Experiment 7

Student Name: Sikander Singh Nanglu **UID: 22BET10031**
Branch: IT **Section/Group: 22BET_IOT-701/A**
Semester: 6th **Date of Performance:04/02/25**
Subject Name: PBLJ Lab **Subject Code: 22ITH-359**

Problem 1

1.Aim: To develop a Java application using JDBC and MVC architecture that allows efficient management of student records.

2.Objective:

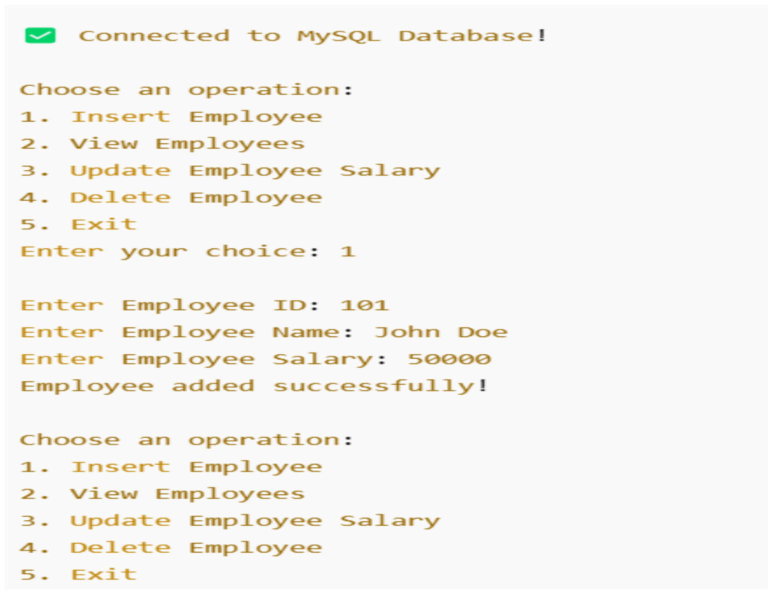
- Implement a MySQL database to store student information securely.
- Develop a Student model class with attributes: `StudentID`, `Name`, `Department`, and `Marks`.
- Design a Controller class to handle database interactions using JDBC.
- Apply the MVC (Model-View-Controller) architecture to ensure a clean and modular code structure.

3.Code:

