Experiment -7

Student Name: Khyati Singh
Branch: BE-CSE
Semester: 6
Subject Name: Java with Lab

UID:22BCS16405
Section/Group:DL_901-A
Date of Performance:4/03/25
Subject Code: 22CSH-359

7.1.1 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..

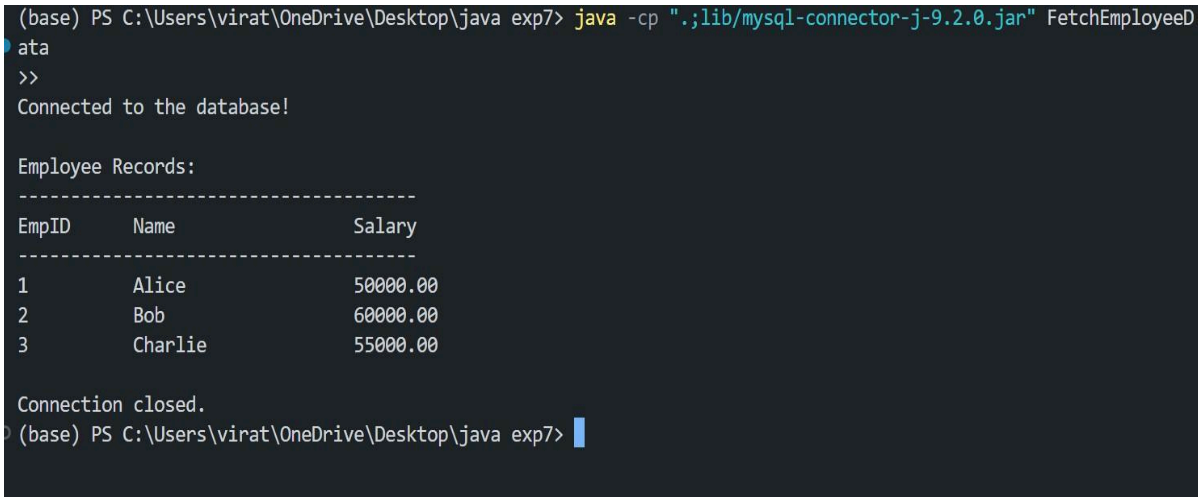
7.1.2 **Objective:** To develop a Java program that connects to a MySQL database, retrieves data from the Employee table, and displays all records, demonstrating basic JDBC connectivity and data retrieval operations.

7.1.3 Code:

```
import java.sql.*;

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

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


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

            // Establish connection

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

            // Create statement and execute query Statement
            stmt = con.createStatement(); ResultSet rs =
            stmt.executeQuery(query);

            // Display results System.out.println("\nEmployee
            Records:");
```

```
                System.out.println("   ---------------------------");
                System.out.printf("%-10s %-20s %-10s%n", "EmpID", "Name", "Salary");
                System.out.println("   ---------------------------");

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

                    System.out.printf("%-10d %-20s %-10.2f%n", empID, name, salary);
                }

                // Close resources
                rs.close();
                stmt.close();
                con.close(); System.out.println("\nConnection
                closed.");

            } catch (ClassNotFoundException e) {
                System.out.println("MySQL Driver not found: " + e.getMessage());
            } catch (SQLException e) {
                System.out.println("SQL Error: " + e.getMessage());
            }
        }
    }
```

