Experiment-7

Student Name: Palak Khullar **UID:** 22BCS13306
Branch: BE-CSE **Section/Group:** 618-B
Semester: 6th **Date of Performance:** 21/03/25
Subject Name: Project Based Learning in Java **Subject Code:** 22CSH-359

EASY:

Aim: Create a Java program to connect to a MySQL database and fetch data from a single table. The program should:

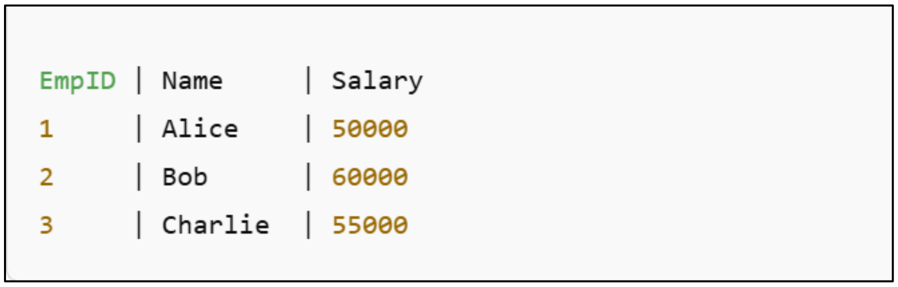
- Use DriverManager and Connection objects.
- Retrieve and display all records from a table named *Employee* with columns *EmpID*, *Name*, and *Salary*.

Objective: To establish a connection to a MySQL database using JDBC and retrieve all records from the *Employee* table. The program demonstrates the use of *DriverManager* and *Connection* objects.

Implementation/Code:

```
import java.sql.*;


public class EasyLevelJDBC {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/your_database";
        String user = "root";
        String password = "password";


        try (Connection conn = DriverManager.getConnection(url, user, password)) {
            System.out.println("Connected to the database!");

            String query = "SELECT EmpID, Name, Salary FROM Employee";
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery(query);

            while (rs.next()) {
                int empID = rs.getInt("EmpID");
                String name = rs.getString("Name");
                double salary = rs.getDouble("Salary");

                System.out.println(empID + " | " + name + " | " + salary);
            }

        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Output



```
EmpID | Name      | Salary
1     | Alice     | 50000
2     | Bob       | 60000
3     | Charlie   | 55000
```

MEDIUM:

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.

Objective: To implement CRUD operations on a *Product* table with transaction handling to maintain data integrity. The program provides a menu-driven interface for creating, reading, updating, and deleting product records.

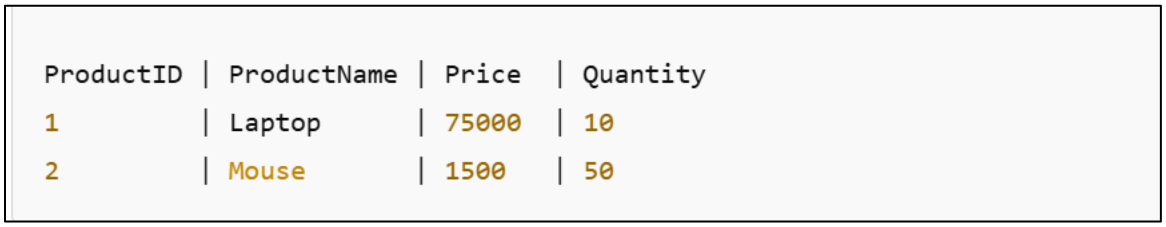
Code/Implementation:

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

public class MediumLevelJDBC {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/your_database";
        String user = "root";
    String password = "password";

    try (Connection conn = DriverManager.getConnection(url, user, password)) {
        System.out.println("Connected to the database!");
        Scanner scanner = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. 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();

            switch (choice) {
                case 1 -> {
                    System.out.print("Enter Product Name: ");
                    String name = scanner.next();
                    System.out.print("Enter Price: ");
                    double price = scanner.nextDouble();
                    System.out.print("Enter Quantity: ");
                    int quantity = scanner.nextInt();

                    String insertQuery = "INSERT INTO Product (ProductName, Price, Quantity)
VALUES (?, ?, ?)";
                    PreparedStatement pstmt = conn.prepareStatement(insertQuery);
                    pstmt.setString(1, name);
                    pstmt.setDouble(2, price);
                    pstmt.setInt(3, quantity);
                    pstmt.executeUpdate();
                    System.out.println("Product Added!");
                }
                case 2 -> {
                    String selectQuery = "SELECT * FROM Product";
                    Statement stmt = conn.createStatement();
                    ResultSet rs = stmt.executeQuery(selectQuery);
                    System.out.println("ProductID | ProductName | Price | Quantity");
                    while (rs.next()) {
                        System.out.println(rs.getInt("ProductID") + " | " +
                                           rs.getString("ProductName") + " | " +
                                           rs.getDouble("Price") + " | " +
                                           rs.getInt("Quantity"));
                    }
                }
                case 3 -> {
                    System.out.print("Enter ProductID to Update: ");
                    int id = scanner.nextInt();
                    System.out.print("Enter New Price: ");
                    double price = scanner.nextDouble();

                    String updateQuery = "UPDATE Product SET Price = ? WHERE ProductID = ?";
                    PreparedStatement pstmt = conn.prepareStatement(updateQuery);
```

```
                pstmt.setDouble(1, price);
                pstmt.setInt(2, id);
                int updated = pstmt.executeUpdate();
                System.out.println(updated > 0 ? "Product Updated!" : "Product Not Found!");
            }
            case 4 -> {
                System.out.print("Enter ProductID to Delete: ");
                int id = scanner.nextInt();
                String deleteQuery = "DELETE FROM Product WHERE ProductID = ?";
                PreparedStatement pstmt = conn.prepareStatement(deleteQuery);
                pstmt.setInt(1, id);
                int deleted = pstmt.executeUpdate();
                System.out.println(deleted > 0 ? "Product Deleted!" : "Product Not Found!");
            }
            case 5 -> {
                System.out.println("Exiting...");
                return;
            }
            default -> System.out.println("Invalid Option!");
        }
    }
  } catch (SQLException e) {
      System.out.println("SQL Exception: " + e.getMessage());
      e.printStackTrace();
  }
}
```

