Experiment 7

Student Name: Kritika Sharma **UID:** 22BCS14943

Branch: CSE **Section/Group:** 22CSE_KRG_IOT-3B

Semester: 6 **Date of Performance:** 18/03/2025

Subject Name: Project Based Learning on Java **Subject Code:** 22CSH-352

1. **Aim:**
   - 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

2. **Objective:**
   - To make a connection to the database
   - To fetch data from the database

3. **Implementation/Code:**

```
import java.sql.*;

public class FetchEmployeeData { public
   static void main(String[] args) {
      String url = "jdbc:mysql://localhost:3306/CompanyDB"; // Change to your database
      String user = "root"; // Your MySQL username
      String password = "root"; // Your MySQL password

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

      try {
         // Load MySQL JDBC Driver
         Class.forName("com.mysql.cj.jdbc.Driver");

         // Establish connection
         Connection conn = DriverManager.getConnection(url, user, password);

         // Create a statement
         Statement stmt = conn.createStatement();

         // Execute query
         ResultSet rs = stmt.executeQuery(query);
```

```
            // Process results while (rs.next()) { int
            empID = rs.getInt("EmpID"); String
            name = rs.getString("Name"); double
            salary = rs.getDouble("Salary");
              System.out.println("EmpID: " + empID + ", Name: " + name + ", Salary: " + salary);
            }

            // Close resources
            rs.close();
            stmt.close();
            conn.close();
          } catch (Exception e) {
            e.printStackTrace();
          }
        }
}
```

5. Output:

```
c:\Users\SHUVAM\Desktop\For vs> c: && cd "c:\Users\SHUVAM\Desktop\For vs" && cmd
ion-pack-jdk\java\21\bin\java.exe @C:\Users\SHUVAM\AppData\Local\Temp\cp_4x8irdb
EmpID: 1, Name: Shuvam, Salary: 50000.0
EmpID: 2, Name: Manik, Salary: 40000.0
EmpID: 3, Name: Manav, Salary: 30000.0
```

6. Learning Outcome:

- Learn how to use Java's connection to the database
- Gain Understanding of how to fetch data
- Knowledge of using sql

Problem 2

1. Aim:

- 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

2. Objective:

- to perform CRUD operations
- Understand how to create rea update and delete
- Display results in a structured and readable format

3. Implementation/Code:

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

public class ProductCRUD { private static final String URL =
   "jdbc:mysql://localhost:3306/ProductDB"; private static final String USER
   = "root"; // Change if needed private static final String PASSWORD = "";
   // Set your MySQL password
public static void main(String[] args) throws SQLException { try (Connection conn =
   DriverManager.getConnection(URL, USER, PASSWORD); Scanner scanner = new
   Scanner(System.in)) {
while (true) {
                System.out.println("\n1. Add 2. View 3. Update 4. Delete 5. Exit");
                switch (scanner.nextInt()) { case 1 -> addProduct(conn, scanner); case
                2 -> viewProducts(conn); case 3 -> updateProduct(conn, scanner);
                case 4 -> deleteProduct(conn, scanner); case 5 -> System.exit(0);
                }
            }
        }
    }
    private static void addProduct(Connection conn, Scanner scanner) throws SQLException {
        System.out.print("Name: "); scanner.nextLine();
        String name = scanner.nextLine();
        System.out.print("Price: "); double price = scanner.nextDouble();
        System.out.print("Quantity: "); int quantity = scanner.nextInt();

        try (PreparedStatement stmt = conn.prepareStatement("INSERT INTO Product
    (ProductName, Price, Quantity) VALUES (?, ?, ?)")) {
            stmt.setString(1, name); stmt.setDouble(2,
            price); stmt.setInt(3, quantity);
            stmt.executeUpdate();
            System.out.println("Product added!");
        }
    }
private static void viewProducts(Connection conn) throws SQLException { try (Statement stmt =
    conn.createStatement(); ResultSet rs = stmt.executeQuery("SELECT
* FROM Product")) { while (rs.next()) System.out.println(rs.getInt(1) + " | " +
rs.getString(2) + " | $" + rs.getDouble(3) + " | " + rs.getInt(4)); }
    }
```