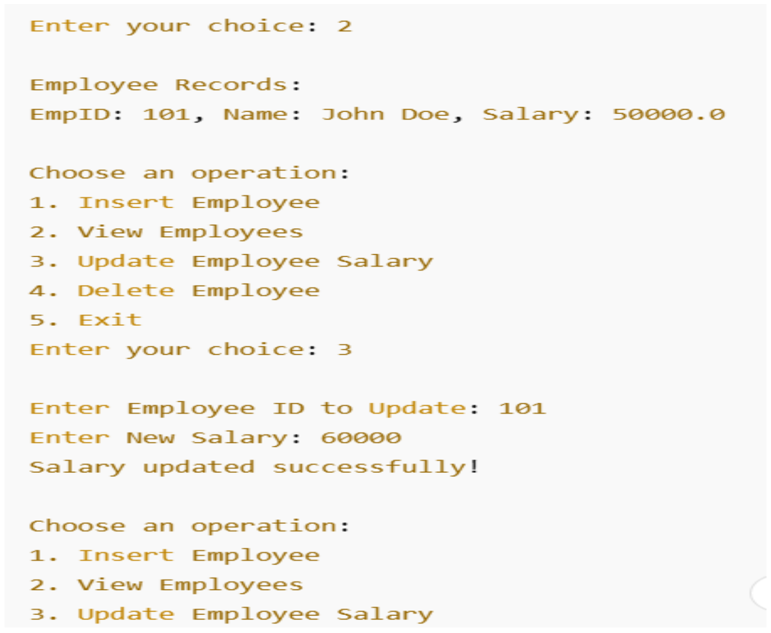
```
  import java.sql.*;
  import java.util.Scanner;

    public class EmployeeApp {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/company";
    private static final String DB_USER = "root"; // Change if needed
    private static final String DB_PASSWORD = "Shivansh@123";

    public static void main(String[] args) {
        Scanner inputScanner = new Scanner(System.in);
        try {
            // Load MySQL JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Establish connection
            Connection dbConnection = DriverManager.getConnection(DB_URL, DB_USER,
DB_PASSWORD);
            System.out.println("✅ Connected to MySQL Database!");

            while (true) {
                System.out.println("\nChoose an operation:");
                System.out.println("1. Insert Employee");
                System.out.println("2. View Employees");
                System.out.println("3. Update Employee Salary");
                System.out.println("4. Delete Employee");
                System.out.println("5. Exit");
                System.out.print("Enter your choice: ");
                int userChoice = inputScanner.nextInt();

                switch (userChoice) {
                    case 1:
                        addEmployee(dbConnection, inputScanner);
                        break;
                    case 2:
                        displayEmployees(dbConnection);
                        break;
                    case 3:
                        modifyEmployeeSalary(dbConnection, inputScanner);
                        break;
                    case 4:
                        removeEmployee(dbConnection, inputScanner);
                        break;
                    case 5:
                        System.out.println(" Exiting program...");
                        dbConnection.close();
                        inputScanner.close();
                        System.exit(0);
                        break;
                    default:
                        System.out.println(" Invalid choice. Try again!");
                }
            }
        } catch (ClassNotFoundException e) {
            System.out.println(" MySQL JDBC Driver Not Found!");
            e.printStackTrace();
        } catch (SQLException e) {
            System.out.println("❌ Database Connection Error!");
            e.printStackTrace();
        }
    }

    private static void addEmployee(Connection dbConnection, Scanner inputScanner) throws SQLException
{
        System.out.print("Enter Employee ID: ");
        int employeeID = inputScanner.nextInt();
        inputScanner.nextLine(); // Consume newline

        System.out.print("Enter Employee Name: ");
        String employeeName = inputScanner.nextLine();

        System.out.print("Enter Employee Salary: ");
        double employeeSalary = inputScanner.nextDouble();
        String sqlQuery = "INSERT INTO Employee (EmpID, Name, Salary) VALUES (?, ?, ?)";
        PreparedStatement preparedStatement = dbConnection.prepareStatement(sqlQuery);
        preparedStatement.setInt(1, employeeID);
        preparedStatement.setString(2, employeeName);
        preparedStatement.setDouble(3, employeeSalary);

        int affectedRows = preparedStatement.executeUpdate();
        if (affectedRows > 0) {
            System.out.println(" Employee added successfully!");
        }
    }

    private static void displayEmployees(Connection dbConnection) throws SQLException {
        String sqlQuery = "SELECT * FROM Employee";
        Statement statement = dbConnection.createStatement();
        ResultSet resultSet = statement.executeQuery(sqlQuery);

        System.out.println("\nEmployee Records:");
        while (resultSet.next()) {
            System.out.println("EmpID: " + resultSet.getInt("EmpID") +
                               ", Name: " + resultSet.getString("Name") +
                               ", Salary: " + resultSet.getDouble("Salary"));
        }
    }

    private static void modifyEmployeeSalary(Connection dbConnection, Scanner inputScanner) throws
SQLException {
        System.out.print("Enter Employee ID to Update: ");
        int employeeID = inputScanner.nextInt();

        System.out.print("Enter New Salary: ");
        double updatedSalary = inputScanner.nextDouble();

        String sqlQuery = "UPDATE Employee SET Salary = ? WHERE EmpID = ?";
        PreparedStatement preparedStatement = dbConnection.prepareStatement(sqlQuery);
        preparedStatement.setDouble(1, updatedSalary);
        preparedStatement.setInt(2, employeeID);

        int affectedRows = preparedStatement.executeUpdate();
        if (affectedRows > 0) {
            System.out.println("Salary updated successfully!");
        } else {
            System.out.println(" Employee not found!");
        }
    }

    private static void removeEmployee(Connection dbConnection, Scanner inputScanner) throws
SQLException {
        System.out.print("Enter Employee ID to Delete: ");
```

```
            int employeeID = inputScanner.nextInt();

            String sqlQuery = "DELETE FROM Employee WHERE EmpID = ?";
            PreparedStatement preparedStatement = dbConnection.prepareStatement(sqlQuery);
            preparedStatement.setInt(1, employeeID);

            int affectedRows = preparedStatement.executeUpdate();
            if (affectedRows > 0) {
                System.out.println(" Employee deleted successfully!");
            } else {
                System.out.println(" Employee not found!");
            }
        }
    }
```