7.1.4 Output:



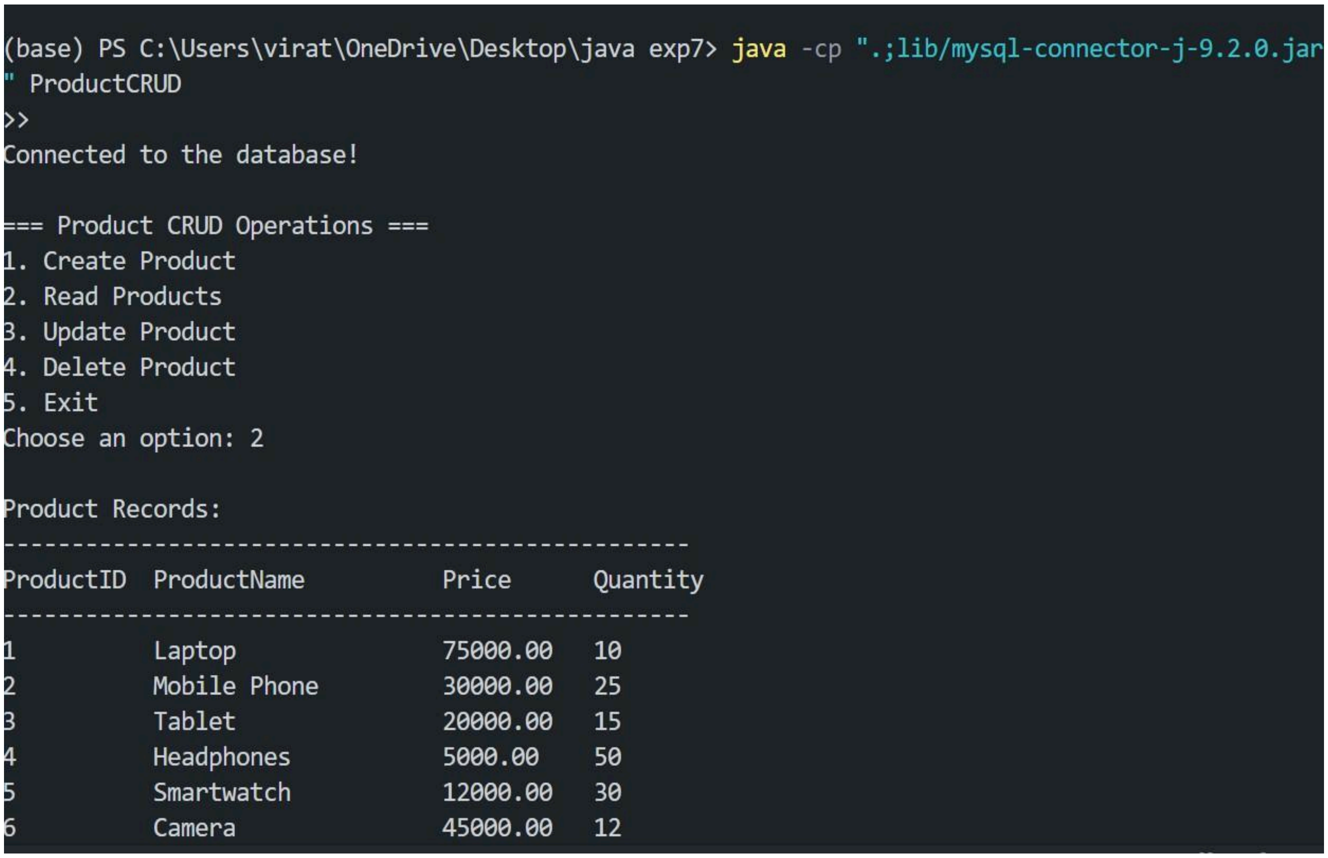
```
(base) PS C:\Users\virat\OneDrive\Desktop\java exp7> java -cp ".;lib/mysql-connector-j-9.2.0.jar" FetchEmployeeD
ata
>>
Connected to the database!

Employee Records:
--------------------------------------
EmpID      Name                 Salary
--------------------------------------
1          Alice                50000.00
2          Bob                  60000.00
3          Charlie              55000.00

Connection closed.
(base) PS C:\Users\virat\OneDrive\Desktop\java exp7>
```

7.2.1 **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.
7.2.2 **Objective**: To develop a Java program that connects to a MySQL database and performs CRUD operations (Create, Read, Update, Delete) on the Product table. The program ensures data integrity by using transaction handling and provides a menu-driven interface for user-friendly interaction.

7.2.3 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";
    private static final String PASSWORD = "password";


    public static void main(String[] args) { Scanner
        scanner = new Scanner(System.in);

        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD)) {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Connected to the database!"); boolean

            exit = false;

            while (!exit) {
                System.out.println("\n=== Product CRUD Operations ===");
                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); case
                    2 -> readProducts(conn);
                    case 3 -> updateProduct(conn, scanner); case 4
                    -> deleteProduct(conn, scanner); case 5 -> exit
                    = true;
                    default -> System.out.println("Invalid option. Try again.");
                }
            }

        } catch (ClassNotFoundException e) {
            System.out.println("MySQL Driver not found: " + e.getMessage());
        } catch (SQLException e) {
            System.out.println("SQL Error: " + e.getMessage());
        }

        scanner.close();
    }

    private static void createProduct(Connection conn, Scanner scanner) throws SQLException {
        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();

        String query = "INSERT INTO Product (ProductName, Price, Quantity) VALUES (?, ?,
?)";

        try (PreparedStatement pstmt = conn.prepareStatement(query)) { conn.setAutoCommit(false);

            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);

            int rows = pstmt.executeUpdate(); conn.commit();

