Experiment 7

Student Name: Om Kumar Kushwaha **UID: 22BCS13906**

Branch: BE-CSE **Section/Group: IOT-642/B**

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

Subject : Project Based Learning In Java with Lab **Subject Code: 22CSH-359**

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.

1. **Objective:** The objective of this program is to develop a Java application that connects to a MySQL database and performs data management using JDBC. The program will establish a secure connection via DriverManager and Connection objects to interact with the database. It will retrieve and display records from the Employee table, which consists of EmpID, Name, and Salary columns. Additionally, the application will implement CRUD operations on a Product table with ProductID, ProductName, Price, and Quantity fields using a menu-driven interface and transaction handling to ensure data integrity. Furthermore, the program will incorporate an MVC-based Student Management System,

Required Tools & Technologies:

1. **MySQL Database** – To store and retrieve employee records.
2. **MySQL Workbench** – To create the database and table.
3. **JDK (Java Development Kit)** – To compile and run the Java program.
4. **JDBC (Java Database Connectivity)** – To enable communication between Java and MySQL.
5. **MySQL Connector/J** – JDBC driver for connecting Java with MySQL.
6. **VS Code or Any Java IDE** – To write, compile, and run the Java code

Implementation/Code:

IN MY SQL:

```
CREATE DATABASE EmployeeDB;
USE EmployeeDB;
CREATE TABLE Employee (
   EmpID INT PRIMARY KEY AUTO_INCREMENT,
   Name VARCHAR(100) NOT NULL,
   Salary DECIMAL(10,2) NOT NULL
);
INSERT INTO Employee (Name, Salary) VALUES
('PRINCE', 50000.00),
('SKY', 60000.00),
('SUKHU', 70000.00);

SELECT * FROM Employee;
```

IN IDE

```
package src;
import java.sql.*;

public class EmployeeDB {
   public static void main(String[] args) {
      String url = "jdbc:mysql://localhost:3306/EmployeeDB";
      String user = "root";
      String password = "50125";
      try {

         Class.forName("com.mysql.cj.jdbc.Driver");
         Connection conn = DriverManager.getConnection(url, user, password);
         System.out.println("Connected to MySQL!");

 String query = "SELECT * FROM Employee";
         Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(query);
         System.out.println("Employee Data:");
         while (rs.next())

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

```
                System.out.println(empID + " | " + name + " | $" + salary);
            }
            rs.close();
            stmt.close();
            conn.close();
            System.out.println("Connection closed.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. Output:

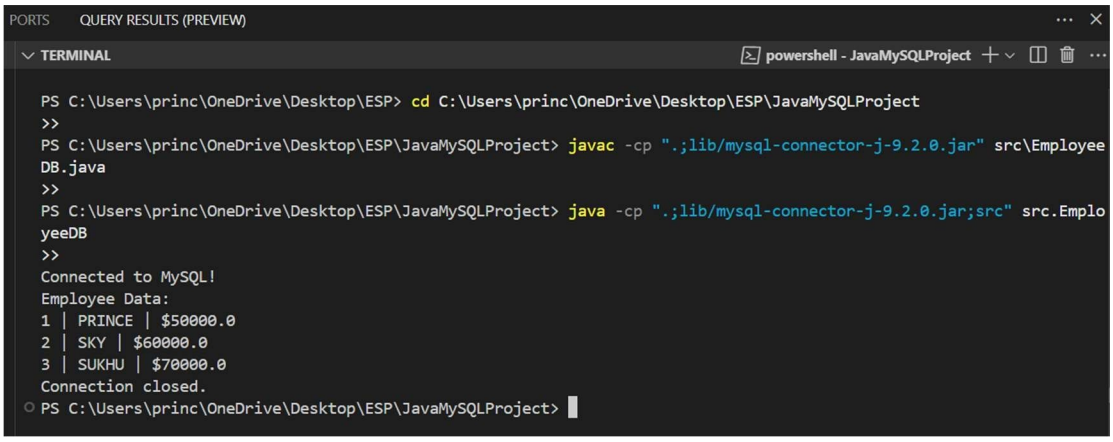

FIG1: OUTPUT IN IDE

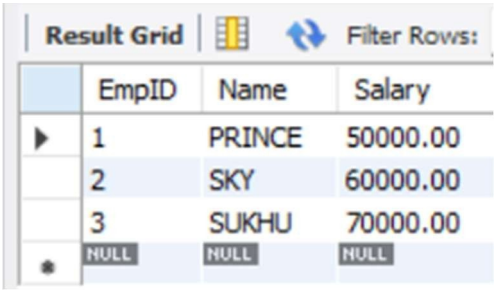

FIG2: OUTPUT IN MYSQL

EXP: 7.2

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.