```
    private static void updateProduct(Connection conn, Scanner scanner) throws SQLException
{
        System.out.print("ID to update: "); int id = scanner.nextInt();
        System.out.print("New Price: "); double price = scanner.nextDouble();
        System.out.print("New Quantity: "); int quantity = scanner.nextInt();

        try (PreparedStatement stmt = conn.prepareStatement("UPDATE Product SET Price=?,
Quantity=? WHERE ProductID=?")) {
            stmt.setDouble(1, price);
            stmt.setInt(2, quantity);
            stmt.setInt(3, id);
            System.out.println(stmt.executeUpdate() > 0 ? "Updated!" : "Not found!");
        }
    }

    private static void deleteProduct(Connection conn, Scanner scanner) throws SQLException
        { System.out.print("ID to delete: "); int id = scanner.nextInt();
        try (PreparedStatement stmt = conn.prepareStatement("DELETE FROM Product WHERE
ProductID=?")) { stmt.setInt(1,
            id);
            System.out.println(stmt.executeUpdate() > 0 ? "Deleted!" : "Not found!");
        }
    }
}
```

4. Output:

```
1. Add
2. View
3. Update
4. Delete
5. Exit
1
Name: watch
Price: 10000
Quantity: 3
Product added!

1. Add
2. View
3. Update
4. Delete
5. Exit
3
ID to update: 3
New Price: 15000
New Quantity: 3
Updated!

1. Add
2. View
3. Update
4. Delete
5. Exit
2
3 | watch | $15000.0 | 3

1. Add
2. View
3. Update
4. Delete
5. Exit
4
ID to delete: 3
Deleted!
```

--

2. Learning Outcome:

a. Understanding of SQL

b. Ability to apply CRUD operations

c. improved understanding the connection to the SQL

Problem 3

1. Aim:

a. Write 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. Objective:

- Learn how to use MVC structure
- Use crud operations
- Implement the database

3. Implementation/Code:

Student.java

```
import java.io.*;
public class Student {
private int id; private String
name; private String
department; private double
marks;

  public Student(int var1, String var2, String var3, double var4) {
   this.id = var1;
   this.name = var2;
   this.department = var3;
   this.marks = var4;
  }
public int getId() {
   return this.id;
  }

  public String getName() {
   return this.name;
  }

  public String getDepartment() {
   return this.department;
  }

  public double getMarks() {
   return this.marks;
  }
 }
```

StudentController.java

```
import
java.sql.Connection;
import
java.sql.DriverManager;
import
java.sql.PreparedStatement;
import java.sql.ResultSet; import
```

```
java.sql.SQLException;     import
java.sql.Statement;        import
java.util.ArrayList;       import
java.util.List;

public class StudentController implements AutoCloseable { //  Implements
AutoCloseable  private  static  final  String  URL  =
   "jdbc:mysql://localhost:3306/CollegeDB"; private static final String USER
   = "root"; private static final String PASSWORD = "root"; private
   Connection conn;