4.OUTPUT:



```
✅ Connected to MySQL Database!

Choose an operation:
1. Insert Employee
2. View Employees
3. Update Employee Salary
4. Delete Employee
5. Exit
Enter your choice: 1

Enter Employee ID: 101
Enter Employee Name: John Doe
Enter Employee Salary: 50000
Employee added successfully!

Choose an operation:
1. Insert Employee
2. View Employees
3. Update Employee Salary
4. Delete Employee
5. Exit
```



```
Enter your choice: 2

Employee Records:
EmpID: 101, Name: John Doe, Salary: 50000.0

Choose an operation:
1. Insert Employee
2. View Employees
3. Update Employee Salary
4. Delete Employee
5. Exit
Enter your choice: 3

Enter Employee ID to Update: 101
Enter New Salary: 60000
Salary updated successfully!

Choose an operation:
1. Insert Employee
2. View Employees
3. Update Employee Salary
```

```
4. Delete Employee
5. Exit
Enter your choice: 4

Enter Employee ID to Delete: 101
Employee deleted successfully!
```

5.Learning Outcomes:

- Understand JDBC (Java Database Connectivity): Learn how to establish a connection between a Java application and a MySQL database.
- Use DriverManager and Connection objects: Implement DriverManager to manage database connections.
- Execute SQL Queries in Java: Fetch and display records from a MySQL database using Statement and ResultSet.
- Enhance Practical Knowledge: Gain hands-on experience with database operations in Java applications.
- Improve Error Handling: Implement proper exception handling for database connectivity issues.

Problem 2

1.Aim:

To design and implement a Java-based application using JDBC that efficiently manages product data in a MySQL database, ensuring seamless interaction through CRUD (Create, Read, Update, Delete) operations with robust transaction management.

2.Objective:

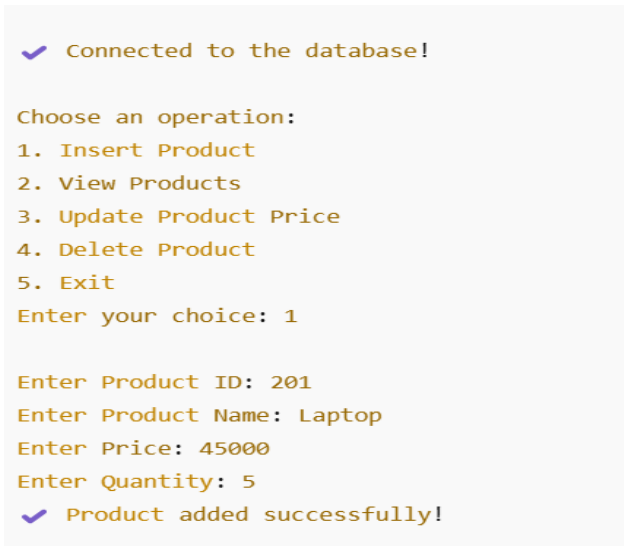
- Implement CRUD operations to handle product data effectively.
- Utilize SQL queries to insert, retrieve, update, and delete records in the Product table.
- Apply transaction handling mechanisms (commit & rollback) to maintain data consistency and integrity.
- Provide a user-friendly menu-driven interface for managing product records dynamically.

