Experiment-7

Name: Goutam **UID: 22BCS15338**
Branch: B.E-CSE **Section/Group: 22BCS-IOT-640/A**
Semester: 6th **Date of Performance: 10/03/2025**
Subject Name: PBJL **Subject Code: 22CSH-359**

1.

Aim: To develop a Java program that connects to a MySQL database using JDBC and retrieves all records from the Employee table, displaying them in the console.

2.

Objective: To develop a Java program that connects to a MySQL database using JDBC and retrieves all records from the Employee table, displaying them in the console.

3. Code:

```
import   java.sql.Connection;
import
java.sql.DriverManager;
import   java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Employee {          public static
void main(String[] args) {
        // Database connection details
        String url = "jdbc:mysql://localhost:3306/EmployeeDB"; // Replace
'mydatabase' with your DB name
        String user = "root"; // Replace with your MySQL username
        String password = "java123"; // Replace with your MySQL
password try {
            // Establishing the database connection
            Connection conn = DriverManager.getConnection(url, user, password);
System.out.println("Connected to the database!");
```

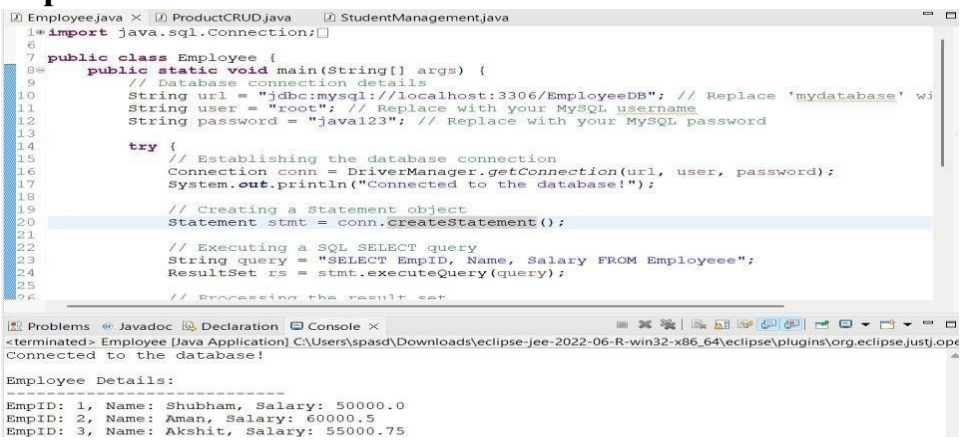
```
            // Creating a Statement object
            Statement stmt = conn.createStatement();
            // Executing a SQL SELECT query
            String query = "SELECT EmpID, Name, Salary FROM Employeee";
ResultSet rs = stmt.executeQuery(query);

            // Processing the result set
            System.out.println("\nEmployee Details:");


  System.out.println(" ------------------- ");
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);
            }

            // Closing resources
            rs.close();
stmt.close();
conn.close();
            System.out.println("Connection closed.");
} catch (SQLException e) {
e.printStackTrace();
        }
    }
}
```

