDoc);
      System.out.println("Document inserted.");
      break;

    case 2: // Read
      System.out.print("Enter name to search: ");
      String searchName = scanner.nextLine();
      Document foundDoc = collection.find(new Document("name", searchName)).first();
      if (foundDoc != null) {
        System.out.println("Found: " + foundDoc.toJson());
      } else {
        System.out.println("No document found with that name.");
      }
      break;

    case 3: // Update
      System.out.print("Enter name of the document to update: ");
```

```
                    String updateName = scanner.nextLine();
                    System.out.print("Enter new age: ");
                    int newAge = scanner.nextInt();
                    Document updatedDoc = new Document("age", newAge);
                    collection.updateOne(new Document("name", updateName), new Document("$set",
updatedDoc));
                    System.out.println("Document updated.");
                    break;

                case 4: // Delete
                    System.out.print("Enter name of the document to delete: ");
                    String deleteName = scanner.nextLine();
                    collection.deleteOne(new Document("name", deleteName));
                    System.out.println("Document deleted.");
                    break;

                case 5: // Exit
                    System.out.println("Exiting...");
                    break;

                default:
                    System.out.println("Invalid choice! Please choose a valid operation.");
                    break;
            }
        } while (choice != 5);

        // Close the connection
        mongoClient.close();
        System.out.println("MongoDB connection closed.");
        scanner.close();
    }
}
```

Output:

```
Created Mongo Connection successfully

Get database is successful


Choose an operation:

1. Create

2. Read

3. Update

4. Delete

5. Exit
```

```
Enter your choice: 1

Enter name: Vishal

Enter age: 20

Document inserted.


Choose an operation:

1. Create

2. Read

3. Update

4. Delete

5. Exit

Enter your choice: 1

Enter name: Raj

Enter age: 20

Document inserted.


Choose an operation:

1. Create

2. Read

3. Update

4. Delete

5. Exit

Enter your choice: 1

Enter name: Avinash

Enter age: 19

Document inserted.


Choose an operation:

1. Create

2. Read

3. Update

4. Delete
```

```
5. Exit
Enter your choice: 1
Enter name: Neha
Enter age: 20
Document inserted.

Choose an operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
Enter your choice: 2
Enter name to search: Neha
Found: {"_id": {"$oid": "6726518760cbce0c464cad18"}, "name": "Neha", "age": 20}

Choose an operation:
1. Create
2. Read
3. Update
4. Delete
5. Exit
Enter your choice: 3
Enter name of the document to update: Avinash
Enter new age: 20
Document updated.

Choose an operation:
1. Create
2. Read
3. Update
4. Delete
```

```
5. Exit

Enter your choice: 2

Enter name to search: Avinash

Found: {"_id": {"$oid": "6726517960cbce0c464cad17"}, "name": "Avinash", "age": 20}


Choose an operation:

1. Create

2. Read

3. Update

4. Delete

5. Exit

Enter your choice: 4

Enter name of the document to delete: Avinash

Document deleted.


Choose an operation:

1. Create

2. Read

3. Update

4. Delete

5. Exit

Enter your choice: 5

Exiting...

MongoDB connection closed.


Process finished with exit code 0
```

23) **Create a collection named Book. Add 5 documents in the collection with keys (book_isbn,title,punlisher_name,author(Name, Address, Phone No[landline, mobile]), publisher_city, price,copies)**

a) Select Book Names whose title is ”JAVA” .

b) Update Book Copies as “120” whose Book Publisher is “OracleCorporation”.

c) Display name of publishers as per no of books published by them in ascending order.

d) Get publisher names who published at least one book written by author name like ‘W%’.

e) Delete the book from Book table written by Author ‘XYZ’.

OUTPUT:

