EXPERIMENT- 7

Student Name: Eklavya Kumar **UID:** 22BCS13380

Branch: CSE **Section/Group:** 22BCS_IOT-639-A

Semester: 6 **Date of Performance:** 21.03.25

Subject Name: Project Based Learning in Java **Subject Code:**22CSH-359

EASY LEVEL

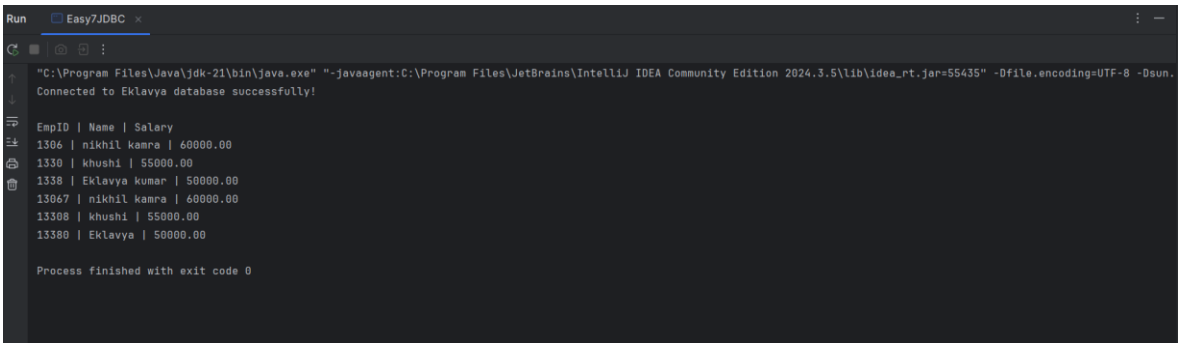
1. **Aim**: Create a Java program to connect to a MySQL database and fetch data from a single table.

2. **Objective:** To retrieve and display all records from a table named Employee with columns EmpID, Name, and Salary.

3. **Implementation/Code:**

4.
```
package Project1; import java.sql.*; public
class Easy7JDBC {
    public static void
    main(String[] args) {
        // Database connection details
        String url = "jdbc:mysql://localhost:3306/student";
        String username = "root";
        String password = "12345";
        // SQL Query
        String query = "SELECT * FROM Employee";
        try (Connection
                    conn = DriverManager.getConnection(url, username, password);
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(query)) {
            System.out.println("Connected to student database successfully!\n");
            System.out.println("EmpID | Name | Salary");
            while (rs.next())
            {
                System.out.printf("%d | %s | %.2f\n",
                        rs.getInt("EmpID"),
                        rs.getString("Name"),
                        rs.getDouble("Salary"));}
        } catch (SQLException e) {
            System.err.println("Connection failed: " + e.getMessage());
        }
    }}
```

5. **Output:**



MEDIUM LEVEL

1. **Aim**: Build a program to perform CRUD operations

2. **Objective:** To perform 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.

