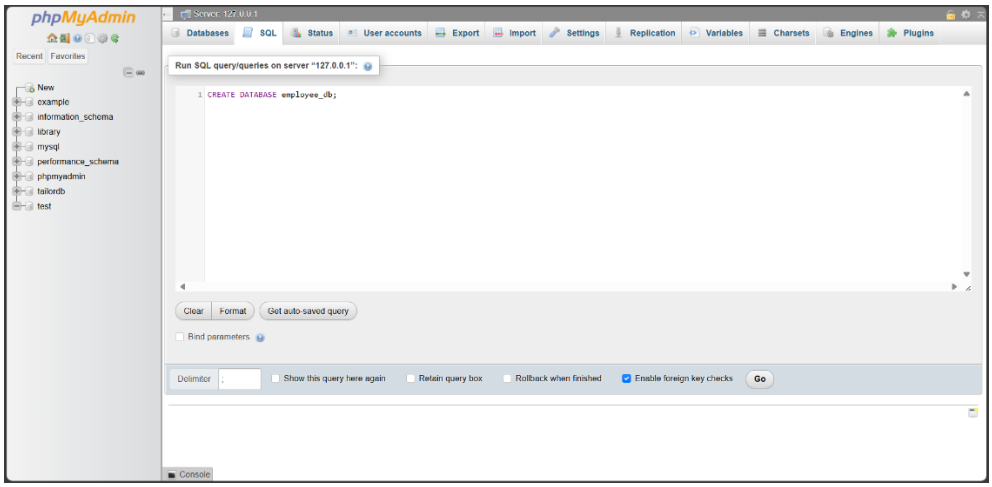
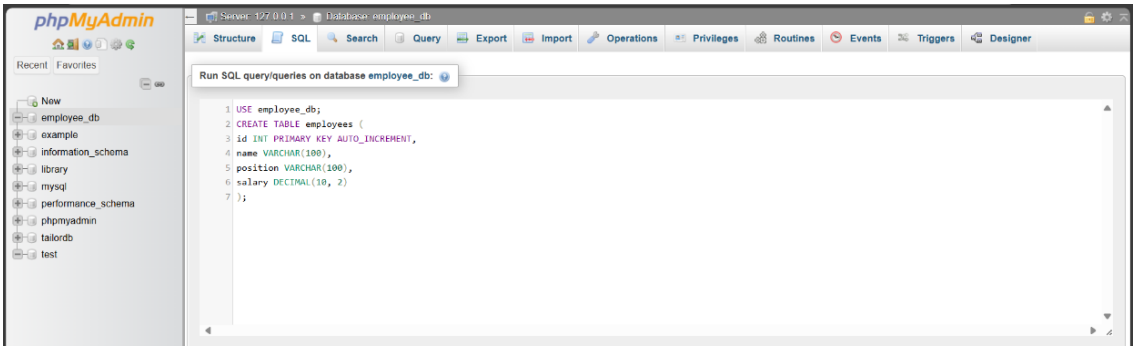
01. JDBC (Java Database Connectivity)

2.1 Set Up MySQL Database

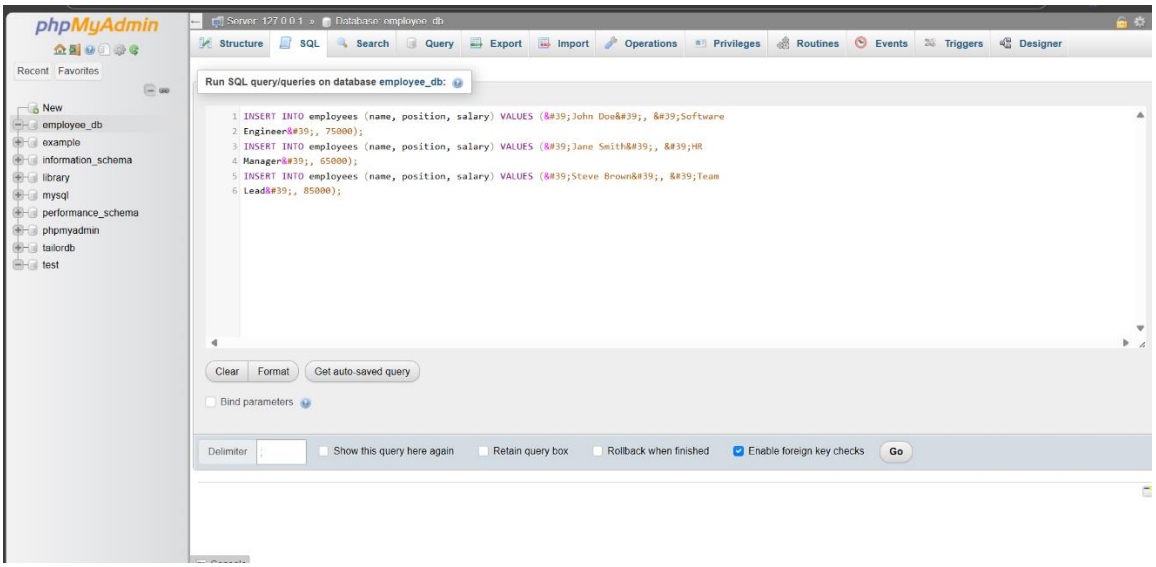
CREATE DATABASE employee_db;