```
test> show dbs
Book     8.00 KiB
admin   40.00 KiB
config  60.00 KiB
local   40.00 KiB
test> use Book
switched to db Book
Book>
db.Book.insert({Book_isbn:1001,Title:"DBMS",P_name:"Technical",Author:{Name:"ABC",Address:"P
une",Phone_no:[{Landline:1234567989,Mobile:365897413}]},P_city:"Baramati",Price:250,Copies:15
0})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb477634a595b13c73bf8') }
}
Book> db.Book.find().pretty()
[
  {
    _id: ObjectId('66ffb477634a595b13c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  }
]
Book> db.Book.find()
[
  {
    _id: ObjectId('66ffb477634a595b13c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  }
]
Book>
db.Book.insert({Book_isbn:1002,Title:"JAVA",P_name:"Technical",Author:{Name:"PQR",Address:"Ba
ramati",Phone_no:[{Landline:24367989,Mobile:986897413}]},P_city:"Nashik",Price:350,Copies:50})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb57c634a595b13c73bf9') }
}
Book>
db.Book.insert({Book_isbn:1002,Title:"MYSQL",P_name:"OracleCorporation",Author:{Name:"XYZ",A
ddress:"Pune",Phone_no:[{Landline:2154359,Mobile:986541233}]},P_city:"Pune",Price:150,Copies:
70})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb631634a595b13c73bfa') }
}
Book> db.Book.insert({Book_isbn:1004,Title:"Data Analytics Made Accesible",P_name:"Data
Analytics",Author:{Name:"Anil
Maheshwari",Address:"Mumbai",Phone_no:[{Landline:21354359,Mobile:78941233}]},P_city:"Pune
",Price:250,Copies:70})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb729634a595b13c73bfb') }
}
Book> db.Book.insert({Book_isbn:1005,Title:"Python for Data Analytics",P_name:"Data Analytics
PY",Author:{Name:"William
McKinney",Address:"Pune",Phone_no:[{Landline:291354359,Mobile:978941233}]},P_city:"A.Nagar"
,Price:540,Copies:55})
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('66ffb7d5634a595b13c73bfc') }
}
Book> db.Book.find().pretty()
[
  {
    _id: ObjectId('66ffb477634a595b13c73bf8'),
    Book_isbn: 1001,
    Title: 'DBMS',
    P_name: 'Technical',
    Author: {
      Name: 'ABC',
      Address: 'Pune',
      Phone_no: [ { Landline: 1234567989, Mobile: 365897413 } ]
    },
    P_city: 'Baramati',
    Price: 250,
    Copies: 150
  },
  {
    _id: ObjectId('66ffb57c634a595b13c73bf9'),
    Book_isbn: 1002,
    Title: 'JAVA',
    P_name: 'Technical',
    Author: {
      Name: 'PQR',
      Address: 'Baramati',
      Phone_no: [ { Landline: 24367989, Mobile: 986897413 } ]
    },
    P_city: 'Nashik',
    Price: 350,
    Copies: 50
  },
  {
    _id: ObjectId('66ffb631634a595b13c73bfa'),
    Book_isbn: 1002,
    Title: 'MYSQL',
    P_name: 'OracleCorporation',
    Author: {
      Name: 'XYZ',
      Address: 'Pune',
      Phone_no: [ { Landline: 2154359, Mobile: 986541233 } ]
    },
    P_city: 'Pune',
    Price: 150,
    Copies: 70
  },
  {
    _id: ObjectId('66ffb729634a595b13c73bfb'),
    Book_isbn: 1004,
    Title: 'Data Analytics Made Accesible',
    P_name: 'Data Analytics',
    Author: {
      Name: 'Anil Maheshwari',
      Address: 'Mumbai',
      Phone_no: [ { Landline: 21354359, Mobile: 78941233 } ]
    },
    P_city: 'Pune',
    Price: 250,
    Copies: 70
  },
  {
    _id: ObjectId('66ffb7d5634a595b13c73bfc'),
    Book_isbn: 1005,
    Title: 'Python for Data Analytics',
    P_name: 'Data Analytics PY',
    Author: {
      Name: 'William McKinney',
      Address: 'Pune',
      Phone_no: [ { Landline: 291354359, Mobile: 978941233 } ]
    },
    P_city: 'A.Nagar',
    Price: 540,
    Copies: 55
  }
]
```

a) Select Book Names whose title is ”JAVA” .

```
Book> db.Book.find({ Title: "JAVA" });

[{

   _id: ObjectId('6713f13e9e77ca7f76c73bf9'),

   Book_isbn: 1002,

   Title: 'JAVA',

   P_name: 'Technical',

   Author: {

     Name: 'PQR',

     Address: 'Baramati',

     Phone_no: [ { Landline: 24367989, Mobile: 986897413 } ]

   },

   P_city: 'Nashik',

   Price: 350,

   Copies: 50}

]
```

b. Update Book Copies as “120” whose Book Publisher is “OracleCorporation”.

```
Book> db.Book.update({P_name:"OracleCorporation"},{$set:{Copies:120}})

DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

c. **Display name of publishers as per no of books published by them in ascending order.**

```
Book> db.Book.aggregate([{$group:{_id:"$P_name", totalBooks:{$sum:1}}},{$sort:{totalBooks:1}}])
[
  { _id: 'Data Analytics PY', totalBooks: 1 },
  { _id: 'Data Analytics', totalBooks: 1 },
  { _id: 'OracleCorporation', totalBooks: 1 },
  { _id: 'Technical', totalBooks: 2 }
]
```

d. **Get publisher names who published at least one book written by author name like ‘W%’.**

```
Book> db.Book.distinct("P_name", { "Author.Name": { $regex: "^W" } })
[ 'Data Analytics PY' ]
```

e. **Delete the book from Book table written by Author ‘XYZ’.**

```
Book> db.Book.deleteOne({ "Author.Name": "XYZ" })
{ acknowledged: true, deletedCount: 1 }
```

24) Consider following structure for MongoDB collections and write a query for following requirements in MongoDB

Teachers(Tname, dno, experience, salary, date_of joining)

Students(Sname, roll_no, class)

- **i) Write a MongoDB query to create above collections & for insertion of some sample documents.**
- **ii) Find the information about all teachers of dno = 2 and having salary greater than or equal to 10,000/-**
- **iii) Find the student information having roll_no = 2 or Sname = Anil**
- **iv) Display Total no of Students of TE Class**
- **v) update salary as 5% increment of teacher whose experience is >10 years.**