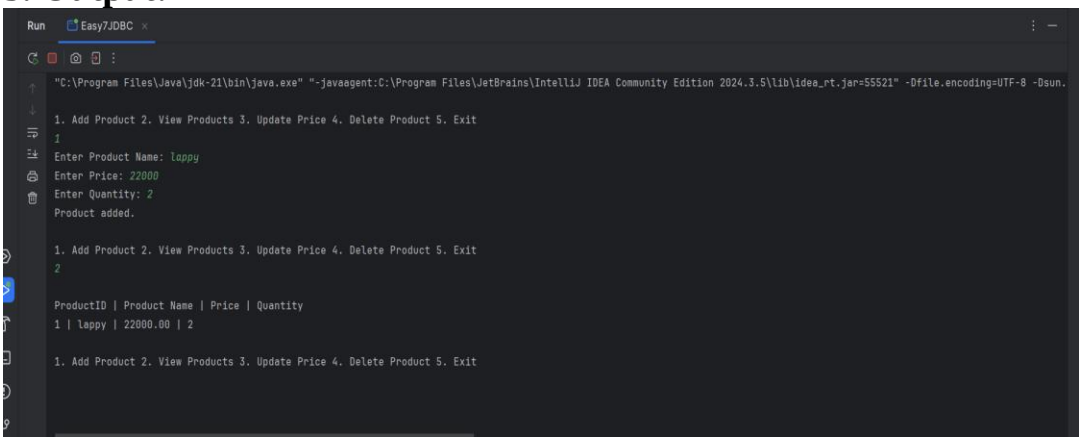
3. **Implementation/Code:**

4.
```
package Project1;

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

public class Easy7JDBC {
    public static void main(String[] args) {
        String url = "jdbc:mysql://localhost:3306/student";
        String user = "root";
        String password = "12345";
        Scanner sc = new Scanner(System.in);

        try (Connection conn = DriverManager.getConnection(url, user,
password)) {
            while (true) {
                System.out.println("\n1. Add Product 2. View Products 3. Update
Price 4. Delete Product 5. Exit");
                int choice = sc.nextInt();

                if (choice == 1) {
                    addProduct(conn, sc);
                } else if (choice == 2) {
                    viewProducts(conn);
                } else if (choice == 3) {
                    updateProduct(conn, sc);
                } else if (choice == 4) {
                    deleteProduct(conn, sc);
                } else if (choice == 5) {
                    break;
                } else {
                    System.out.println("Invalid choice.");
                }
            }
```

```
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        static void addProduct(Connection conn, Scanner sc) throws SQLException {
            System.out.print("Enter Product Name: ");
            sc.nextLine(); // Consume the newline character
            String name = sc.nextLine();

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

            System.out.print("Enter Quantity: ");
            int quantity = sc.nextInt();

            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.");
        }

        static void viewProducts(Connection conn) throws SQLException {
            ResultSet rs = conn.createStatement().executeQuery("SELECT * FROM
Product");

            System.out.println("\nProductID | Product Name | Price | Quantity");

            while (rs.next()) {
                System.out.printf("%d | %s | %.2f | %d\n", rs.getInt(1),
rs.getString(2), rs.getDouble(3), rs.getInt(4));
            }
        }

        static void updateProduct(Connection conn, Scanner sc) throws SQLException
{
            System.out.print("Enter ProductID to update: ");
            int id = sc.nextInt();

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

            PreparedStatement stmt = conn.prepareStatement("UPDATE Product SET
Price=? WHERE ProductID=?");
            stmt.setDouble(1, price);
            stmt.setInt(2, id);

            stmt.executeUpdate();
            System.out.println("Product updated.");
        }

        static void deleteProduct(Connection conn, Scanner sc) throws SQLException
{
            System.out.print("Enter ProductID to delete: ");
            int id = sc.nextInt();
```

```
            PreparedStatement stmt = conn.prepareStatement("DELETE FROM Product
    WHERE ProductID=?");
            stmt.setInt(1, id);

            stmt.executeUpdate();
            System.out.println("Product deleted.");
        }
    }
```

5. Output:



HARD LEVEL

1. **Aim**: Develop a Java application using JDBC and MVC architecture to manage student data.

2. **Objective:** To Use a Student class as the model with fields like StudentID, Name, Department, and Marks. Include a database table to store student data.

3. **Implementation/Code:**

4.
```
package Project1;

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

public class StudentView {
    public static void main(String[] args) {
        try {
            // Create a controller object to handle student operations
            StudentController controller = new StudentController();
            Scanner sc = new Scanner(System.in);

            while (true) {
                // Display menu to the user
                System.out.println("\n1. Add Student 2. View Students 3. Update
Marks 4. Delete Student 5. Exit");
                int choice = sc.nextInt();

                if (choice == 1) {
                    // Option to add a new student
                    sc.nextLine(); // Consume the newline left by nextInt()
                    System.out.print("Enter Name: ");
                    String name = sc.nextLine();
                    System.out.print("Enter Department: ");
                    String dept = sc.nextLine();
                    System.out.print("Enter Marks: ");
                    double marks = sc.nextDouble();

                    // Add the new student
                    controller.addStudent(new Student(0, name, dept, marks));
                    System.out.println("Student added.");

                } else if (choice == 2) {
                    // Option to view all students
                    List<Student> students = controller.getStudents();
                    System.out.println("\nStudentID | Name | Department |
Marks");
                    for (Student s : students) {
                        System.out.printf("%d | %s | %s | %.2f\n",
                                s.getStudentID(), s.getName(),
s.getDepartment(), s.getMarks());
                    }

                } else if (choice == 3) {
```

```
                    // Option to update marks of a student
                    System.out.print("Enter StudentID to update: ");
                    int id = sc.nextInt();
                    System.out.print("Enter new Marks: ");
                    double marks = sc.nextDouble();
                    controller.updateStudentMarks(id, marks);
                    System.out.println("Student marks updated.");

                } else if (choice == 4) {
                    // Option to delete a student
                    System.out.print("Enter StudentID to delete: ");
                    int id = sc.nextInt();
                    controller.deleteStudent(id);
                    System.out.println("Student deleted.");

                } else if (choice == 5) {
                    // Option to exit the program
                    System.out.println("Exiting...");
                    break;

                } else {
                    // Invalid choice
                    System.out.println("Invalid choice.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

5. Output:

```
StudentView
"C:\Program Files\Java\jdk-20\bin\java.exe" "-javaagent:C:\Program Files\JetBrains

1. Add Student  2. View Students  3. Update Marks  4. Delete Student  5. Exit
1
Enter Name: Shivani Singh
Enter Department: Computer Science
Enter Marks: 95.7
Student added successfully.

1. Add Student  2. View Students  3. Update Marks  4. Delete Student  5. Exit
2

StudentID | Name | Department | Marks
-------------------------------------
1 | Shivani Singh | Computer Science | 95.70

1. Add Student  2. View Students  3. Update Marks  4. Delete Student  5. Exit
5
Exiting...

Process finished with exit code 0
```

6. Learning Outcomes:

(i) Learn how to **establish a connection** between a Java application and a MySQL database using **JDBC**.

(ii) Understand the use of **DriverManager and Connection objects** to interact with the database.

(iii) Learn to use **PreparedStatement** to securely execute SQL queries.