3.Code:

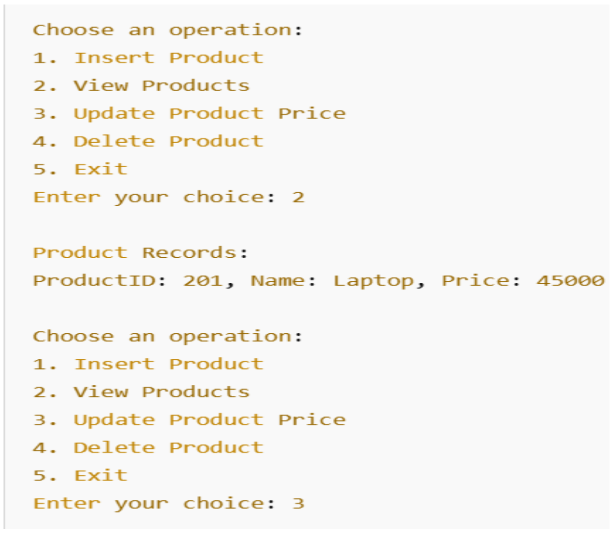
```
import java.sql.*;
import java.util.Scanner;

public class ProductDatabase {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/product";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "Shivansh@123";

    public static void main(String[] args) {
        try (Scanner input = new Scanner(System.in);
             Connection dbConn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD))
{

            Class.forName("com.mysql.cj.jdbc.Driver");
            dbConn.setAutoCommit(false);
            System.out.println("✔ Connected to the database!");

            while (true) {
                System.out.println("\nChoose an operation:");
                System.out.println("1. Insert Product");
                System.out.println("2. View Products");
                System.out.println("3. Update Product Price");
                System.out.println("4. Delete Product");
                System.out.println("5. Exit");
                System.out.print("Enter your choice: ");
                int choice = input.nextInt();

                switch (choice) {
                    case 1 -> insertProduct(dbConn, input);
                    case 2 -> viewProducts(dbConn);
                case 3 -> updatePrice(dbConn, input);
                case 4 -> deleteProduct(dbConn, input);
                case 5 -> {
                    System.out.println(" Exiting program...");
                    return;
                }
                default -> System.out.println("❌ Invalid choice. Try again!");
            }
        }
    } catch (ClassNotFoundException e) {
        System.out.println("❌ MySQL JDBC Driver Not Found!");
    } catch (SQLException e) {
        System.out.println("❌ Database Connection Error!");
    }
}

private static void insertProduct(Connection dbConn, Scanner input) throws SQLException {
    try {
        System.out.print("Enter Product ID: ");
        int id = input.nextInt();
        input.nextLine(); // Consume newline

        System.out.print("Enter Product Name: ");
        String name = input.nextLine();

        System.out.print("Enter Price: ");
        double price = input.nextDouble();

        System.out.print("Enter Quantity: ");
        int qty = input.nextInt();

        String sql = "INSERT INTO Product (ProductID, ProductName, Price, Quantity) VALUES (?, ?,
?, ?)";
        try (PreparedStatement stmt = dbConn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            stmt.setString(2, name);
            stmt.setDouble(3, price);
            stmt.setInt(4, qty);
            stmt.executeUpdate();
            dbConn.commit();
            System.out.println("✔ Product added successfully!");
        }
    } catch (SQLException e) {
        dbConn.rollback();
        System.out.println("❌ Error inserting product. Transaction rolled back!");
    }
}
```

```
    private static void viewProducts(Connection dbConn) throws SQLException {
        String sql = "SELECT * FROM Product";
        try (Statement stmt = dbConn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            System.out.println("\nProduct Records:");
            while (rs.next()) {
                System.out.printf("ProductID: %d, Name: %s, Price: %.2f, Quantity: %d%n",
                        rs.getInt("ProductID"), rs.getString("ProductName"), rs.getDouble("Price"),
rs.getInt("Quantity"));
            }
        }
    }

    private static void updatePrice(Connection dbConn, Scanner input) throws SQLException {
        try {
            System.out.print("Enter Product ID to Update: ");
            int id = input.nextInt();

            System.out.print("Enter New Price: ");
            double price = input.nextDouble();

            String sql = "UPDATE Product SET Price = ? WHERE ProductID = ?";
            try (PreparedStatement stmt = dbConn.prepareStatement(sql)) {
                stmt.setDouble(1, price);
                stmt.setInt(2, id);
                int rows = stmt.executeUpdate();
                dbConn.commit();
                System.out.println(rows > 0 ? "✔ Price updated successfully!" : "❌ Product not found!");
            }
        } catch (SQLException e) {
            dbConn.rollback();
            System.out.println("❌ Error updating product. Transaction rolled back!");
        }
    }

    private static void deleteProduct(Connection dbConn, Scanner input) throws SQLException {
        try {
            System.out.print("Enter Product ID to Delete: ");
            int id = input.nextInt();

            String sql = "DELETE FROM Product WHERE ProductID = ?";
            try (PreparedStatement stmt = dbConn.prepareStatement(sql)) {
                stmt.setInt(1, id);
                int rows = stmt.executeUpdate();
                dbConn.commit();
                System.out.println(rows > 0 ? "✔ Product deleted successfully!" : "❌ Product not found!");
            }
        } catch (SQLException e) {
            dbConn.rollback();
            System.out.println("❌ Error deleting product. Transaction rolled back!");
        }
    }}
```