OUTPUT:

```
i)Book> db.Teachers.insertMany([
...   {
...     Tname: "John Doe",
...     dno: 2,
...     experience: 12,
...     salary: 12000,
...     date_of_joining: new Date("2010-01-15")
...   },
...   {
```

```
...         Tname: "Jane Smith",
...         dno: 1,
...         experience: 8,
...         salary: 9500,
...         date_of_joining: new Date("2015-03-10")
...     },
...     {
...         Tname: "Mike Johnson",
...         dno: 2,
...         experience: 15,
...         salary: 15000,
...         date_of_joining: new Date("2008-06-25")
...     }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67275617620f1f5f8cc73bfe'),
    '1': ObjectId('67275617620f1f5f8cc73bff'),
    '2': ObjectId('67275617620f1f5f8cc73c00')
  }
}
Book> db.Students.insertMany([
...     {
...         Sname: "Anil",
...         roll_no: 1,
...         class: "TE"
...     },
...     {
...         Sname: "Ravi",
...         roll_no: 2,
...         class: "SE"
...     },
...     {
...         Sname: "Priya",
...         roll_no: 3,
...         class: "TE"
...     },
...     {
...         Sname: "Sita",
...         roll_no: 4,
...         class: "BE"
...     }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6727562c620f1f5f8cc73c01'),
    '1': ObjectId('6727562c620f1f5f8cc73c02'),
    '2': ObjectId('6727562c620f1f5f8cc73c03'),
    '3': ObjectId('6727562c620f1f5f8cc73c04')
  }
}
```

```
ii) Book> db.Teachers.find({ dno: 2, salary: { $gte: 10000 } });
[
  {
    _id: ObjectId('67275617620f1f5f8cc73bfe'),
    Tname: 'John Doe',
    dno: 2,
    experience: 12,
    salary: 12000,
    date_of_joining: ISODate('2010-01-15T00:00:00.000Z')
  },
  {
    _id: ObjectId('67275617620f1f5f8cc73c00'),
    Tname: 'Mike Johnson',
    dno: 2,
    experience: 15,
    salary: 15000,
    date_of_joining: ISODate('2008-06-25T00:00:00.000Z')
  }
]
```

```
iii)Book> db.Students.find({ $or: [ { roll_no: 2 }, { Sname: "Anil" } ] });
[
  {
    _id: ObjectId('6727562c620f1f5f8cc73c01'),
    Sname: 'Anil',
    roll_no: 1,
    class: 'TE'
  },
  {
    _id: ObjectId('6727562c620f1f5f8cc73c02'),
    Sname: 'Ravi',
    roll_no: 2,
    class: 'SE'
  }
]
iv)Book> db.Students.countDocuments({ class: "TE" });
2
```

```
v)Book> db.Teachers.updateMany( { experience: { $gt: 10 } }, { $mul: { salary: 1.05 } }  );
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 2,
  modifiedCount: 2,
  upsertedCount: 0
}
```

25) Design a map-reduce operations on a collection “orders” that contains documents of the following prototype. Solve the following . { cust_id: "abc123", ord_date: new Date("Oct 04, 2012"), status: 'A', price: 25, gender :’F’, rating: 1 }

a) Count the number of female (F) and male (M) respondents in the orders collection

b) Count the number of each type of rating (1, 2, 3, 4 or 5) for each orders

OUTPUT:

```
Orders> db.orders.insertMany([
...     {
...         cust_id: "abc123",
...         ord_date: new Date("Oct 04, 2012"),
...         status: 'A',
...         price: 25,
...         gender: 'F',
...         rating: 1
...     },
...     {
...         cust_id: "def456",
...         ord_date: new Date("Nov 10, 2012"),
...         status: 'A',
...         price: 30,
...         gender: 'M',
...         rating: 3
...     },
...     {
...         cust_id: "ghi789",
...         ord_date: new Date("Dec 15, 2012"),
...         status: 'A',
...         price: 45,
...         gender: 'F',
...         rating: 5
...     },
...     {
...         cust_id: "jkl012",
...         ord_date: new Date("Jan 22, 2013"),
...         status: 'A',
...         price: 20,
...         gender: 'M',
...         rating: 2
...     },
...     {
...         cust_id: "mno345",
...         ord_date: new Date("Feb 28, 2013"),
...         status: 'A',
...         price: 50,
...         gender: 'F',
...         rating: 4
...     },
...     {
...         cust_id: "pqr678",
...         ord_date: new Date("Mar 12, 2013"),
...         status: 'A',
...         price: 15,
...         gender: 'M',
...         rating: 1
...     }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('67274614620f1f5f8cc73bf8'),
    '1': ObjectId('67274614620f1f5f8cc73bf9'),
    '2': ObjectId('67274614620f1f5f8cc73bfa'),
    '3': ObjectId('67274614620f1f5f8cc73bfb'),
    '4': ObjectId('67274614620f1f5f8cc73bfc'),
    '5': ObjectId('67274614620f1f5f8cc73bfd')
  }
}
```

a) Orders> db.Orders.mapReduce(function () {emit (this.gender, 1);}, function (key, values) {return Array.sum(values);},{out:'gender_count'});
{ result: 'gender_count', ok: 1 }

```
Orders> db.orders.aggregate([ { $group: { _id: "$gender", } } ]);

[ { _id: 'F' }, { _id: 'M' } ]
```

b) Orders> db.Orders.mapReduce(function () {emit (this.rating, 1);}, function (key, values) {return Array.sum(values);},{out:'rating_count'});
{ result: 'rating_count', ok: 1 }

```
Orders> db.orders.aggregate([ { $group: { _id: "$rating", count: { $sum: 1 } } } ]);
[
  { _id: 3, count: 1 },
  { _id: 5, count: 1 },
  { _id: 2, count: 1 },
  { _id: 1, count: 2 },
  { _id: 4, count: 1 }
]
```

SQL & PLSQL

26) Create the following table with the fields given below : PRODUCT (P_ID, Model, Price, Name, Date_of Manufacture, Date_of Expiry)

(a) Display name and date_of expiry of all the products whose price is more than 500.

(b) Display name, product_ID and price of all the products whose date_of manufacture is after "01-01-2018".

(c) Display name and date_of manufacture and date- of expiry of all the products whose price is between 5,000 and 10,000.

(d) Display name, product_ID and model of all the products which are going to expire after two months from today.