IN MY SQL:

```
CREATE DATABASE ProductDB;
USE ProductDB;

CREATE TABLE Product (
    ProductID INT AUTO_INCREMENT PRIMARY KEY,
    ProductName VARCHAR(100) NOT NULL,
    Price DECIMAL(10,2) NOT NULL,
    Quantity INT NOT NULL
);
```

IN IDE:

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

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

      while (true) {
         System.out.println("\n===== Product Management System =====");
         System.out.println("1. Insert 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)
```

```
            { case1:
               insertProduct(s
               canner); break;
               case 2:
                  readProducts();
                  break;
               case 3:
                  updateProduct(scanner);
                  break;
               case 4:
                  deleteProduct(scanner);
                  break;
               case 5:
                  System.out.println("Exiting...");
                  scanner.close();
                  return;
               default:
                  System.out.println("Invalid choice! Try again.");
            }
         }
      }
      private static void insertProduct(Scanner scanner) {
         try (Connection conn = DBConnection.getConnection())
            { 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();
            scanner.nextLine();
            String sql = "INSERT INTO Product (ProductName, Price, Quantity)
VALUES (?, ?, ?)";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setString(1, name);
            pstmt.setDouble(2, price);
            pstmt.setInt(3, quantity);
            int rowsInserted = pstmt.executeUpdate();
            System.out.println(rowsInserted + " product(s) added successfully.");
         } catch (Exception e) {
            e.printStackTrace();
         }

}
      private static void readProducts() {
```

```
        try (Connection conn = DBConnection.getConnection();
             Statement stmt = conn.createStatement()) {
            String sql = "SELECT * FROM Product";
            ResultSet rs = stmt.executeQuery(sql);
            System.out.println("\nProduct List:");
            while (rs.next()) {
                System.out.println(rs.getInt("ProductID") + " | " +
                                   rs.getString("ProductName") + " | $" +
                                   rs.getDouble("Price") + " | " +
                                   rs.getInt("Quantity"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }


    private static void updateProduct(Scanner scanner) {
        try (Connection conn = DBConnection.getConnection())
            { System.out.print("Enter Product ID to update: ");
            int productId = scanner.nextInt();
            scanner.nextLine();
            System.out.print("Enter new Price: ");
            double newPrice = scanner.nextDouble();
            System.out.print("Enter new Quantity: ");
            int newQuantity = scanner.nextInt();
            scanner.nextLine();
            String sql = "UPDATE Product SET Price = ?, Quantity = ? WHERE
ProductID = ?";

            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setDouble(1, newPrice);
            pstmt.setInt(2, newQuantity);
            pstmt.setInt(3, productId);

            int rowsUpdated = pstmt.executeUpdate();
            System.out.println(rowsUpdated + " product(s) updated
successfully.");



        } catch (Exception e)

{
```

```
            e.printStackTrace();

}
    }
private static void deleteProduct(Scanner scanner) {
        try (Connection conn = DBConnection.getConnection())
            { System.out.print("Enter Product ID to delete: ");
            int productId = scanner.nextInt();
            scanner.nextLine();
            String sql = "DELETE FROM Product WHERE ProductID = ?";
            PreparedStatement pstmt = conn.prepareStatement(sql);
            pstmt.setInt(1, productId);

            int rowsDeleted = pstmt.executeUpdate();
            System.out.println(rowsDeleted + " product(s) deleted successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

OUTPUT

OPERATION 1 CREATE (CRUD)

```
===== Product Management System =====
1. Insert Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 1
Enter Product Name: IPHONE 16 PRO
Enter Price: 80000
Enter Quantity: 2
1 product(s) added successfully.
```

Result Grid | Filter Rows: | Edit:

ProductID	ProductName	Price	Quantity
1	LAPTOP	14000.00	2
2	IPHONE 16 PRO	80000.00	2
NULL	NULL	NULL	NULL

OPERATION 3 UPDATE

```
===== Product Management System =====
1. Insert Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 3
Enter Product ID to update: 1
Enter new Price: 75000
Enter new Quantity: 3
1 product(s) updated successfully.
```

