Experiment 7

Student Name: Dipendra Kumar Sah **UID: 22BCS17184**
Branch: CSE **Section: IOT-642 -B**
Semester: 6th **DOP: 17/03/025**
Subject: Java **Subject Code:22CSH-359**

Problem - 7.1

Aim:

1. Setup MySQL Database
   - Ensure MySQL is installed and running.
   - Create a database and an `Employee` table with columns `EmpID`, `Name`, and `Salary`.

2. Update Database Credentials
   - Replace `your_database`, `your_username`, and `your_password` in the code with actual database credentials.

3. Add MySQL JDBC Driver
   - Download and add `mysql-connector-java.jar` to your project's classpath.

4. Compile and Run the Program
   - Compile: `javac MySQLConnection.java`
   - Run: `java MySQLConnection`

5. Verify Output
   - Ensure that employee records are displayed correctly from the database.

Code:

//SQL CODE

```
CREATE DATABASE CompanyDB;

USE CompanyDB;
CREATE TABLE Employee (
    EmpID INT PRIMARY KEY,
    Name VARCHAR(100),
    Salary DECIMAL(10, 2)
);

INSERT INTO Employee (EmpID, Name, Salary) VALUES
(101, 'Dipendra', 50000.00),
(102, 'Yubraj', 60000.00),
(103, 'Om', 55000.00);
```

//JAVA CODE

```
package JavaExp7;

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

public class MySQLConnection {
    public static void main(String[] args) {
        // Step 2: Update these with your actual database credentials
        String url = "jdbc:mysql://localhost:3306/CompanyDB";
        String username = "root";
        String password = "Dipendra@1232";

        try {
            // Step 3: Load MySQL JDBC Driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Step 4: Establish connection
            Connection connection = DriverManager.getConnection(url, "root", "Dipendra@1232");
            System.out.println("Connected to the database successfully!");

            // Execute SQL query
            Statement statement = connection.createStatement();
            String query = "SELECT * FROM Employee";
            ResultSet resultSet = statement.executeQuery(query);

            // Step 5: Verify Output
            System.out.println("Fetching employee records from database...\n");
            while (resultSet.next()) {
                int empId = resultSet.getInt("EmpID");
                String name = resultSet.getString("Name");
                double salary = resultSet.getDouble("Salary");
```

```
                System.out.println("EmpID: " + empId + ", Name: " + name + ", Salary: " + salary);
            }



            resultSet.close();
            statement.close();
            connection.close();

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

Output:

```
Connected to the database successfully!
Fetching employee records from database...

EmpID: 101, Name: Dipendra, Salary: 50000.0
EmpID: 102, Name: Yubraj, Salary: 60000.0
EmpID: 103, Name: Om, Salary: 55000.0

Process finished with exit code 0
```

Problem - 7.2

Aim : Instructions to Run the Java CRUD Program

1. Setup MySQL Database

   - Ensure MySQL is installed and running.

   - Create a database and a `Product` table with columns `ProductID`, `ProductName`, `Price`, and `Quantity`

2. Update Database Credentials

   - Replace `your_database`, `your_username`, and `your_password` in the code with actual database credentials

3. Add MySQL JDBC Driver

   - Download and add `mysql-connector-java.jar` to your project's classpath.

4. Compile and Run the Progra

   - Compile: `javac ProductCRUD.java`

   - Run: `java ProductCRUD`

5. Menu-Driven Operations

   - Select options to **Create**, **Read**, **Update**, or **Delete** products.

   - Input values as prompted.

6. Transaction Handling

   - Transactions ensure data integrity.

   - If an error occurs, changes are rolled back.

7. Verify Output

- Ensure product records are correctly manipulated in the database.

Code :

```
//SQL CODE
CREATE DATABASE IF NOT EXISTS InventoryDB;

USE InventoryDB;

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

//JAVA CODE
package JavaExp7;

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

public class ProductCRUD {
    static final String URL = "jdbc:mysql://localhost:3306/InventoryDB";
    static final String USER = "root";
    static final String PASSWORD = "Dipendra@1232";

    public static void main(String[] args) {
        try (
                Connection conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/InventoryDB",
"root", "Dipendra@1232");
                Scanner scanner = new Scanner(System.in)
        ) {
            Class.forName("com.mysql.cj.jdbc.Driver");
            System.out.println("Connected to database.");

            while (true) {
                System.out.println("\n====== Product CRUD Menu ======");
                System.out.println("1. Add Product");
                System.out.println("2. View 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();  // consume newline
```