OUTPUT:

```
mysql> CREATE DATABASE PRODUCTION;
```

```
Query OK, 1 row affected (0.01 sec)


mysql> USE PRODUCTION;

Database changed

mysql> CREATE TABLE PRODUCT ( P_ID INT PRIMARY KEY, Model VARCHAR(50), Price DECIMAL(10, 2),
Name VARCHAR(100), Date_of_Manufacture DATE, Date_of_Expiry DATE );

Query OK, 0 rows affected (0.02 sec)


mysql> INSERT INTO PRODUCT (P_ID, Model, Price, Name, Date_of_Manufacture, Date_of_Expiry)
VALUES(1, 'Model A', 600, 'Product 1', '2017-05-01', '2023-05-01'),(2, 'Model B', 800, 'Product 2', '2018-02-
15', '2024-02-15'),(3, 'Model C', 300, 'Product 3', '2020-07-10', '2025-07-10'),(4, 'Model D', 9500, 'Product
4', '2021-03-20', '2024-09-20'),(5, 'Model E', 12000, 'Product 5', '2022-01-01', '2026-01-01');

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0


a)mysql> SELECT Name, Date_of_Expiry FROM PRODUCT WHERE Price > 500;

+-----------+----------------+

| Name      | Date_of_Expiry |

+-----------+----------------+

| Product 1 | 2023-05-01     |

| Product 2 | 2024-02-15     |

| Product 4 | 2024-09-20     |

| Product 5 | 2026-01-01     |

+-----------+----------------+

4 rows in set (0.00 sec)


b)mysql> SELECT Name, P_ID, Price FROM PRODUCT WHERE Date_of_Manufacture > '2018-01-01';

+-----------+------+----------+

| Name      | P_ID | Price    |

+-----------+------+----------+

| Product 2 |    2 |   800.00 |

| Product 3 |    3 |   300.00 |

| Product 4 |    4 |  9500.00 |

| Product 5 |    5 | 12000.00 |
+-----------+------+----------+
4 rows in set (0.00 sec)
```

c)mysql> SELECT Name, Date_of_Manufacture, Date_of_Expiry FROM PRODUCT WHERE Price BETWEEN 5000 AND 10000;

```
+-----------+---------------------+----------------+
| Name      | Date_of_Manufacture | Date_of_Expiry |
+-----------+---------------------+----------------+
| Product 4 | 2021-03-20          | 2024-09-20     |
+-----------+---------------------+----------------+
1 row in set (0.00 sec)
```

d)mysql> SELECT Name, P_ID, Model FROM PRODUCT WHERE Date_of_Expiry > DATE_ADD(CURRENT_DATE, INTERVAL 2 MONTH);

```
+-----------+------+---------+
| Name      | P_ID | Model   |
+-----------+------+---------+
| Product 3 |    3 | Model C |
| Product 5 |    5 | Model E |
+-----------+------+---------+
2 rows in set (0.00 sec)
```

27) Create a table named STUDENT with the following fields : 20 (FIRST NAME, MIDDLE NAME, LAST NAME, STUDENT_ENRLNO, DATE_OF_BIRTH, CLASS, SECTION, GENDER, YEAR_OF JOIN, ADMISSION_NO, ADDRESS1, ADDRESS2, CITY, STATE, RESPHONE, PIN_CODE)

(a) Display all the list of students who are in class - 6, section - A.

(b) To display all the students list whose first name starts with "A".

(c) To display all the students list who are girls.

(d) To display all the students whose YEAR-OF-JOIN is 2000.

(e) Sort the records of students with respect to their ADMISSION_NO, in ascending order

OUTPUT:

```
mysql> CREATE DATABASE STUD;

Query OK, 1 row affected (0.01 sec)
```

```
mysql> USE STUD;

Database changed

mysql> CREATE TABLE STUDENT ( FIRST_NAME VARCHAR(50), MIDDLE_NAME VARCHAR(50), LAST_NAME
VARCHAR(50), STUDENT_ENRLNO INT PRIMARY KEY, DATE_OF_BIRTH DATE, CLASS INT, SECTION CHAR(1),
GENDER CHAR(1), YEAR_OF_JOIN INT, ADMISSION_NO INT, ADDRESS1 VARCHAR(100), ADDRESS2
VARCHAR(100), CITY VARCHAR(50), STATE VARCHAR(50), RESPHONE VARCHAR(15), PIN_CODE
VARCHAR(10) );

Query OK, 0 rows affected (0.02 sec)


mysql> INSERT INTO STUDENT (FIRST_NAME, MIDDLE_NAME, LAST_NAME, STUDENT_ENRLNO,
DATE_OF_BIRTH, CLASS, SECTION, GENDER, YEAR_OF_JOIN, ADMISSION_NO, ADDRESS1, ADDRESS2, CITY,
STATE, RESPHONE, PIN_CODE) VALUES('Alice', 'Marie', 'Johnson', 1, '2010-05-20', 6, 'A', 'F', 2020, 1001, '123
Maple St', 'Apt 4B', 'Springfield', 'IL', '555-0123', '62701'),('Michael', 'David', 'Smith', 2, '2009-09-15', 6, 'B',
'M', 2020, 1002, '456 Oak St', '', 'Springfield', 'IL', '555-4567', '62702'),('Annie', 'Rose', 'Taylor', 3, '2011-01-
12', 5, 'A', 'F', 2021, 1003, '789 Pine St', '', 'Springfield', 'IL', '555-7890', '62703'),('Emma', 'Grace', 'Davis', 4,
'2008-03-30', 6, 'A', 'F', 2020, 1004, '321 Elm St', 'Suite 101', 'Springfield', 'IL', '555-2345', '62704'),('Daniel',
'Lee', 'Brown', 5, '2010-12-10', 5, 'B', 'M', 2021, 1005, '654 Cedar St', '', 'Springfield', 'IL', '555-6789',
'62705');

Query OK, 5 rows affected (0.01 sec)

Records: 5  Duplicates: 0  Warnings: 0
```