Result Grid | Filter Rows: | Edit:

ProductID	ProductName	Price	Quantity
1	LAPTOP	75000.00	3
2	IPHONE 16 PRO	80000.00	2
NULL	NULL	NULL	NULL

OPERATION 4 DELETION

```
===== Product Management System =====
1. Insert Product
2. Read Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 4
Enter Product ID to delete: 2
1 product(s) deleted successfully.
```

Result Grid | Filter Rows: | Edit:

ProductID	ProductName	Price	Quantity
1	LAPTOP	75000.00	3
NULL	NULL	NULL	NULL

7.3

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.

IN MY SQL:

```
CREATE DATABASE StudentDB;
USE StudentDB;
CREATE TABLE Student (
    StudentID INT PRIMARY KEY AUTO_INCREMENT,
    Name VARCHAR(100),
    Department VARCHAR(50),
    Marks INT
);
```

IN IDE:

WE need Such Folder Directory

- `src/main/` → Contains the `Main.java` file, which starts the application.
- `src/model/` → Defines Java classes that map to database tables (like `Student.java`).
- `src/controller/` → Contains logic for handling CRUD operations.
- `src/view/` → Handles user input and output (menu-driven interface).
- `src/utils/` → Stores utility/helper classes like `DBConnection.java` for database connectivity.

Model.java:->

```
package model;
public class Student
{
    private int studentID;
    private String name;

private String department;
    private int marks;
    public Student(int studentID, String name, String department, int 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 int getMarks() { return marks; }
}
```

Student.java:

```
package model;
public class Student
{
    private int studentID;
    private String name;
    private String department;
    private int marks;
    public Student(int studentID, String name, String department, int 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 int getMarks() { return marks; }
}
```

StudentView.java

```
package view;
import controller.StudentController;
import model.Student;
import java.util.List;
import java.util.Scanner;

public class StudentView {
    public static void main(String[] args)
        { Scanner scanner = new
        Scanner(System.in); while (true) {
            System.out.println("\n===== Student Management System =====");
            System.out.println("1. Add Student");
            System.out.println("2. View 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();
        switch (choice) {
            case 1:
                System.out.print("Enter Name: ");

                String name = scanner.nextLine();
                System.out.print("Enter Department: ");
                String dept = scanner.nextLine();
                System.out.print("Enter Marks: ");
                int marks = scanner.nextInt();
                StudentController.addStudent(name, dept, marks);
                break;
            case 2:
                List<Student> students = StudentController.getAllStudents();
                System.out.println("\nStudent List:");
                for (Student s : students) {
                    System.out.println(s.getStudentID() + " | " + s.getName() + " | " +
s.getDepartment() + " | " + s.getMarks());
                }
                break;
            case 3:
                System.out.print("Enter Student ID to update: ");
                int updateID = scanner.nextInt();
                scanner.nextLine();
                System.out.print("Enter New Name: ");
                String newName = scanner.nextLine();
                System.out.print("Enter New Department: ");
                String newDept = scanner.nextLine();
                System.out.print("Enter New Marks: ");
                int newMarks = scanner.nextInt();
                StudentController.updateStudent(updateID, newName, newDept,
                newMarks); break;
            case 4:
                System.out.print("Enter Student ID to delete: ");
                int deleteID = scanner.nextInt();
                StudentController.deleteStudent(deleteID);
                break;
            case 5:
                System.out.println("Exiting...");
                scanner.close();
                return;
```

```
            default:
                System.out.println("Invalid choice! Try again.");
        }
    }
  }
}
```