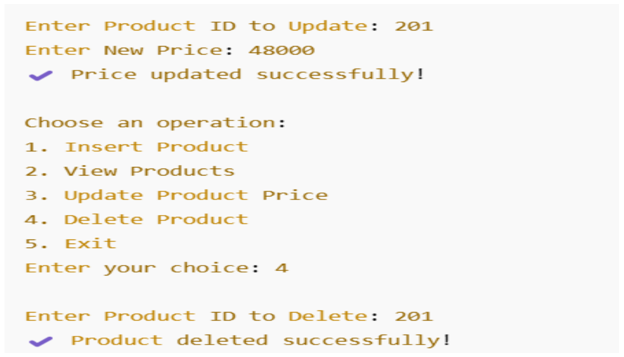
4.Output:



```
✔ Connected to the database!

Choose an operation:
1. Insert Product
2. View Products
3. Update Product Price
4. Delete Product
5. Exit
Enter your choice: 1

Enter Product ID: 201
Enter Product Name: Laptop
Enter Price: 45000
Enter Quantity: 5
✔ Product added successfully!
```



```
Choose an operation:
1. Insert Product
2. View Products
3. Update Product Price
4. Delete Product
5. Exit
Enter your choice: 2

Product Records:
ProductID: 201, Name: Laptop, Price: 45000

Choose an operation:
1. Insert Product
2. View Products
3. Update Product Price
4. Delete Product
5. Exit
Enter your choice: 3
```



```
Enter Product ID to Update: 201
Enter New Price: 48000
✔ Price updated successfully!

Choose an operation:
1. Insert Product
2. View Products
3. Update Product Price
4. Delete Product
5. Exit
Enter your choice: 4

Enter Product ID to Delete: 201
✔ Product deleted successfully!
```

.

5.Learning Outcomes:

- Implement CRUD operations (Create, Read, Update, Delete) in a Java program.
- Use JDBC to establish a connection with a MySQL database.
- Handle transactions using commit() and rollback() for data integrity.
- Interact with databases dynamically using user input and SQL queries.
- Develop robust database applications with error handling and exception management.

Problem 3

1.Aim: To develop a Java application using JDBC and MVC architecture for managing student data, enabling Create, Read, Update, and Delete (CRUD) operations through a menu-driven interface.

2.Objective:

- Understand JDBC and MVC by implementing a structured Java application with separate Model, View, and Controller components.
- Perform CRUD operations on a MySQL database containing student records.