(a) Display all the list of students who are in class - 6, section - A.

```
mysql> SELECT * FROM STUDENT WHERE CLASS = 6 AND SECTION = 'A';

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+-----------+-------------+-------+----------+----------+

| FIRST_NAME | MIDDLE_NAME | LAST_NAME | STUDENT_ENRLNO | DATE_OF_BIRTH | CLASS | SECTION
| GENDER | YEAR_OF_JOIN | ADMISSION_NO | ADDRESS1     | ADDRESS2  | CITY        | STATE | RESPHONE |
PIN_CODE |

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+-----------+-------------+-------+----------+----------+

| Alice      | Marie       | Johnson   |              1 | 2010-05-20    |     6 | A       | F      |         2020 |         1001 | 123
Maple St | Apt 4B    | Springfield | IL    | 555-0123 | 62701    |

| Emma       | Grace       | Davis     |              4 | 2008-03-30    |     6 | A       | F      |         2020 |         1004 | 321
Elm St   | Suite 101 | Springfield | IL    | 555-2345 | 62704    |

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+-----------+-------------+-------+----------+----------+

2 rows in set (0.00 sec)
```

(b) To display all the students list whose first name starts with "A".

```
mysql> SELECT * FROM STUDENT WHERE FIRST_NAME LIKE 'A%';

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+----------+-------------+-------+----------+----------+

| FIRST_NAME | MIDDLE_NAME | LAST_NAME | STUDENT_ENRLNO | DATE_OF_BIRTH | CLASS | SECTION
| GENDER | YEAR_OF_JOIN | ADMISSION_NO | ADDRESS1     | ADDRESS2 | CITY        | STATE | RESPHONE |
PIN_CODE |

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+----------+-------------+-------+----------+----------+

| Alice      | Marie       | Johnson   |              1 | 2010-05-20    |     6 | A       | F      |         2020 |         1001 | 123
Maple St | Apt 4B   | Springfield | IL    | 555-0123 | 62701    |

| Annie      | Rose        | Taylor    |              3 | 2011-01-12    |     5 | A       | F      |         2021 |         1003 | 789
Pine St  |          | Springfield | IL    | 555-7890 | 62703    |

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+----------+-------------+-------+----------+----------+

2 rows in set (0.00 sec)
```

(c) To display all the students list who are girls.

```
mysql> SELECT * FROM STUDENT WHERE GENDER = 'F';

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+-----------+-------------+-------+----------+----------+

| FIRST_NAME | MIDDLE_NAME | LAST_NAME | STUDENT_ENRLNO | DATE_OF_BIRTH | CLASS | SECTION
| GENDER | YEAR_OF_JOIN | ADMISSION_NO | ADDRESS1     | ADDRESS2  | CITY        | STATE | RESPHONE |
PIN_CODE |

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+-----------+-------------+-------+----------+----------+

| Alice      | Marie       | Johnson   |              1 | 2010-05-20    |     6 | A       | F      |         2020 |         1001 | 123
Maple St | Apt 4B    | Springfield | IL    | 555-0123 | 62701    |

| Annie      | Rose        | Taylor    |              3 | 2011-01-12    |     5 | A       | F      |         2021 |         1003 | 789
Pine St  |           | Springfield | IL    | 555-7890 | 62703    |

| Emma       | Grace       | Davis     |              4 | 2008-03-30    |     6 | A       | F      |         2020 |         1004 | 321
Elm St   | Suite 101 | Springfield | IL    | 555-2345 | 62704    |

+------------+-------------+-----------+----------------+---------------+-------+---------+--------+--------------+--------------+-----
---------+-----------+-------------+-------+----------+----------+

3 rows in set (0.00 sec)
```

(d) To display all the students whose YEAR-OF-JOIN is 2000.

```
mysql> SELECT * FROM STUDENT WHERE YEAR_OF_JOIN = 2000;
```

```
Empty set (0.00 sec)
```

(e) Sort the records of students with respect to their ADMISSION_NO, in ascending order

```
mysql> SELECT *FROM STUDENT ORDER BY ADMISSION_NO ASC;

+------------+-------------+-----------+---------------+--------------+-------+---------+--------+-------------+-------------+-----
---------+-----------+------------+-------+----------+----------+

| FIRST_NAME | MIDDLE_NAME | LAST_NAME | STUDENT_ENRLNO | DATE_OF_BIRTH | CLASS | SECTION
| GENDER | YEAR_OF_JOIN | ADMISSION_NO | ADDRESS1     | ADDRESS2  | CITY       | STATE | RESPHONE |
PIN_CODE |

+------------+-------------+-----------+---------------+--------------+-------+---------+--------+-------------+-------------+-----
---------+-----------+------------+-------+----------+----------+

| Alice      | Marie       | Johnson   |             1 | 2010-05-20   |     6 | A       | F      |        2020 |        1001 | 123
Maple St | Apt 4B    | Springfield | IL    | 555-0123 | 62701    |

| Michael    | David       | Smith     |             2 | 2009-09-15   |     6 | B       | M      |        2020 |        1002 |
456 Oak St   |           | Springfield | IL    | 555-4567 | 62702    |

| Annie      | Rose        | Taylor    |             3 | 2011-01-12   |     5 | A       | F      |        2021 |        1003 | 789
Pine St  |           | Springfield | IL    | 555-7890 | 62703    |

| Emma       | Grace       | Davis     |             4 | 2008-03-30   |     6 | A       | F      |        2020 |        1004 | 321
Elm St   | Suite 101 | Springfield | IL    | 555-2345 | 62704    |

| Daniel     | Lee         | Brown     |             5 | 2010-12-10   |     5 | B       | M      |        2021 |        1005 | 654
Cedar St |           | Springfield | IL    | 555-6789 | 62705    |

+------------+-------------+-----------+---------------+--------------+-------+---------+--------+-------------+-------------+-----
---------+-----------+------------+-------+----------+----------+

5 rows in set (0.00 sec)
```