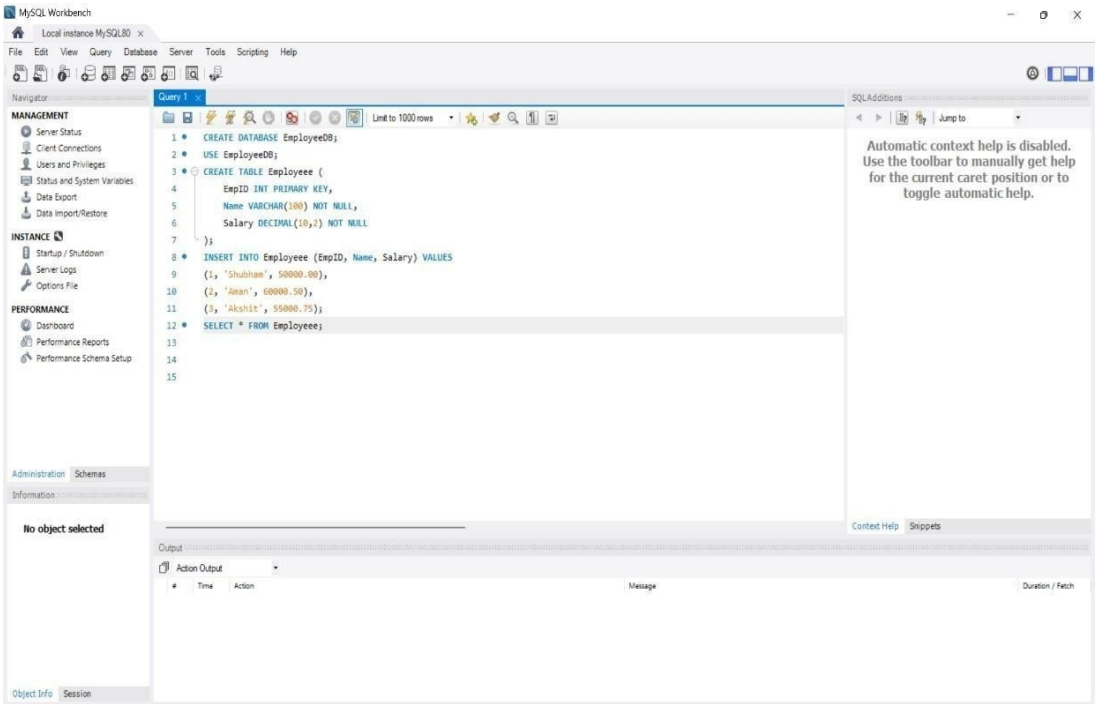
//MYSQLCODE:

```
CREATE DATABASE EmployeeDB;
USE EmployeeDB;
CREATE TABLE Employeee (EmpID INT PRIMARY KEY,Name
VARCHAR(100)NOT NULL,Salary DECIMAL(10,2) NOT NULL);
INSERT INTO Employeee (EmpID, Name, Salary) VALUES
(1, 'Shubham', 50000.00),
(2, 'Aman', 60000.50),
(3, 'Akshit', 55000.75);
```

SELECT * FROM Employeee;

4. Output:





5. Learning Outcomes:

- Learn how to establish a connection between Java and a MySQL database using the DriverManager class.
- Understand how to execute SQL queries using Statement and ResultSet.
- Learn how to perform a SELECT query to retrieve data from a MySQL table.

- Understand how to process the results using ResultSet methods like .getInt(), .getString(), and .getDouble().
- Learn the importance of closing database resources (ResultSet, Statement, Connection) to prevent memory leaks.

Problem 2

1. **Aim:** To develop a Java program that performs CRUD (Create, Read, Update, Delete) operations on a MySQL database table Product, ensuring data integrity using transaction handling.

2. **Objective**: Build a menu-driven Java application that allows users to manage product records in a MySQL database with transactional support.