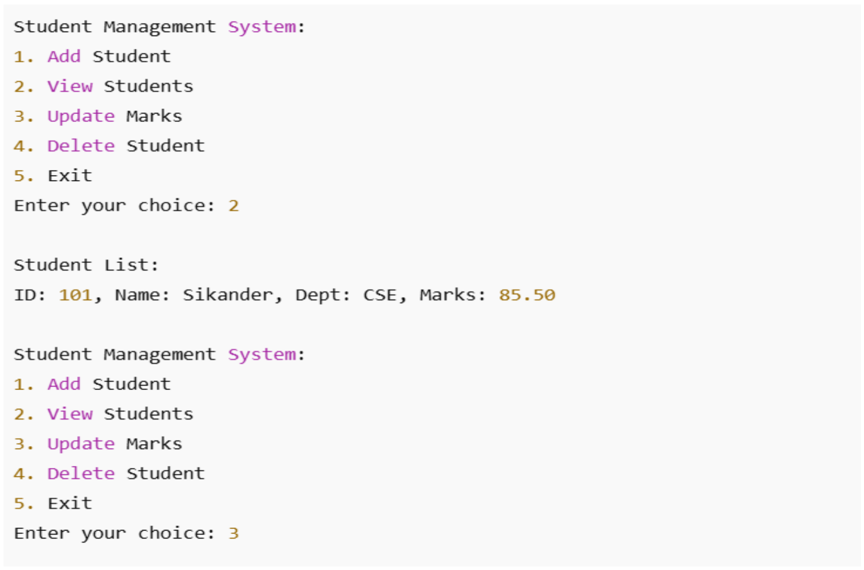
3.Code:

```
import java.sql.*;
import java.util.Scanner;

public class StudentManagementApp {
    private static final String DB_URL = "jdbc:mysql://localhost:3306/college";
    private static final String DB_USER = "root";
    private static final String DB_PASSWORD = "Shivansh@123";

    public static void main(String[] args) {
        try (Scanner input = new Scanner(System.in);
             Connection dbConn = DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD)) {

            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("✔ Connected to the database!");

            while (true) {
                System.out.println("\nStudent Management System:");
                System.out.println("1. Add Student");
                System.out.println("2. View Students");
                System.out.println("3. Update Marks");
                System.out.println("4. Delete Student");
                System.out.println("5. Exit");
                System.out.print("Enter your choice: ");
                int option = input.nextInt();

                switch (option) {
                    case 1 -> addStudent(dbConn, input);
                    case 2 -> viewStudents(dbConn);
                    case 3 -> updateMarks(dbConn, input);
                    case 4 -> deleteStudent(dbConn, input);
                    case 5 -> {
                        System.out.println(" Exiting... Thank you!");
                        return;
                    }
                    default -> System.out.println("❌ Invalid choice! Try again.");
                }
            }
        } catch (Exception e) {
            System.out.println("❌ Database Error!");
        }
    }

    private static void addStudent(Connection dbConn, Scanner input) throws SQLException {
        System.out.print("Enter Student ID: ");
        int id = input.nextInt();
        input.nextLine(); // Consume newline

        System.out.print("Enter Name: ");
        String name = input.nextLine();

        System.out.print("Enter Department: ");
        String dept = input.nextLine();

        System.out.print("Enter Marks: ");
        double marks = input.nextDouble();

        String sql = "INSERT INTO students (StudentID, Name, Department, Marks) VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = dbConn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            stmt.setString(2, name);
            stmt.setString(3, dept);
            stmt.setDouble(4, marks);
            stmt.executeUpdate();
            System.out.println("✔ Student added successfully!");
        }
    }

    private static void viewStudents(Connection dbConn) throws SQLException {
        String sql = "SELECT * FROM students";
        try (Statement stmt = dbConn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            System.out.println("\nStudent List:");
            while (rs.next()) {
                System.out.printf("ID: %d, Name: %s, Dept: %s, Marks: %.2f%n",
                        rs.getInt("StudentID"), rs.getString("Name"), rs.getString("Department"),
rs.getDouble("Marks"));
            }
        }
```