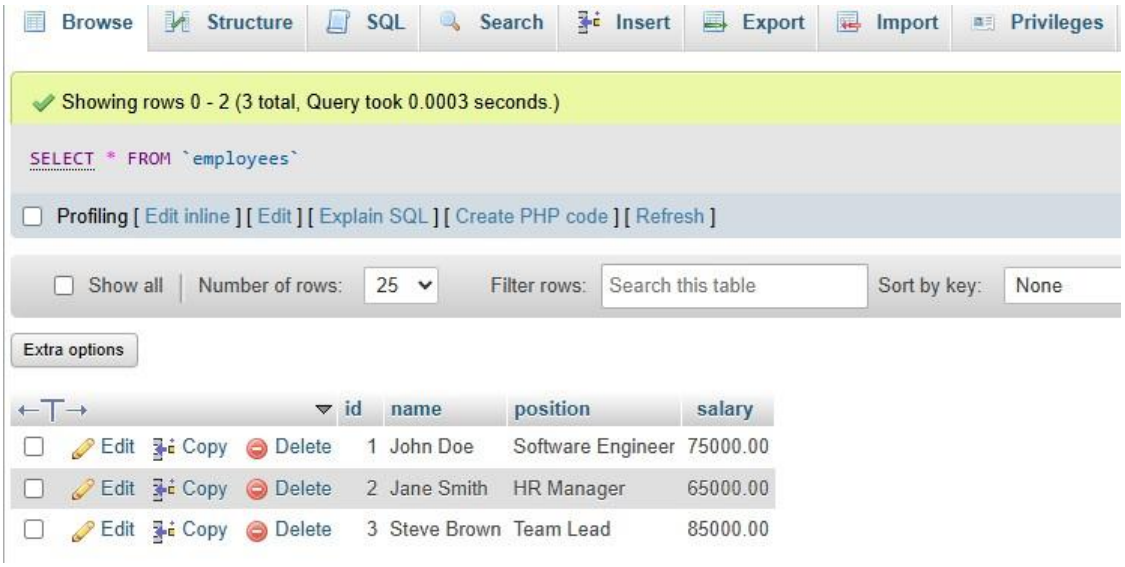


USE employee_db;
CREATE TABLE employees (
id INT PRIMARY KEY AUTO_INCREMENT,
name VARCHAR(100),
position VARCHAR(100),
salary DECIMAL(10, 2)
);



INSERT INTO employees (name, position, salary) VALUES ('John Doe', 'Software
Engineer', 75000);
INSERT INTO employees (name, position, salary) VALUES ('Jane Smith', 'HR
Manager', 65000);
INSERT INTO employees (name, position, salary) VALUES ('Steve Brown', 'Team
Lead', 85000);