3. **Code:**

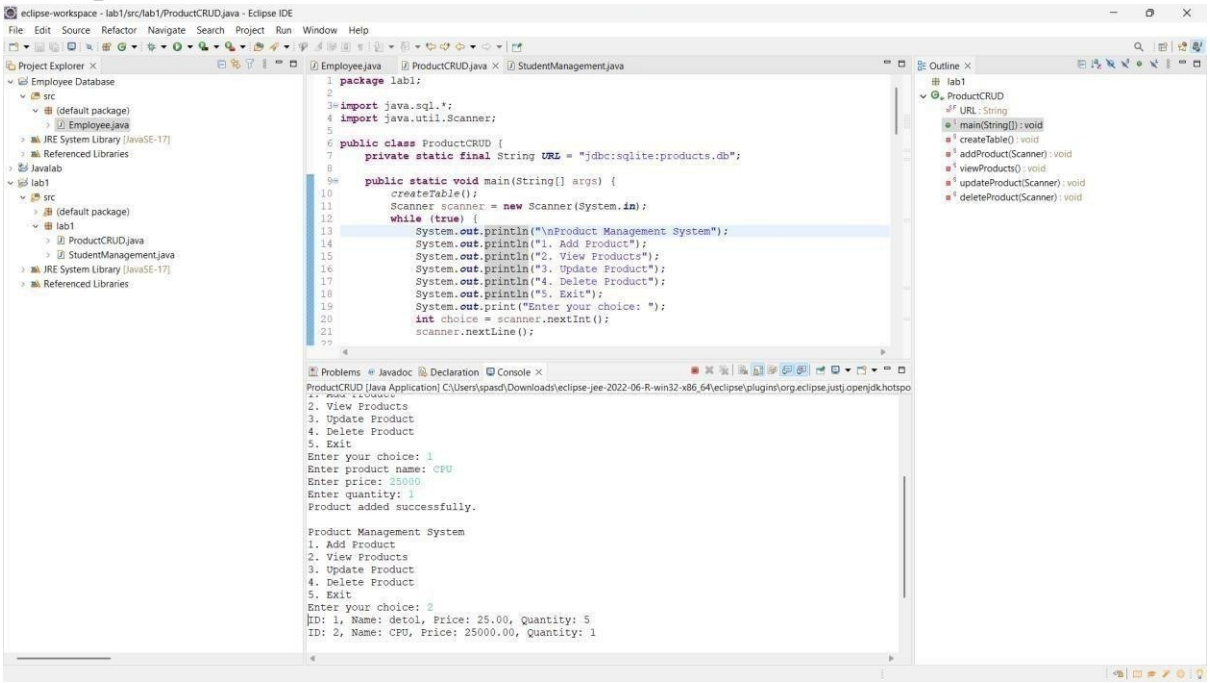
```
package lab1; import
java.sql.*; import
java.util.Scanner; public
class
ProductCRUD {
private static final String URL = "jdbc:sqlite:products.db";
  public static void main(String[] args) {
createTable();
  Scanner scanner = new Scanner(System.in);
while (true) {
      System.out.println("\nProduct Management System");
      System.out.println("Add Product");
      System.out.println("View Products");
      System.out.println("Update Product");
      System.out.println("Delete Product");
      System.out.println("Exit");
      System.out.print("Enter your choice: ");
int choice = scanner.nextInt();
```

```
scanner.nextLine(); switch (choice) {        case
1 -> addProduct(scanner);
case 2 -> viewProducts();                case 3 -
                                         >
updateProduct(scanner);                 case 4 ->
deleteProduct(scanner);                case 5 -> {
                System.out.println("Exiting...");
scanner.close();                    return;
            } default -> System.out.println("Invalid choice. Try
        again."); }
    } }
  private static void createTable() {
    String sql = "CREATE TABLE IF NOT EXISTS Product (ProductID INTEGER
PRIMARY KEY AUTOINCREMENT, ProductName TEXT, Price REAL, Quantity
INTEGER)"; try (Connection conn = DriverManager.getConnection(URL);
    Statement stmt =
conn.createStatement()) {
stmt.execute(sql); }
catch (SQLException
e) {
e.printStackTrace(); }
} private static void addProduct(Scanner scanner)
    { 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 sql = "INSERT INTO Product (ProductName, Price, Quantity) VALUES (?,
?, ?)"; try (Connection conn =
    DriverManager.getConnection(URL);
PreparedStatement pstmt = conn.prepareStatement(sql)) {        pstmt.setString(1, name);
    pstmt.setDouble(2, price);     pstmt.setInt(3, quantity); pstmt.executeUpdate();
```

```
            System.out.println("Product added
successfully."); } catch (SQLException e) {
        e.printStackTrace(); }
    }
    private static void viewProducts() {
String sql = "SELECT * FROM Product"; try (Connection conn =
        DriverManager.getConnection(URL); Statement stmt =
conn.createStatement(); ResultSet rs = stmt.executeQuery(sql)) {
while (rs.next()) {
                System.out.printf("ID: %d, Name: %s, Price: %.2f, Quantity: %d\n",
rs.getInt("ProductID"), rs.getString("ProductName"), rs.getDouble("Price"),
rs.getInt("Quantity"));
            }
        } catch (SQLException e)
{ e.printStackTrace();


    }} private static void updateProduct(Scanner
scanner) { System.out.print("Enter ProductID to update:
");        int id = scanner.nextInt();
System.out.print("Enter new price: ");       double price
= scanner.nextDouble();
System.out.print("Enter new quantity: "); int quantity =
scanner.nextInt();
        String sql = "UPDATE Product SET Price = ?, Quantity = ? WHERE ProductID =
?";
        try (Connection conn = DriverManager.getConnection(URL);
PreparedStatement pstmt = conn.prepareStatement(sql)) {           pstmt.setDouble(1,
price);        pstmt.setInt(2, quantity);        pstmt.setInt(3, id); int rows =
pstmt.executeUpdate();
            System.out.println(rows > 0 ? "Product updated successfully." : "Product not
found.");
```

```
        } catch (SQLException e)
{ e.printStackTrace();
        }
    } private static void deleteProduct(Scanner scanner)
{ System.out.print("Enter ProductID to delete: "); int
id
= scanner.nextInt();
        String sql = "DELETE FROM Product WHERE ProductID = ?"; try
        (Connection conn = DriverManager.getConnection(URL);
PreparedStatement pstmt = conn.prepareStatement(sql)) { pstmt.setInt(1, id);
            int rows = pstmt.executeUpdate();
            System.out.println(rows > 0 ? "Product deleted successfully." : "Product not
found.");
        } catch (SQLException e)
{ e.printStackTrace();
        }
    } }
```

