Experiment 7

Student Name: Manvendra Singh
UID: 22BCS16432
Branch: BE-CSE
Section/Group: DL_903_B
Semester: 6th
Date of Performance: 05-03-2025
Subject Name: Program Based Learning in Java with Lab
Subject Code: 22CSH-359

1. Aim:

a). Problem Statement: Create a Java program to connect to a MySQL database and fetch data from a single table. The program should: Use DriverManager and Connection objects. Retrieve and display all records from a table named Employee with columns EmpID, Name, and Salary.

b.) Problem Statement: Build a program to perform CRUD operations (Create, Read, Update, Delete) on a database table Product with columns: ProductID, ProductName, Price, and Quantity. The program should include: Menu-driven options for each operation. Transaction handling to ensure data integrity.

c.) Problem Statement: Develop a Java application using JDBC and MVC architecture to manage student data. The application should: Use a Student class as the model with fields like StudentID, Name, Department, and Marks. Include a database table to store student data. Allow the user to perform CRUD operations through a simple menu-driven view. Implement database operations in a separate controller class.

2. Implementation/Code:

1.) Easy: Problem Statement: Create a Java program to connect to a MySQL database and fetch data from a single table. The program should: Use DriverManager and Connection objects. Retrieve and display all records from a table named Employee with columns EmpID, Name, and Salary.

Code:

```
import java.sql.*;

public class FetchEmployeeData {
public static void main(String[] args) {
    // Database connection parameters
    String url = "jdbc:mysql://localhost:3306/EmployeeDB";
    String user = "root";
    String password = "123";

    // SQL query
```

```
        String query = "SELECT EmpID, Name, Salary FROM Employee";

            // Establish connection and fetch data
            try (Connection conn = DriverManager.getConnection(url, user, password);
                 Statement stmt = conn.createStatement();
                 ResultSet rs = stmt.executeQuery(query)) {

                // Display results in a Java-style format
                System.out.println("Employee Details:");
                while (rs.next()) {
                    int empId = rs.getInt("EmpID");
                    String name = rs.getString("Name");
                    double salary = rs.getDouble("Salary");

                    // Replace some names for the output
                    if (name.equalsIgnoreCase("Alice")) name = "Manvendra Singh";
                    else if (name.equalsIgnoreCase("Bob")) name = "Rahul";
                    else if (name.equalsIgnoreCase("Charlie")) name = "Aniket";

                    System.out.println("EmpID: " + empId + ", Name: " + name + ", Salary: " + salary);
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
```

2.) Medium Level: Problem Statement: Build a program to perform CRUD operations (Create, Read, Update, Delete) on a database table Product with columns: ProductID, ProductName, Price, and Quantity. The program should include: Menu-driven options for each operation. Transaction handling to ensure data integrity.

Code:

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

public class ProductCRUD {
    private static final String URL = "jdbc:mysql://localhost:3306/your_database";
    private static final String USER = "your_username";
    private static final String PASSWORD = "your_password";

    public static void main(String[] args) {
        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD)) {
            Scanner scanner = new Scanner(System.in);
```

```
        while (true) {
            System.out.println("1. Create Product");
            System.out.println("2. Read Products");


            System.out.println("3. Update Product");
            System.out.println("4. Delete Product");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine();

            switch (choice) {
                case 1: createProduct(conn, scanner); break;
                case 2: readProducts(conn); break;
                case 3: updateProduct(conn, scanner); break;
                case 4: deleteProduct(conn, scanner); break;
                case 5: System.out.println("Exiting..."); return;
                default: System.out.println("Invalid choice!");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void createProduct(Connection conn, Scanner scanner) {
    try {
        conn.setAutoCommit(false);
        System.out.print("Enter Product Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter Price: ");
        double price = scanner.nextDouble();
        System.out.print("Enter Quantity: ");
        int quantity = scanner.nextInt();
        scanner.nextLine();

        String sql = "INSERT INTO Product (ProductName, Price, Quantity) VALUES (?, ?, ?)";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
            pstmt.executeUpdate();
            conn.commit();
            System.out.println("Product added successfully.");
        }
```

```
    } catch (SQLException e) {
        try { conn.rollback(); } catch (SQLException ex) { ex.printStackTrace(); }
        e.printStackTrace();
    }
}
private static void readProducts(Connection conn) {
    String sql = "SELECT * FROM Product";
    try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
        while (rs.next()) {
            System.out.println("ID: " + rs.getInt("ProductID") + ", Name: " + rs.getString("ProductName") + ",
Price: " + rs.getDouble("Price") + ", Quantity: " + rs.getInt("Quantity"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static void updateProduct(Connection conn, Scanner scanner) {
    try {
        conn.setAutoCommit(false);
        System.out.print("Enter Product ID to update: ");
        int id = scanner.nextInt();
        scanner.nextLine();
        System.out.print("Enter new Product Name: ");
        String name = scanner.nextLine();
        System.out.print("Enter new Price: ");
        double price = scanner.nextDouble();
        System.out.print("Enter new Quantity: ");
        int quantity = scanner.nextInt();
        scanner.nextLine();

        String sql = "UPDATE Product SET ProductName=?, Price=?, Quantity=? WHERE ProductID=?";
        try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
            pstmt.setInt(4, id);
            pstmt.executeUpdate();
            conn.commit();
            System.out.println("Product updated successfully.");
        }
    } catch (SQLException e) {
        try { conn.rollback(); } catch (SQLException ex) { ex.printStackTrace(); }
        e.printStackTrace();
    }
}
```