2.2 Set Up NetBeans Project

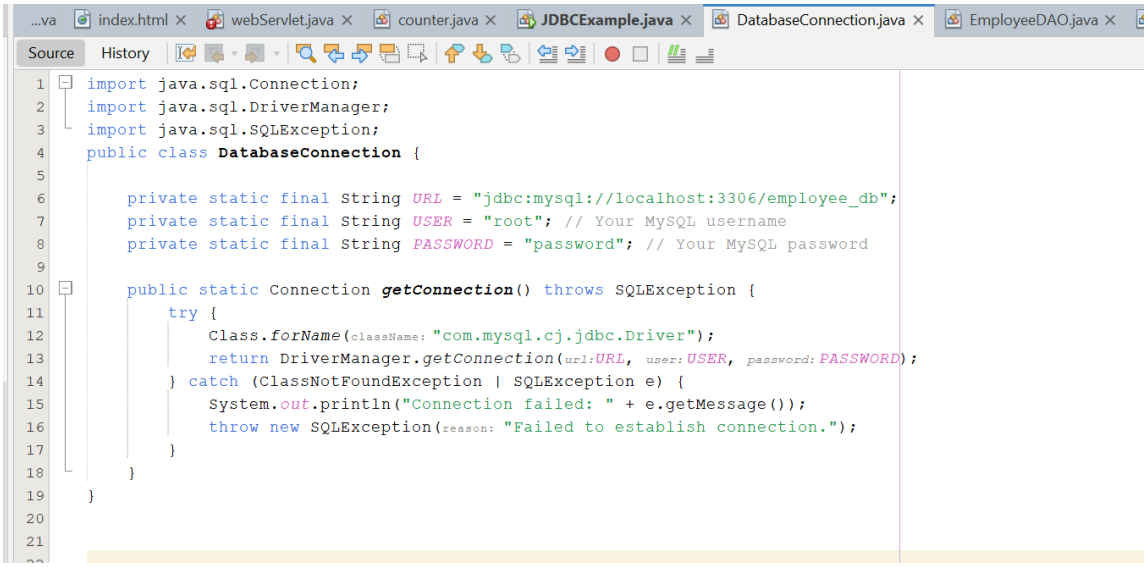
```java
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package jdbcexample;

/**
 *
 * @author hansi
 */
public class JDBCExample {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

2.3 Establish JDBC Connection

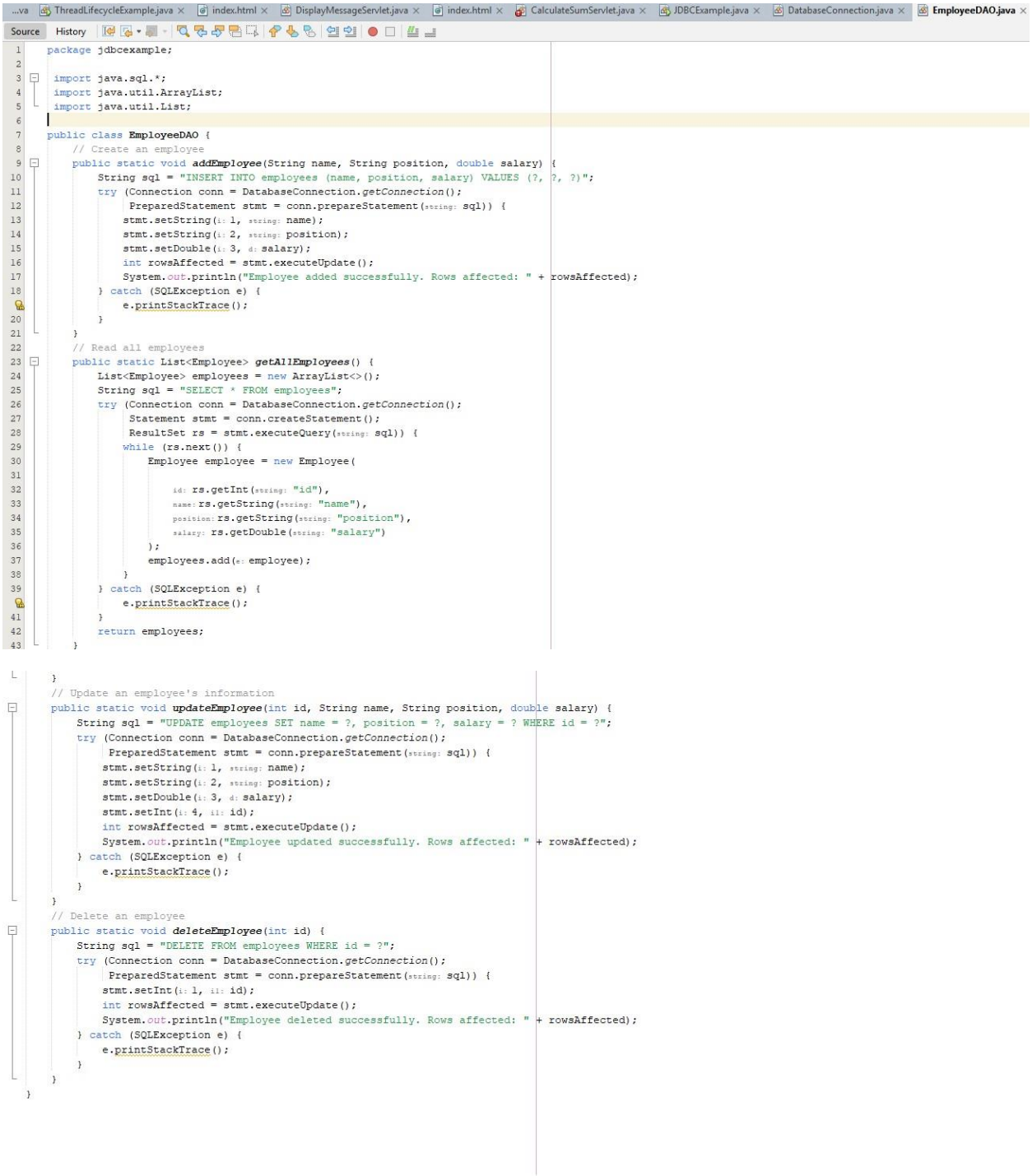