```
            switch (choice) {
                case 1 -> addProduct(conn, scanner);
                case 2 -> viewProducts(conn);
                case 3 -> updateProduct(conn, scanner);
                case 4 -> deleteProduct(conn, scanner);
                case 5 -> {
                    System.out.println("Exiting program.");
                    return;
                }
                default -> System.out.println("Invalid option. Try again.");
            }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
}

// Create
private static void addProduct(Connection conn, 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 qty = scanner.nextInt();

    String sql = "INSERT INTO Product (ProductName, Price, Quantity) VALUES (?, ?, ?)";
    try (PreparedStatement stmt = conn.prepareStatement(sql)) {
        conn.setAutoCommit(false);  // begin transaction

        stmt.setString(1, name);
        stmt.setDouble(2, price);
        stmt.setInt(3, qty);
        stmt.executeUpdate();

        conn.commit();  // commit transaction
        System.out.println("Product added successfully.");
    } catch (SQLException e) {
        try {
            conn.rollback();  // rollback on error
            System.out.println("Error occurred. Transaction rolled back.");
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        e.printStackTrace();
    }
}

// Read
```

```
private static void viewProducts(Connection conn) {
    String sql = "SELECT * FROM Product";
    try (Statement stmt = conn.createStatement();
         ResultSet rs = stmt.executeQuery(sql)) {

        System.out.println("\n-- Product List --");
        while (rs.next()) {
            int id = rs.getInt("ProductID");
            String name = rs.getString("ProductName");
            double price = rs.getDouble("Price");
            int qty = rs.getInt("Quantity");
            System.out.printf("ID: %d | Name: %s | Price: %.2f | Quantity: %d%n",
                    id, name, price, qty);
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

// Update
private static void updateProduct(Connection conn, Scanner scanner) {
    System.out.print("Enter Product ID to update: ");
    int id = scanner.nextInt();
    scanner.nextLine();  // consume newline

    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 qty = scanner.nextInt();

    String sql = "UPDATE Product SET ProductName=?, Price=?, Quantity=? WHERE ProductID=?";
    try (PreparedStatement stmt = conn.prepareStatement(sql)) {
        conn.setAutoCommit(false);  // begin transaction

        stmt.setString(1, name);
        stmt.setDouble(2, price);
        stmt.setInt(3, qty);
        stmt.setInt(4, id);
        int rows = stmt.executeUpdate();

        if (rows > 0) {
            conn.commit();
            System.out.println("Product updated successfully.");
        } else {
            conn.rollback();
            System.out.println("Product ID not found. No changes made.");
        }
    } catch (SQLException e) {
        try {
```

```
                conn.rollback();
                System.out.println("Error occurred. Transaction rolled back.");
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
            e.printStackTrace();
        }
    }

    // Delete
    private static void deleteProduct(Connection conn, Scanner scanner) {
        System.out.print("Enter Product ID to delete: ");
        int id = scanner.nextInt();

        String sql = "DELETE FROM Product WHERE ProductID=?";
        try (PreparedStatement stmt = conn.prepareStatement(sql)) {
            conn.setAutoCommit(false);  // begin transaction

            stmt.setInt(1, id);
            int rows = stmt.executeUpdate();

            if (rows > 0) {
                conn.commit();
                System.out.println("Product deleted successfully.");
            } else {
                conn.rollback();
                System.out.println("Product ID not found. No changes made.");
            }
        } catch (SQLException e) {
            try {
                conn.rollback();
                System.out.println("Error occurred. Transaction rolled back.");
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
            e.printStackTrace();
        }
    }
}
```

Output :

```
Connected to database.

====== Product CRUD Menu ======
1. Add Product
2. View Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 1
Enter Product Name: mobile
Enter Price: 20000
Enter Quantity: 5
Product added successfully.

====== Product CRUD Menu ======
1. Add Product
2. View Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 2

-- Product List --
ID: 2 | Name: mobile | Price: 20000.00 | Quantity: 5
```

```
====== Product CRUD Menu ======
1. Add Product
2. View Products
3. Update Product
4. Delete Product
5. Exit
Choose an option: 4
Enter Product ID to delete: 2
Product deleted successfully.
```