28)Create the following table CATALOG with the following fields : (BOOK ID, BOOK TITLE, AUTHOR, AUTHOR_ID, PUBLISHER_ID, CATEGORY_ID, YEAR, ISBN, PRICE)

(a) To display all the books of the CATEGORY_ID : "COMPUTERS".

(b) List all the books whose PRICE is greater than or equal to 1000/-.

(c) List all the books whose PUBLISHER_ID is "Tata McGraw-Hill".

(d) List all the books whose YEAR of publication is 2013.

(e) List all the BOOK_TITLEs whose AUTHOR_ID is "123".

OUTPUT:

```
mysql> CREATE DATABASE CATALOG;

Query OK, 1 row affected (0.01 sec)
```

```
mysql> USE CATALOG;

Database changed

mysql> CREATE TABLE CATALOG ( BOOK_ID INT PRIMARY KEY, BOOK_TITLE VARCHAR(100), AUTHOR
VARCHAR(100), AUTHOR_ID VARCHAR(50), PUBLISHER_ID VARCHAR(50), CATEGORY_ID VARCHAR(50),
YEAR INT, ISBN VARCHAR(20), PRICE DECIMAL(10, 2) );

Query OK, 0 rows affected (0.02 sec)


mysql> INSERT INTO CATALOG (BOOK_ID, BOOK_TITLE, AUTHOR, AUTHOR_ID, PUBLISHER_ID,
CATEGORY_ID, YEAR, ISBN, PRICE) VALUES(1, 'Learning Python', 'Mark Lutz', '101', 'OReilly', 'COMPUTERS',
2013, '978-1449365480', 1500.00),(2, 'Introduction to Algorithms', 'Thomas H. Cormen', '102', 'MIT Press',
'COMPUTERS', 2013, '978-0262033848', 2000.00),(3, 'Head First Java', 'Kathy Sierra', '103', 'OReilly',
'COMPUTERS', 2012, '978-0596009205', 1200.00),(4, 'Data Science from Scratch', 'Joel Grus', '104', 'Tata
McGraw-Hill', 'DATA_SCIENCE', 2015, '978-1492041139', 800.00),(5, 'Effective Java', 'Joshua Bloch', '105',
'Addison-Wesley', 'COMPUTERS', 2018, '978-0134686097', 1800.00);

Query OK, 5 rows affected (0.00 sec)

Records: 5  Duplicates: 0  Warnings: 0
```

(a) To display all the books of the CATEGORY_ID : "COMPUTERS".

```
mysql> SELECT *FROM CATALOG WHERE CATEGORY_ID = 'COMPUTERS';

+---------+----------------------------+------------------+-----------+----------------+------------+------+----------------+---------+

| BOOK_ID | BOOK_TITLE                 | AUTHOR           | AUTHOR_ID | PUBLISHER_ID   | CATEGORY_ID | YEAR
| ISBN           | PRICE   |

+---------+----------------------------+------------------+-----------+----------------+------------+------+----------------+---------+

|       1 | Learning Python            | Mark Lutz        | 101       | OReilly        | COMPUTERS   | 2013 | 978-
1449365480 | 1500.00 |

|       2 | Introduction to Algorithms | Thomas H. Cormen | 102       | MIT Press      | COMPUTERS   | 2013 |
978-0262033848 | 2000.00 |

|       3 | Head First Java            | Kathy Sierra     | 103       | OReilly        | COMPUTERS   | 2012 | 978-
0596009205 | 1200.00 |

|       5 | Effective Java             | Joshua Bloch     | 105       | Addison-Wesley | COMPUTERS   | 2018 | 978-
0134686097 | 1800.00 |

+---------+----------------------------+------------------+-----------+----------------+------------+------+----------------+---------+

4 rows in set (0.00 sec)
```

(b) List all the books whose PRICE is greater than or equal to 1000/-.

```
mysql> SELECT *FROM CATALOG WHERE PRICE >= 1000;

+---------+----------------------------+------------------+-----------+----------------+------------+------+----------------+---------+
| BOOK_ID | BOOK_TITLE              | AUTHOR          | AUTHOR_ID | PUBLISHER_ID   | CATEGORY_ID | YEAR
| ISBN          | PRICE   |

+---------+--------------------------+-----------------+-----------+----------------+-------------+------+---------------+---------+

|       1 | Learning Python          | Mark Lutz       | 101       | OReilly        | COMPUTERS   | 2013 | 978-
1449365480 | 1500.00 |

|       2 | Introduction to Algorithms | Thomas H. Cormen | 102       | MIT Press      | COMPUTERS   | 2013 |
978-0262033848 | 2000.00 |

|       3 | Head First Java          | Kathy Sierra    | 103       | OReilly        | COMPUTERS   | 2012 | 978-
0596009205 | 1200.00 |

|       5 | Effective Java           | Joshua Bloch    | 105       | Addison-Wesley | COMPUTERS   | 2018 | 978-
0134686097 | 1800.00 |

+---------+--------------------------+-----------------+-----------+----------------+-------------+------+---------------+---------+

4 rows in set (0.00 sec)
```