}

Output:



```
ProductID | ProductName | Price  | Quantity
1         | Laptop      | 75000  | 10
2         | Mouse       | 1500   | 50
```

<u>HARD:</u>

Aim: 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.

Objective: To build a Java application using JDBC and MVC architecture for managing *Student* data. The application separates data handling into a controller class and provides a menu-driven interface for CRUD operations.

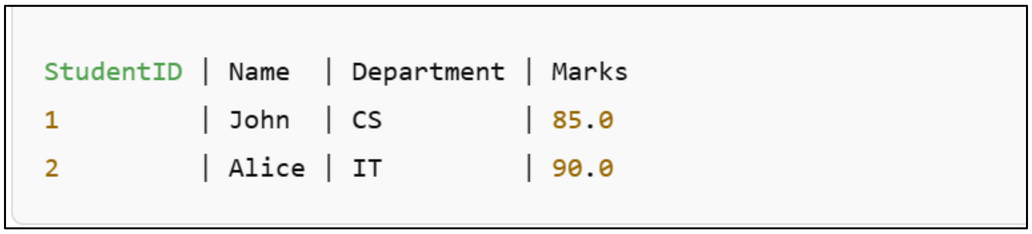
Code/Implementation:

```
import java.sql.*;

import java.util.Scanner;


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; }
}

class StudentController {
    private Connection conn;

    public StudentController() throws SQLException {
        String url = "jdbc:mysql://localhost:3306/your_database";
        String user = "root";
        String password = "password";
        conn = DriverManager.getConnection(url, user, password);
    }

    public void addStudent(Student student) throws SQLException {
        String query = "INSERT INTO Student (Name, Department, Marks)
VALUES (?, ?, ?)";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setString(1, student.getName());
        pstmt.setString(2, student.getDepartment());
        pstmt.setDouble(3, student.getMarks());
        pstmt.executeUpdate();
    }

    public void displayStudents() throws SQLException {
        String query = "SELECT * FROM Student";
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            System.out.println(rs.getInt("StudentID") + " | " +
                    rs.getString("Name") + " | " +
                    rs.getString("Department") + " | " +
                    rs.getDouble("Marks"));
        }
    }

    public void updateStudentMarks(int studentID, double newMarks) throws
SQLException {
        String query = "UPDATE Student SET Marks = ? WHERE StudentID = ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setDouble(1, newMarks);
        pstmt.setInt(2, studentID);
        pstmt.executeUpdate();
    }

    public void deleteStudent(int studentID) throws SQLException {
        String query = "DELETE FROM Student WHERE StudentID = ?";
        PreparedStatement pstmt = conn.prepareStatement(query);
        pstmt.setInt(1, studentID);
        pstmt.executeUpdate();
    }
}

public class Main {
    public static void main(String[] args) {
        try {
            StudentController controller = new StudentController();
            Scanner scanner = new Scanner(System.in);
```

```
while (true) {
    System.out.println("\n1. Add Student");
    System.out.println("2. View Students");
    System.out.println("3. Update Student Marks");
    System.out.println("4. Delete Student");
    System.out.println("5. Exit");
    System.out.print("Choose an option: ");
    int choice = scanner.nextInt();

    switch (choice) {
        case 1 -> {
            System.out.print("Enter Name: ");
            String name = scanner.next();
            System.out.print("Enter Department: ");
            String department = scanner.next();
            System.out.print("Enter Marks: ");
            double marks = scanner.nextDouble();
            Student student = new Student(0, name, department, marks);
            controller.addStudent(student);
        }
        case 2 -> controller.displayStudents();
        case 3 -> {
            System.out.print("Enter StudentID: ");
```

```
                    int id = scanner.nextInt();
                    System.out.print("Enter New Marks: ");
                    double marks = scanner.nextDouble();
                    controller.updateStudentMarks(id, marks);
                }
                case 4 -> {
                    System.out.print("Enter StudentID: ");
                    int id = scanner.nextInt();
                    controller.deleteStudent(id);
                }
                case 5 -> {
                    System.out.println("Exiting...");
                    return;
                }
                default -> System.out.println("Invalid Option");
            }
        }
    } catch (SQLException e) {
        System.out.println("SQL Exception: " + e.getMessage());
    }
  }
}
```

Output:



```
StudentID | Name  | Department | Marks
1         | John  | CS         | 85.0
2         | Alice | IT         | 90.0
```

Learning Outcome:

1. Understand and implement JDBC to connect Java applications with a MySQL database, perform CRUD operations, and handle transactions.
2. Apply the MVC architecture to separate concerns in Java applications, enhancing code organization and maintainability while managing database interactions.