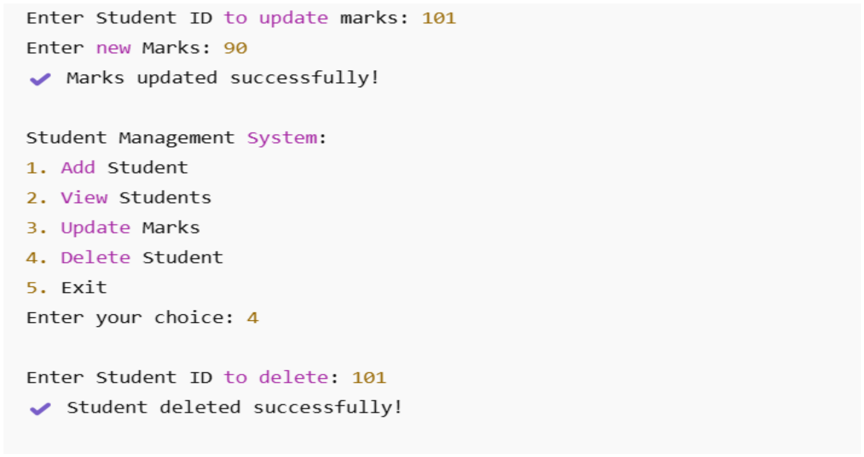
```
}

private static void updateMarks(Connection dbConn, Scanner input) throws SQLException {
    System.out.print("Enter Student ID to update marks: ");
    int id = input.nextInt();

    System.out.print("Enter new Marks: ");
    double marks = input.nextDouble();

    String sql = "UPDATE students SET Marks = ? WHERE StudentID = ?";
    try (PreparedStatement stmt = dbConn.prepareStatement(sql)) {
        stmt.setDouble(1, marks);
        stmt.setInt(2, id);
        int rows = stmt.executeUpdate();
        System.out.println(rows > 0 ? "✔ Marks updated successfully!" : "❌ Student not found!");
    }
}

private static void deleteStudent(Connection dbConn, Scanner input) throws SQLException {
    System.out.print("Enter Student ID to delete: ");
    int id = input.nextInt();

    String sql = "DELETE FROM students WHERE StudentID = ?";
    try (PreparedStatement stmt = dbConn.prepareStatement(sql)) {
        stmt.setInt(1, id);
        int rows = stmt.executeUpdate();
        System.out.println(rows > 0 ? "✔ Student deleted successfully!" : "❌ Student not found!");
    }
}
```

4.OUTPUT:

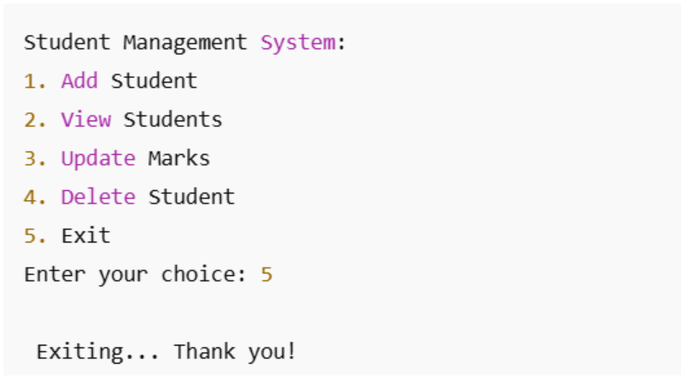
```
✔ Connected to the database!

Student Management System:
1. Add Student
2. View Students
3. Update Marks
4. Delete Student
5. Exit
Enter your choice: 1

Enter Student ID: 101
Enter Name: Sikander
Enter Department: CSE
Enter Marks: 85.5
✔ Student added successfully!
```



```
Student Management System:
1. Add Student
2. View Students
3. Update Marks
4. Delete Student
5. Exit
Enter your choice: 2

Student List:
ID: 101, Name: Sikander, Dept: CSE, Marks: 85.50

Student Management System:
1. Add Student
2. View Students
3. Update Marks
4. Delete Student
5. Exit
Enter your choice: 3
```



```
Enter Student ID to update marks: 101
Enter new Marks: 90
✔ Marks updated successfully!

Student Management System:
1. Add Student
2. View Students
3. Update Marks
4. Delete Student
5. Exit
Enter your choice: 4

Enter Student ID to delete: 101
✔ Student deleted successfully!
```



```
Student Management System:
1. Add Student
2. View Students
3. Update Marks
4. Delete Student
5. Exit
Enter your choice: 5

 Exiting... Thank you!
```

5.Learning Outcomes:

- Establish a secure connection between a Java application and a MySQL database using JDBC.
- Implement advanced CRUD (Create, Read, Update, Delete) operations efficiently in Java.
- Apply MVC architecture to separate concerns between data management, business logic, and user interaction.
- Handle SQL transactions using commit and rollback to maintain data integrity.
- Design and manage database tables with appropriate constraints like primary keys and data types.
- Develop a menu-driven application that interacts dynamically with the database.