            System.out.println(rows + " product(s) inserted successfully!");
        } catch (SQLException e) {
            conn.rollback();
            System.out.println("Transaction rolled back due to error: " + e.getMessage());
        } finally { conn.setAutoCommit(true);
        }
    }

    private static void readProducts(Connection conn) throws SQLException { String query =
        "SELECT * FROM Product";

        try (Statement stmt = conn.createStatement(); ResultSet rs
             = stmt.executeQuery(query)) {

            System.out.println("\nProduct Records:");
            System.out.println("                                                    ");
            System.out.printf("%-10s %-20s %-10s %-10s%n", "ProductID", "ProductName", "Price", "Quantity");
            System.out.println("    -----------------------------------------------");

            while (rs.next()) {
                int id = rs.getInt("ProductID");
                String name = rs.getString("ProductName"); double
                price = rs.getDouble("Price");
                int quantity = rs.getInt("Quantity");

                System.out.printf("%-10d %-20s %-10.2f %-10d%n", id, name, price, quantity);
            }
        }
    }


    private static void updateProduct(Connection conn, Scanner scanner) throws SQLException
{
        System.out.print("Enter product ID to update: "); int id =
        scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter new 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();

        String query = "UPDATE Product SET ProductName = ?, Price = ?, Quantity = ? WHERE ProductID =
    ?";

        try (PreparedStatement pstmt = conn.prepareStatement(query)) { conn.setAutoCommit(false);

            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
            pstmt.setInt(4, id);

            int rows = pstmt.executeUpdate(); conn.commit();

            System.out.println(rows + " product(s) updated successfully!");

        } catch (SQLException e) {
            conn.rollback();
            System.out.println("Transaction rolled back due to error: " + e.getMessage());
        } finally { conn.setAutoCommit(true);
        }
      }

      private static void deleteProduct(Connection conn, Scanner scanner) throws SQLException {
        System.out.print("Enter product ID to delete: ");
        int id = scanner.nextInt();
        String query = "DELETE FROM Product WHERE ProductID = ?"; try


        (PreparedStatement pstmt = conn.prepareStatement(query)) {


            conn.setAutoCommit(false);

            pstmt.setInt(1, id);
            int rows = pstmt.executeUpdate(); conn.commit();

            System.out.println(rows + " product(s) deleted successfully!");
        } catch (SQLException e) {
            conn.rollback();
            System.out.println("Transaction rolled back due to error: " + e.getMessage());
        } finally { conn.setAutoCommit(true);
        }
    }
}
```

7.2.4 Output:



```
(base) PS C:\Users\virat\OneDrive\Desktop\java exp7> java -cp ".;lib/mysql-connector-j-9.2.0.jar
" ProductCRUD
>>
Connected to the database!

=== Product CRUD Operations ===
1. Create Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 2

Product Records:
--------------------------------------------------
ProductID  ProductName          Price      Quantity
--------------------------------------------------
1          Laptop               75000.00   10
2          Mobile Phone         30000.00   25
3          Tablet               20000.00   15
4          Headphones           5000.00    50
5          Smartwatch           12000.00   30
6          Camera               45000.00   12
```

7.3.1 **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.

7.3.2 **Objective:**The objective of this program is to develop a menu-driven Java application that allows users to add employee details, display all stored employees, and exit the program. Employee details, including ID, name, designation, and salary, are stored persistently in a file using serialization.

7.3.3 Code:

StudentController.java

```
package controller;

import model.Student; import
java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class StudentController {

    private static final String URL = "jdbc:mysql://localhost:3306/StudentDB"; private static
    final String USER = "root";
    private static final String PASSWORD = "rishuraman1@V";

    // Method to create a new student
    public void createStudent(Student student) throws SQLException {
        String query = "INSERT INTO Student (Name, Department, Marks) VALUES (?, ?, ?)";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD); PreparedStatement
             pstmt = conn.prepareStatement(query)) {

            pstmt.setString(1, student.getName());
            pstmt.setString(2, student.getDepartment());
            pstmt.setDouble(3, student.getMarks());

            pstmt.executeUpdate(); System.out.println("Student added
            successfully!");
        }
    }

    // Method to retrieve all students
```

```
public List<Student> getAllStudents() throws SQLException { List<Student>
    students = new ArrayList<>();
    String query = "SELECT * FROM Student";

    try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD); Statement stmt =
         conn.createStatement();
         ResultSet rs = stmt.executeQuery(query)) {

        while (rs.next()) { students.add(new
            Student(
                    rs.getInt("StudentID"), rs.getString("Name"),
                    rs.getString("Department"),
                    rs.getDouble("Marks")
            ));
        }
    }
    return students;
}