Student Controller.java

```
package controller;
import model.Student;
import utils.DBConnection;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class StudentController {
   public static void addStudent(String name, String department, int marks) {
      String sql = "INSERT INTO Student (Name, Department, Marks) VALUES (?, ?,
      ?)"; try (Connection conn = DBConnection.getConnection();
          PreparedStatement pstmt = conn.prepareStatement(sql))
          { pstmt.setString(1, name);
          pstmt.setString(2, department);
          pstmt.setInt(3, marks);
          pstmt.executeUpdate();
          System.out.println("Student added successfully.");
      } catch (Exception e) {
          e.printStackTrace();
      }
   }
   public static List<Student> getAllStudents()
      { List<Student> students = new ArrayList<>();
      String sql = "SELECT * FROM Student";
      try (Connection conn = DBConnection.getConnection();
          Statement stmt = conn.createStatement();
          ResultSet rs = stmt.executeQuery(sql))
          { while (rs.next()) {
              students.add(new Student(rs.getInt("StudentID"), rs.getString("Name"),
                    rs.getString("Department"), rs.getInt("Marks")));
          }
      } catch (Exception e) {
          e.printStackTrace();
      }
      return students;

 }
```

```
  public static void updateStudent(int studentID, String name, String department,
int marks) {
        String sql = "UPDATE Student SET Name = ?, Department = ?, Marks = ? WHERE
StudentID = ?";
        try (Connection conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql))
            { pstmt.setString(1, name);
            pstmt.setString(2, department);
            pstmt.setInt(3, marks);
            pstmt.setInt(4, studentID);
            pstmt.executeUpdate();
            System.out.println("Student updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void deleteStudent(int studentID) {
        String sql = "DELETE FROM Student WHERE StudentID = ?";
        try (Connection conn = DBConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql))
            { pstmt.setInt(1, studentID);
            pstmt.executeUpdate();
            System.out.println("Student deleted
            successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
DBConnection.java
package controller;
import model.Student;
import utils.DBConnection;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;

public class StudentController {
    public static void addStudent(String name, String department, int marks) {
        String sql = "INSERT INTO Student (Name, Department, Marks) VALUES (?, ?,
        ?)"; try (Connection conn = DBConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql))
            { pstmt.setString(1, name);
            pstmt.setString(2, department);

 pstmt.setInt(3, marks);
```

```
            pstmt.executeUpdate();
            System.out.println("Student added
            successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static List<Student> getAllStudents()
        { List<Student> students = new ArrayList<>();

String sql = "SELECT * FROM Student";
        try (Connection conn = DBConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql))
            { while (rs.next()) {
                students.add(new Student(rs.getInt("StudentID"), rs.getString("Name"),
                        rs.getString("Department"), rs.getInt("Marks")));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return students;
    }
    public static void updateStudent(int studentID, String name, String department, int
marks) {
        String sql = "UPDATE Student SET Name = ?, Department = ?, Marks = ? WHERE
StudentID = ?";
        try (Connection conn = DBConnection.getConnection();
            PreparedStatement pstmt = conn.prepareStatement(sql))
            { pstmt.setString(1, name);
            pstmt.setString(2, department);
            pstmt.setInt(3, marks);
            pstmt.setInt(4, studentID);
            pstmt.executeUpdate();
            System.out.println("Student updated successfully.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static void deleteStudent(int studentID) {
        String sql = "DELETE FROM Student WHERE StudentID = ?";
        try (Connection conn = DBConnection.getConnection();
             PreparedStatement pstmt = conn.prepareStatement(sql))
            { pstmt.setInt(1, studentID);
            pstmt.executeUpdate();
            System.out.println("Student deleted
            successfully.");
```

```
            e.printStackTrace();
        }
    }
}
```

OUTPUT IN IDE:

```
PS C:\Users\princ\OneDrive\Desktop\ESP\StudentManagement> 
jar" main.Main
>>
Starting Student Management System...

===== Student Management System =====
1. Add Student
2. View Students
3. Update Student
4. Delete Student
5. Exit
Choose an option: 1
Enter Name: PRINCE
Enter Department: COMPUTER SCIENCE ENGINEERING
Enter Marks: 20
```

```
Enter Student ID to update: 1
Enter New Name: PRINCE DOGRA
Enter New Department: CSE-AIML
Enter New Marks: 20
Student updated successfully.
```

OUTPUT IN MYSQL WORKBENCH

Result Grid | Filter Rows: | Edit:

StudentID	Name	Department	Marks
1	PRINCE DOGRA	CSE-AIML	20
2	SKY	MSC	18
NULL	NULL	NULL	NULL

5. Learning Outcomes:

1. Establishing database connectivity in Java using DriverManager and Connection objects to connect with MySQL.
2. Performing basic and advanced database operations by retrieving data using SQL SELECT queries and displaying records from tables like Employee and Product.
3. Implementing CRUD operations with transaction handling to create, read, update, and delete records in the Product table while ensuring data integrity.
4. Designing applications using the MVC architecture, separating concerns with Model (Student.java), View (StudentView.java), and Controller (StudentController.java) for structured program design.