(c) List all the books whose PUBLISHER_ID is "Tata McGraw-Hill".

```
mysql> SELECT *FROM CATALOG WHERE PUBLISHER_ID = 'Tata McGraw-Hill';

+---------+--------------------------+-----------+-----------+------------------+--------------+------+----------------+--------+

| BOOK_ID | BOOK_TITLE              | AUTHOR    | AUTHOR_ID | PUBLISHER_ID     | CATEGORY_ID  | YEAR |
ISBN           | PRICE  |

+---------+--------------------------+-----------+-----------+------------------+--------------+------+----------------+--------+

|       4 | Data Science from Scratch | Joel Grus | 104       | Tata McGraw-Hill | DATA_SCIENCE | 2015 | 978-
1492041139 | 800.00 |

+---------+--------------------------+-----------+-----------+------------------+--------------+------+----------------+--------+

1 row in set (0.00 sec)
```

(d) List all the books whose YEAR of publication is 2013.

```
mysql> SELECT *FROM CATALOG WHERE YEAR = 2013;

+---------+--------------------------+-----------------+-----------+--------------+-------------+------+----------------+---------+

| BOOK_ID | BOOK_TITLE              | AUTHOR          | AUTHOR_ID | PUBLISHER_ID | CATEGORY_ID | YEAR |
ISBN           | PRICE   |

+---------+--------------------------+-----------------+-----------+--------------+-------------+------+----------------+---------+

|       1 | Learning Python          | Mark Lutz       | 101       | OReilly      | COMPUTERS   | 2013 | 978-
1449365480 | 1500.00 |

|       2 | Introduction to Algorithms | Thomas H. Cormen | 102       | MIT Press    | COMPUTERS   | 2013 |
978-0262033848 | 2000.00 |

+---------+--------------------------+-----------------+-----------+--------------+-------------+------+----------------+---------+
2 rows in set (0.00 sec)
```

(e) List all the BOOK_TITLEs whose AUTHOR_ID is "123".

```
mysql> SELECT BOOK_TITLE FROM CATALOG WHERE AUTHOR_ID = '123';

Empty set (0.00 sec)
```

29) Create the following tables :Student(roll-no, name, date-of-birth, course id)Course (Course-id, name, fee, duration, status) Write PL/SQL procedure to do the following :Set the status of course to "not offered" in which the number of candidates is less than 5

OUTPUT:

```
SQL> connect system

Enter password:

Connected.

SQL> set serveroutput on;

SQL> CREATE TABLE Course ( Course_id INT PRIMARY KEY, Name VARCHAR(100), Fee DECIMAL(10, 2),
Duration VARCHAR(50), Status VARCHAR(20) );

Table created.


SQL> CREATE TABLE Student (    Roll_no INT PRIMARY KEY,    Name VARCHAR(100),    Date_of_Birth DATE,
Course_id INT,    FOREIGN KEY (Course_id) REFERENCES Course(Course_id));

Table created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (1, 'Computer Science', 50000,
'4 Years', 'offered');

1 row created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (2, 'Mechanical Engineering',
55000, '4 Years', 'offered');

1 row created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (3, 'Civil Engineering', 52000, '4
Years', 'offered');

1 row created.
```

```
SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (4, 'Electronics Engineering', 53000, '4 Years', 'offered');

1 row created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (5, 'Data Science', 60000, '2 Years', 'offered');

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (101, 'Alice Smith', TO_DATE('2000-05-20', 'YYYY-MM-DD'), 1);

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (102, 'Bob Johnson', TO_DATE('2001-04-15', 'YYYY-MM-DD'), 1);

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (103, 'Charlie Brown', TO_DATE('2000-06-10', 'YYYY-MM-DD'), 2);

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (104, 'David Wilson', TO_DATE('2002-08-30', 'YYYY-MM-DD'), 2);

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (105, 'Eve Davis', TO_DATE('2001-12-01', 'YYYY-MM-DD'), 3);

1 row created.

SQL> ed

Wrote file afiedt.buf

  1  CREATE OR REPLACE PROCEDURE UpdateCourseStatus AS

  2  BEGIN

  3     UPDATE Course c

  4     SET Status = 'not offered'

  5     WHERE c.Course_id IN (

  6        SELECT Course_id
```

```
 7        FROM Student
 8        GROUP BY Course_id
 9        HAVING COUNT(*) < 5
10      );
11      COMMIT;
12* END;
13  /
Procedure created.


SQL> ed
Wrote file afiedt.buf
  1  BEGIN
  2     UpdateCourseStatus;
  3* END;
SQL> SELECT * FROM Course WHERE Status = 'not offered';
no rows selected


SQL> SELECT Status, COUNT(*) AS Total FROM Course GROUP BY Status;
STATUS                   TOTAL
-------------------- ----------
offered                      5
```

30) Write PL/SQL procedure to do the following :Set the status of course to "offered" in which the number of candidates is atleast 10 otherwise set it to "not offered"