// Method to update student data
public void updateStudent(Student student) throws SQLException {
    String query = "UPDATE Student SET Name = ?, Department = ?, Marks = ? WHERE StudentID = ?";

    try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD); PreparedStatement
         pstmt = conn.prepareStatement(query)) {

        pstmt.setString(1, student.getName());
        pstmt.setString(2, student.getDepartment());
        pstmt.setDouble(3, student.getMarks()); pstmt.setInt(4,
        student.getStudentID());

        int rows = pstmt.executeUpdate(); if
        (rows > 0) {
            System.out.println("Student updated successfully!");
        } else {
            System.out.println("Student not found.");
        }
    }
}
```

```
    // Method to delete a student
    public void deleteStudent(int studentID) throws SQLException { String query =
        "DELETE FROM Student WHERE StudentID = ?";

        try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD); PreparedStatement
             pstmt = conn.prepareStatement(query)) {

            pstmt.setInt(1, studentID);
            int rows = pstmt.executeUpdate(); if
            (rows > 0) {
                System.out.println("Student deleted successfully!");
            } else {
                System.out.println("Student not found.");
            }
        }
    }
}
```

Student.java

```
package model; public
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;
    }

    // Getters and Setters public int
    getStudentID() {
        return studentID;
    }

    public void setStudentID(int studentID) { this.studentID =
        studentID;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }

    public double getMarks() { return
        marks;
    }

    public void setMarks(double marks) {
        this.marks = marks;
    }

    @Override
    public String toString() {
        return String.format("ID: %d, Name: %s, Dept: %s, Marks: %.2f", studentID, name,
                department, marks);
    }
}
```

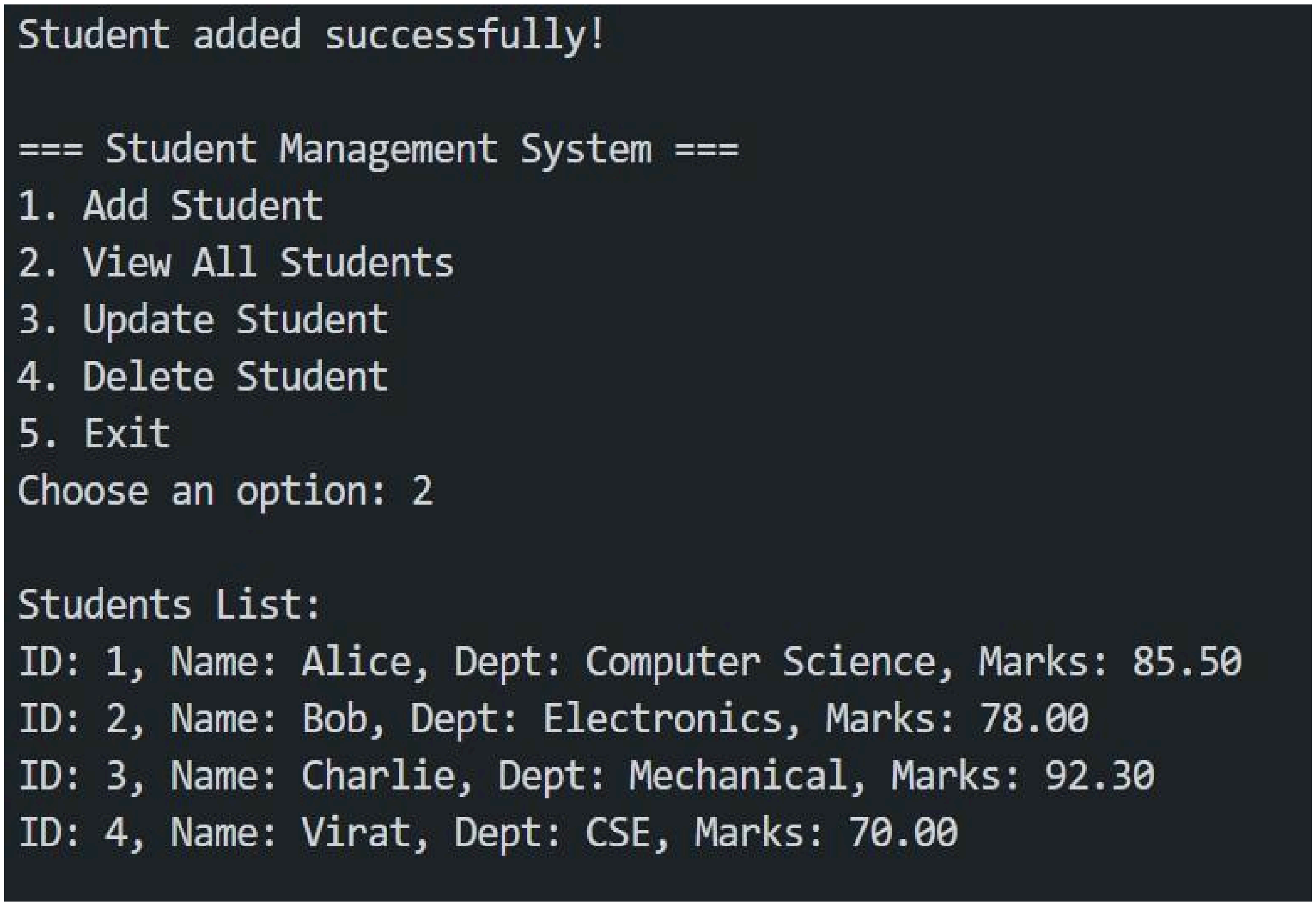
StudentView.java