4. Output:



5. **Learning Outcomes:**
   - Establish and close a database connection using JDBC.
   - ExecuteSQL queries (INSERT, SELECT, UPDATE, DELETE) using PreparedStatement for security and efficiency.
   - Implement Create, Read, Update, and Delete functionalities in Java with a MySQL database.
   - Implement Create, Read, Update, and Delete functionalities in Java with a MySQL database.
   - Learn to build interactive Java applications with user input handling.

Problem 3

1.

Aim: To develop a Java application using JDBC and the Model-View-Controller (MVC) architecture to manage student records in a MySQL database. **2.**

Objective: Create a structured Java program that follows the MVC architecture, enabling users to perform CRUD (Create, Read, Update, Delete) operations on student data stored in a database. **3. Code:**

```
package lab1;


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; }
```

```
    @Override     public String toString() {         return "ID: " + studentID + ", Name: "
+ name + ", Department: " + department + ", Marks: " + marks;
    }
}
// Controller: Handles Database Operations class
StudentController {     private static final String URL =
"jdbc:sqlite:students.db";     public StudentController()
{ createTable();
    }

    private void createTable() {
        String sql = "CREATE TABLE IF NOT EXISTS Student ("
                + "StudentID INTEGER PRIMARY KEY AUTOINCREMENT, "
                + "Name TEXT NOT NULL, "


 + "Department TEXT NOT NULL, "
                + "Marks REAL NOT NULL)"; try (Connection
conn = DriverManager.getConnection(URL); Statement stmt =
conn.createStatement()) { stmt.execute(sql); } catch
(SQLException e) { e.printStackTrace();
        } }
    public void addStudent(String name, String department, double marks) {
        String sql = "INSERT INTO Student (Name, Department, Marks) VALUES (?, ?,
?)";
        try (Connection conn = DriverManager.getConnection(URL);
PreparedStatement pstmt = conn.prepareStatement(sql)) { pstmt.setString(1,
      name);            pstmt.setString(2,       department); pstmt.setDouble(3,
      marks);           pstmt.executeUpdate(); System.out.println("Student added
successfully."); } catch
(SQLException e) { e.printStackTrace(); } } public
    void viewStudents() {
```