   public StudentController() throws SQLException { conn =
      DriverManager.getConnection(URL, USER, PASSWORD);
   }
   public void addStudent(Student student) throws SQLException {
     String sql = "INSERT INTO Students (StudentID, Name, Department, Marks) VALUES
(?,   ?,   ?,   ?)";   try   (PreparedStatement   stmt   =
      conn.prepareStatement(sql))  {  stmt.setInt(1,
      student.getId());   stmt.setString(2,   student.getName());
      stmt.setString(3,                 student.getDepartment());
      stmt.setDouble(4,                    student.getMarks());
      stmt.executeUpdate();
         System.out.println("Student Added!"); }
   }
   public List<Student> getAllStudents() throws SQLException {
      List<Student> students = new ArrayList<>(); String
      sql = "SELECT * FROM Students";
      try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
         while (rs.next()) {
            students.add(new Student(rs.getInt(1), rs.getString(2), rs.getString(3),
rs.getDouble(4)));
         }
      }
      return students;
   }
   public void updateStudent(int id, double marks) throws SQLException { String
      sql = "UPDATE Students SET Marks=? WHERE StudentID=?";
      try (PreparedStatement stmt = conn.prepareStatement(sql)) {
         stmt.setDouble(1, marks); stmt.setInt(2, id);
         System.out.println(stmt.executeUpdate() > 0 ? "Updated!" : "Student Not Found!");
      }
   }
   public void deleteStudent(int id) throws SQLException { String
      sql = "DELETE FROM Students WHERE StudentID=?"; try
```

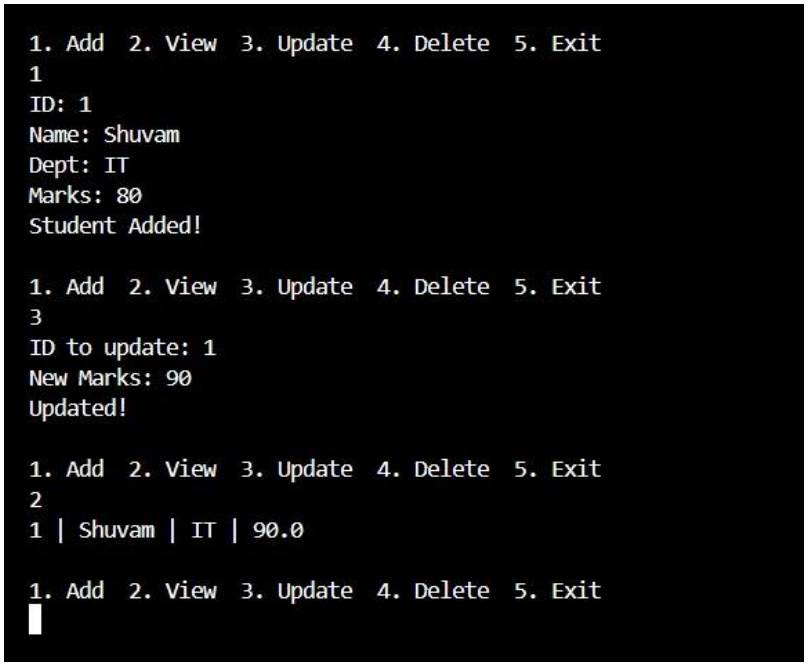
```
        (PreparedStatement stmt = conn.prepareStatement(sql)) {
        stmt.setInt(1, id);
          System.out.println(stmt.executeUpdate() > 0 ? "Deleted!" : "Student Not Found!");
        }
    }
    @Override
    public void close() throws SQLException { // Implements AutoCloseable to close
connection if (conn != null)
   conn.close(); }
}
```

StudentView.java import
java.sql.SQLException;
import java.util.Scanner;

```
public class StudentView { public static void main(String[] args) { try (Scanner scanner
   = new Scanner(System.in); StudentController controller = new
StudentController()) { while
         (true) {
            System.out.println("\n1. Add 2. View 3. Update 4. Delete 5. Exit");
            switch (scanner.nextInt()) { case 1 -> {
                  System.out.print("ID: "); int id = scanner.nextInt();
                  System.out.print("Name: "); scanner.nextLine(); String name =
scanner.nextLine();
                  System.out.print("Dept: "); String dept = scanner.nextLine();
                  System.out.print("Marks: "); double marks = scanner.nextDouble();
                  controller.addStudent(new Student(id, name, dept, marks));
               }
               case 2 -> controller.getAllStudents().forEach(s ->
                  System.out.println(s.getId() + " | " + s.getName() + " | " + s.getDepartment() +
" | " + s.getMarks())); case
               3 -> {
                  System.out.print("ID to update: "); int id = scanner.nextInt();
                  System.out.print("New Marks: "); double marks = scanner.nextDouble();
                  controller.updateStudent(id, marks);
               }
               case 4 -> {
                  System.out.print("ID to delete: "); int id = scanner.nextInt();
                  controller.deleteStudent(id);
               }
               case 5 -> System.exit(0);
            } }
      } catch (SQLException e) {
         e.printStackTrace();
      }
```

```
        }
    }
```

4. **Output:**


1. Add 2. View 3. Update 4. Delete 5. Exit
1
ID: 1
Name: Shuvam
Dept: IT
Marks: 80
Student Added!
1. Add 2. View 3. Update 4. Delete 5. Exit
3
ID to update: 1
New Marks: 90
Updated!
1. Add 2. View 3. Update 4. Delete 5. Exit
2
1 | Shuvam | IT | 90.0
1. Add 2. View 3. Update 4. Delete 5. Exit

5. Learning Outcome:

- Understanding the use of sql
- Ability to use MVC structure
- Practical experience fetching data from the database