```
package view;

import controller.StudentController;
import model.Student;

import java.util.List;
import java.util.Scanner; public

class StudentView {

    private static final Scanner scanner = new Scanner(System.in);
    private static final StudentController controller = new StudentController();

    public void displayMenu() { boolean
        exit = false;

        while (!exit) {
            System.out.println("\n=== Student Management System ===");
            System.out.println("1. Add Student");
            System.out.println("2. View All Students");
            System.out.println("3. Update Student");
            System.out.println("4. Delete Student");
            System.out.println("5. Exit");
            System.out.print("Choose an option: ");

            int choice = scanner.nextInt(); scanner.nextLine(); //
            Consume newline

            try {
                switch (choice) {
                    case 1 -> addStudent(); case 2
                    -> viewStudents(); case 3 ->
                    updateStudent(); case 4 ->
                    deleteStudent(); case 5 -> exit
                    = true;
                    default -> System.out.println("Invalid option. Try again.");
                }
            } catch (Exception e) { System.out.println("Error: " +
                e.getMessage());
            }
        }
        scanner.close();
    }
    private void addStudent() throws Exception {
        System.out.print("Enter name: ");
        String name = scanner.nextLine();
        System.out.print("Enter department: "); String
        department = scanner.nextLine();
        System.out.print("Enter marks: "); double
        marks = scanner.nextDouble();

        Student student = new Student(0, name, department, marks); controller.createStudent(student);
    }
    private void viewStudents() throws Exception { List<Student> students
        = controller.getAllStudents(); System.out.println("\nStudents List:");
        for (Student student : students) {
            System.out.println(student);
        }
    }
    private void updateStudent() throws Exception {
        System.out.print("Enter student ID to update: "); int id =
        scanner.nextInt();
        scanner.nextLine();

        System.out.print("Enter new name: "); String name
        = scanner.nextLine(); System.out.print("Enter new
        department: "); String department =
        scanner.nextLine(); System.out.print("Enter new
        marks: "); double marks = scanner.nextDouble();

        Student student = new Student(id, name, department, marks); controller.updateStudent(student);
    }

    private void deleteStudent() throws Exception {
        System.out.print("Enter student ID to delete: "); int id =
        scanner.nextInt(); controller.deleteStudent(id);
    }
}
```

MainApp.java

```
import view.StudentView;

public class MainApp {
    public static void main(String[] args) { StudentView view
        = new StudentView(); view.displayMenu();}}
```

7.3.4 Output:



```
Student added successfully!

=== Student Management System ===
1. Add Student
2. View All Students
3. Update Student
4. Delete Student
5. Exit
Choose an option: 2

Students List:
ID: 1, Name: Alice, Dept: Computer Science, Marks: 85.50
ID: 2, Name: Bob, Dept: Electronics, Marks: 78.00
ID: 3, Name: Charlie, Dept: Mechanical, Marks: 92.30
ID: 4, Name: Virat, Dept: CSE, Marks: 70.00
```

Learning Outcomes:

1. Understanding JDBC Integration**:**Gained practical experience in integrating JDBC with a Java application for database connectivity.
2. MVC Architecture Implementation**:**Learned how to implement the Model-View-Controller (MVC) architecture in Java for better code organization and separation of concerns.
3. Database CRUD Operations:Acquired the ability to perform CRUD operations (Create, Read, Update, Delete) using SQL queries in Java applications.
4. Transaction Handling**:**Understood the importance of transaction handling for maintaining data integrity during database operations.