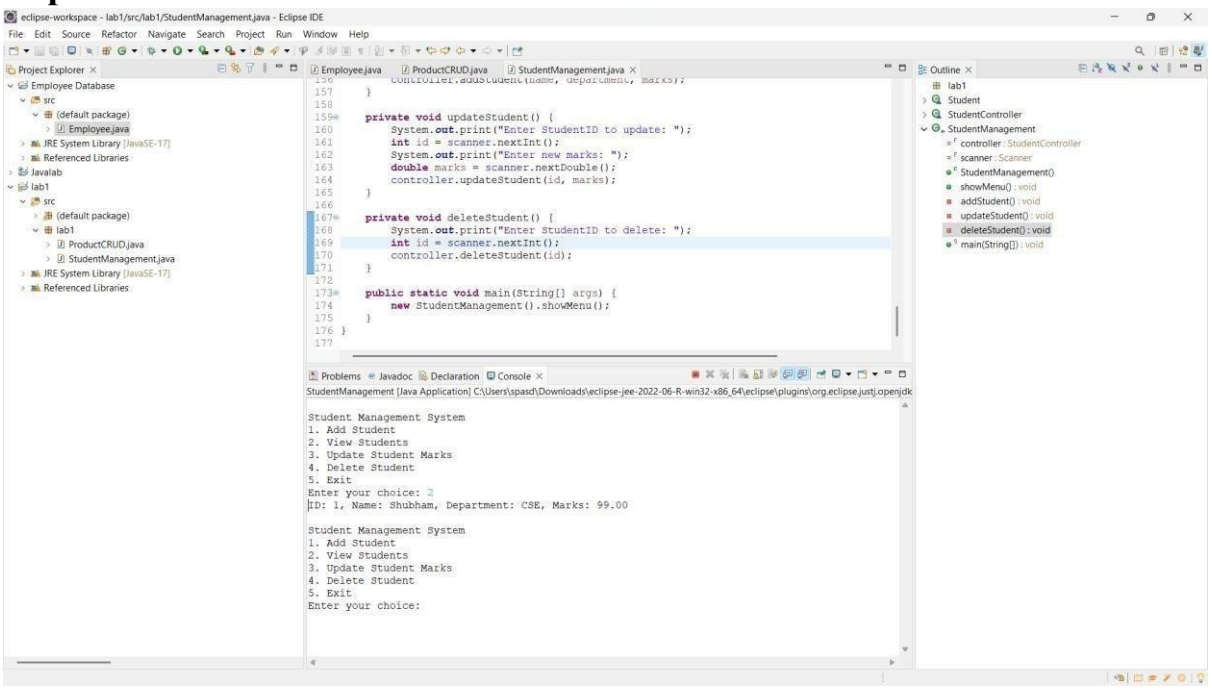
```
        String sql = "SELECT * FROM Student"; try (Connection
        conn = DriverManager.getConnection(URL); Statement stmt
        = conn.createStatement();

ResultSet rs = stmt.executeQuery(sql)) {            if
(!rs.isBeforeFirst()) {
                System.out.println("No students found.");
return; } while (rs.next())
            {
                System.out.printf("ID: %d, Name: %s, Department: %s, Marks: %.2f%n",
rs.getInt("StudentID"), rs.getString("Name"),     rs.getString("Department"),
rs.getDouble("Marks"));


  }} catch (SQLException e)
{ e.printStackTrace();
        } }
    public void updateStudent(int studentID, double newMarks) {
        String sql = "UPDATE Student SET Marks = ? WHERE StudentID =
?"; try (Connection conn = DriverManager.getConnection(URL);
PreparedStatement pstmt = conn.prepareStatement(sql)) {
pstmt.setDouble(1, newMarks); pstmt.setInt(2, studentID);       int rows =
pstmt.executeUpdate();
            System.out.println(rows > 0 ? "Student updated successfully." : "Student not
found.");
        } catch (SQLException e)
{ e.printStackTrace();
        } }
    public void deleteStudent(int studentID) {
        String sql = "DELETE FROM Student WHERE StudentID = ?";
try (Connection conn = DriverManager.getConnection(URL);
PreparedStatement pstmt = conn.prepareStatement(sql))
```

```
{ pstmt.setInt(1, studentID);        int rows = pstmt.executeUpdate();
System.out.println(rows > 0 ? "Student deleted successfully." : "Student not found.");
        } catch (SQLException e)
{ e.printStackTrace();
        }
    }
}

// View: User Interface (Menu) public class
      StudentManagement {           private final
StudentController controller;     private final
Scanner scanner;     public
StudentManagement() {


 controller = new StudentController();
scanner = new Scanner(System.in);
    } public void
    showMenu()
{ while (true) {
            System.out.println("\nStudent Management System");
            System.out.println("1. 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("Enter your choice: "); int
choice = scanner.nextInt();
scanner.nextLine();          switch (choice) {
case 1 -> addStudent();     case 2 ->
controller.viewStudents(); case 3 ->
updateStudent();    case 4 -> deleteStudent();
      case 5 -> { System.out.println("Exiting...");
```

```
scanner.close();                    return;
            } default -> System.out.println("Invalid choice. Try
            again.");
        }
    } }

    private void addStudent()
        { 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();
controller.addStudent(name, department,
marks);
    }
    private void updateStudent() {
        System.out.print("Enter StudentID to update: "); int
id = scanner.nextInt();
        System.out.print("Enter new marks: ");
double marks = scanner.nextDouble();
controller.updateStudent(id, marks);
    }
    private void deleteStudent() {
        System.out.print("Enter StudentID to delete: ");
int  id        =        scanner.nextInt();
controller.deleteStudent(id); }
    public static void main(String[]
    args)
{ new StudentManagement().showMenu();
    }
```

```
}
```

4. Output:



5. Learning Outcomes:

- Learn how to separate concerns in a Java application using Model (Student class), View (User Interface), and Controller (Database operations).
- Establish a connection with MySQL using JDBC.
- Use PreparedStatements to securely execute SQL queries.
- Implement Create, Read, Update, and Delete functions to manage student records.
- Develop an interactive user interface for managing student data.

CU
CHANDIGARH UNIVERSITY
Discover. Learn. Empower.

DEPARTMENT OF COMPUTERSCIENCE&ENGINEERING