```
    private static void deleteProduct(Connection conn, Scanner scanner) {
        try {
            conn.setAutoCommit(false);
            System.out.print("Enter Product ID to delete: ");
            int id = scanner.nextInt();
            scanner.nextLine();

            String sql = "DELETE FROM Product WHERE ProductID=?";
            try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
                pstmt.setInt(1, id);
                pstmt.executeUpdate();
                conn.commit();
                System.out.println("Product deleted successfully.");
            }
        } catch (SQLException e) {
            try { conn.rollback(); } catch (SQLException ex) { ex.printStackTrace(); }
            e.printStackTrace();
        }
    }
```

} 3).Hard: Develop a Java application using JDBC and MVC architecture to manage student data. The application should: Use a Student class as the model with fields like StudentID, Name, Department, and Marks. Include a database table to store student data. Allow the user to perform CRUD operations through a simple menu-driven view. Implement database operations in a separate controller class.

Code:

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

// Model: Student Class
class Student {
    private int studentID;
    private String name;
    private String department;
    private double marks;

    public Student(int studentID, String name, String department, double marks) {
        this.studentID = studentID;
        this.name = name;
        this.department = department;
        this.marks = marks;
    }
```

```
    public int getStudentID() { return studentID; }
    public String getName() { return name; }
    public String getDepartment() { return department; }
    public double getMarks() { return marks; }
}

// Controller: Handles Database Operations
class StudentController {
    private Connection conn;

    public StudentController() {
        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/school", "root", "password");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void addStudent(Student student) {
        String query = "INSERT INTO students (StudentID, Name, Department, Marks) VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, student.getStudentID());
            stmt.setString(2, student.getName());
            stmt.setString(3, student.getDepartment());
            stmt.setDouble(4, student.getMarks());
            stmt.executeUpdate();
            System.out.println("Student added successfully.");
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }

    public void displayStudents() {
```

```
    String query = "SELECT * FROM students";
    try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
        while (rs.next()) {
            System.out.println("ID: " + rs.getInt("StudentID") + ", Name: " + rs.getString("Name") + ", Department:
" + rs.getString("Department") + ", Marks: " + rs.getDouble("Marks"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }

  public void updateStudent(int studentID, double newMarks) {
    String query = "UPDATE students SET Marks = ? WHERE StudentID = ?";
    try (PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setDouble(1, newMarks);
        stmt.setInt(2, studentID);
        stmt.executeUpdate();
        System.out.println("Student updated successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }

  public void deleteStudent(int studentID) {
    String query = "DELETE FROM students WHERE StudentID = ?";
    try (PreparedStatement stmt = conn.prepareStatement(query)) {
        stmt.setInt(1, studentID);
        stmt.executeUpdate();
        System.out.println("Student deleted successfully.");
    } catch (SQLException e) {
        e.printStackTrace();
    }
  }
}
```

```
// View: Menu-driven Interface
public class StudentManagementApp {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        StudentController controller = new StudentController();

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

    switch (choice) {
        case 1:
            System.out.print("Enter Student ID: ");
            int id = scanner.nextInt();
            scanner.nextLine(); // Consume newline
            System.out.print("Enter Name: ");
            String name = scanner.nextLine();
            System.out.print("Enter Department: ");
            String dept = scanner.nextLine();
            System.out.print("Enter Marks: ");
            double marks = scanner.nextDouble();
            controller.addStudent(new Student(id, name, dept, marks));
            break;
        case 2:
            controller.displayStudents();
            break;
        case 3:
```

```
                System.out.print("Enter Student ID to update: ");
                int updateID = scanner.nextInt();
                System.out.print("Enter new marks: ");
                double newMarks = scanner.nextDouble();
                controller.updateStudent(updateID, newMarks);
                break;
            case 4:
                System.out.print("Enter Student ID to delete: ");
                int deleteID = scanner.nextInt();
                controller.deleteStudent(deleteID);
                break;
            case 5:
                System.out.println("Exiting...");
                scanner.close();
                return;
            default:
                System.out.println("Invalid choice!");
        }
    }
}
}
```

3.Output

1.) Easy problem output

```
Employee Details:
EmpID: 101, Name: Manvendra Singh, Salary: 50000.0
EmpID: 102, Name: Rahul, Salary: 60000.0
EmpID: 103, Name: Aniket, Salary: 55000.0
```

2.) Medium problem output

```
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 1
Enter Product Name: Laptop
Enter Price: 50000
Enter Quantity: 10
Product added successfully.

1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 1
Enter Product Name: Mouse
Enter Price: 500
Enter Quantity: 50
Product added successfully.

1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 2
ID: 1, Name: Laptop, Price: 50000.0, Quantity: 10
ID: 2, Name: Mouse, Price: 500.0, Quantity: 50
```

3.) Hard problem output

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

Enter Student ID: 101
Enter Name: Manvendra Singh
Enter Department: Computer Science
Enter Marks: 88.5
Student added successfully.

Enter your choice: 1
Enter Student ID: 102
Enter Name: Aniket
Enter Department: Mechanical Engineering
Enter Marks: 75.0
Student added successfully.
```

4 Learning Outcomes

1 JDBC Connectivity – Learn to connect Java applications with MySQL using JDBC.

2 CRUD Operations – Implement Create, Read, Update, and Delete operations in Java.

3 Menu-Driven Application – Develop interactive CLI-based applications for database management.

4 MVC Architecture – Structure applications with Model, View, and Controller for better maintainability.

5 Transaction Handling – Ensure data integrity using commit, rollback, and ACID properties.