OUTPUT:

```
SQL> connect system
Enter password:
Connected.
SQL> set serveroutput on;
SQL> CREATE TABLE Course ( Course_id INT PRIMARY KEY, Name VARCHAR(100), Fee DECIMAL(10, 2),
Duration VARCHAR(50), Status VARCHAR(20) );
Table created.
```

```
SQL> CREATE TABLE Student (    Roll_no INT PRIMARY KEY,    Name VARCHAR(100),    Date_of_Birth DATE,
Course_id INT,    FOREIGN KEY (Course_id) REFERENCES Course(Course_id));

Table created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (1, 'Computer Science', 50000,
'4 Years', 'offered');

1 row created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (2, 'Mechanical Engineering',
55000, '4 Years', 'offered');

1 row created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (3, 'Civil Engineering', 52000, '4
Years', 'offered');

1 row created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (4, 'Electronics Engineering',
53000, '4 Years', 'offered');

1 row created.


SQL> INSERT INTO Course (Course_id, Name, Fee, Duration, Status) VALUES (5, 'Data Science', 60000, '2
Years', 'offered');

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (101, 'Alice Smith',
TO_DATE('2000-05-20', 'YYYY-MM-DD'), 1);

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (102, 'Bob Johnson',
TO_DATE('2001-04-15', 'YYYY-MM-DD'), 1);

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (103, 'Charlie Brown',
TO_DATE('2000-06-10', 'YYYY-MM-DD'), 2);

1 row created.
```

```
SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (104, 'David Wilson',
TO_DATE('2002-08-30', 'YYYY-MM-DD'), 2);

1 row created.


SQL> INSERT INTO Student (Roll_no, Name, Date_of_Birth, Course_id) VALUES (105, 'Eve Davis',
TO_DATE('2001-12-01', 'YYYY-MM-DD'), 3);

1 row created.


SQL> ed

Wrote file afiedt.buf


  1  CREATE OR REPLACE PROCEDURE UpdateCourseStatus AS
  2  BEGIN
  3     UPDATE Course c
  4     SET c.Status = CASE
  5         WHEN (SELECT COUNT(*) FROM Student s WHERE s.Course_id = c.Course_id) >= 10 THEN 'offered'
  6         ELSE 'not offered'
  7     END;
  8     COMMIT;
  9* END;
 10  /


Procedure created.


SQL> ed

Wrote file afiedt.buf


  1  BEGIN
  2     UpdateCourseStatus;
  3* END;
  4  /
```

```
PL/SQL procedure successfully completed.

SQL> SELECT * FROM Course;

 COURSE_ID
----------
NAME
--------------------------------------------------------------------------------
       FEE DURATION
---------- --------------------------------------------------
STATUS
--------------------
         1
Computer Science
     50000 4 Years
not offered


 COURSE_ID
----------
NAME
--------------------------------------------------------------------------------
       FEE DURATION
---------- --------------------------------------------------
STATUS
--------------------
         2
Mechanical Engineering
     55000 4 Years
not offered
```

```
 COURSE_ID

----------

NAME

--------------------------------------------------------------------------------

      FEE DURATION

---------- --------------------------------------------------

STATUS

--------------------

        3

Civil Engineering

    52000 4 Years

not offered



 COURSE_ID

----------

NAME

--------------------------------------------------------------------------------

      FEE DURATION

---------- --------------------------------------------------

STATUS

--------------------

        4

Electronics Engineering

    53000 4 Years

not offered



 COURSE_ID

----------

NAME

--------------------------------------------------------------------------------
```

```
      FEE DURATION
---------- ------------------------------------------------
STATUS
--------------------
        5
Data Science
     60000 2 Years
not offered
```

MONGODB

31) { "address": { "building": "1007", "coord": [-73.856077, 40.848447], "street": "Morris Park Ave", "zipcode": "10462" }, "borough": "Bronx", "cuisine": "Bakery", "grades": [{ "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 }, { "date": { "$date": 1378857600000 }, "grade": "A", "score": 6 }, { "date": { "$date": 1358985600000 }, "grade": "A", "score": 10 }, { "date": { "$date": 1322006400000 }, "grade": "A", "score": 9 }, { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }], "name": "Morris Park Bake Shop", "restaurant_id": "30075445" }

a.Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

b. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

c. Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant

OUTPUT:

```
test> db.createCollection("restaurant");
{ ok: 1 }
test> db.restaurant.insertOne({
...   "address": {
...     "building": "1007",
...     "coord": [-73.856077, 40.848447],
...     "street": "Morris Park Ave",
...     "zipcode": "10462"
...   },
...   "borough": "Bronx",
...   "cuisine": "Bakery",
...   "grades": [
...     { "date": new Date(1393804800000), "grade": "A", "score": 2 },
...     { "date": new Date(1378857600000), "grade": "A", "score": 6 },
...     { "date": new Date(1358985600000), "grade": "A", "score": 10 },
...     { "date": new Date(1322006400000), "grade": "A", "score": 9 },
...     { "date": new Date(1299715200000), "grade": "B", "score": 14 }
...   ],
...   "name": "Morris Park Bake Shop",
...   "restaurant_id": "30075445"
... });
{
  acknowledged: true,
  insertedId: ObjectId('6727609afb98a1c7bac73bf8')
}
```

a.Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

```
test> db.restaurant.find( {}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } );
[
  {
    _id: ObjectId('6727609afb98a1c7bac73bf8'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
```

b. Write a MongoDB query to display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.

```
test> db.restaurant.find( {}, { restaurant_id: 1, name: 1, borough: 1, cuisine: 1 } );
[
  {
    _id: ObjectId('6727609afb98a1c7bac73bf8'),
    borough: 'Bronx',
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
```

c. Write a MongoDB query to display the fields restaurant_id, name, borough and zip code, but exclude the field _id for all the documents in the collection restaurant

```
test> db.restaurant.find( {}, { restaurant_id: 1, name: 1, borough: 1, "address.zipcode": 1, _id: 0 } );
[
  {
    address: { zipcode: '10462' },
    borough: 'Bronx',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  }
]
```