```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DatabaseConnection {

    private static final String URL = "jdbc:mysql://localhost:3306/employee_db";
    private static final String USER = "root"; // Your MySQL username
    private static final String PASSWORD = "password"; // Your MySQL password

    public static Connection getConnection() throws SQLException {
        try {
            Class.forName(className: "com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(url:URL, user:USER, password:PASSWORD);
        } catch (ClassNotFoundException | SQLException e) {
            System.out.println("Connection failed: " + e.getMessage());
            throw new SQLException(reason: "Failed to establish connection.");
        }
    }
}
```

2.4 Perform CRUD Operations

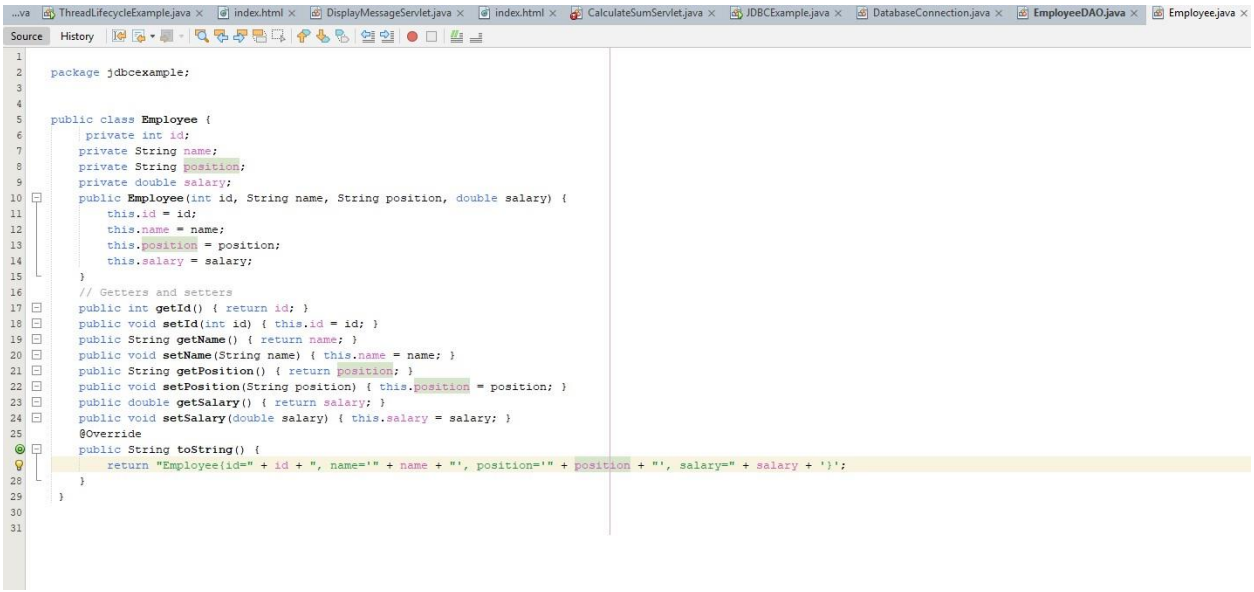


```java
package jdbcexample;

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

public class EmployeeDAO {
    // Create an employee
    public static void addEmployee(String name, String position, double salary) {
        String sql = "INSERT INTO employees (name, position, salary) VALUES (?, ?, ?)";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, name);
            stmt.setString(2, position);
            stmt.setDouble(3, salary);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Employee added successfully. Rows affected: " + rowsAffected);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Read all employees
    public static List<Employee> getAllEmployees() {
        List<Employee> employees = new ArrayList<>();
        String sql = "SELECT * FROM employees";
        try (Connection conn = DatabaseConnection.getConnection();
             Statement stmt = conn.createStatement();
             ResultSet rs = stmt.executeQuery(sql)) {
            while (rs.next()) {
                Employee employee = new Employee(
                    rs.getInt("id"),
                    rs.getString("name"),
                    rs.getString("position"),
                    rs.getDouble("salary")
                );
                employees.add(employee);
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return employees;
    }
    // Update an employee's information
    public static void updateEmployee(int id, String name, String position, double salary) {
        String sql = "UPDATE employees SET name = ?, position = ?, salary = ? WHERE id = ?";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setString(1, name);
            stmt.setString(2, position);
            stmt.setDouble(3, salary);
            stmt.setInt(4, id);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Employee updated successfully. Rows affected: " + rowsAffected);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    // Delete an employee
    public static void deleteEmployee(int id) {
        String sql = "DELETE FROM employees WHERE id = ?";
        try (Connection conn = DatabaseConnection.getConnection();
             PreparedStatement stmt = conn.prepareStatement(sql)) {
            stmt.setInt(1, id);
            int rowsAffected = stmt.executeUpdate();
            System.out.println("Employee deleted successfully. Rows affected: " + rowsAffected);
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

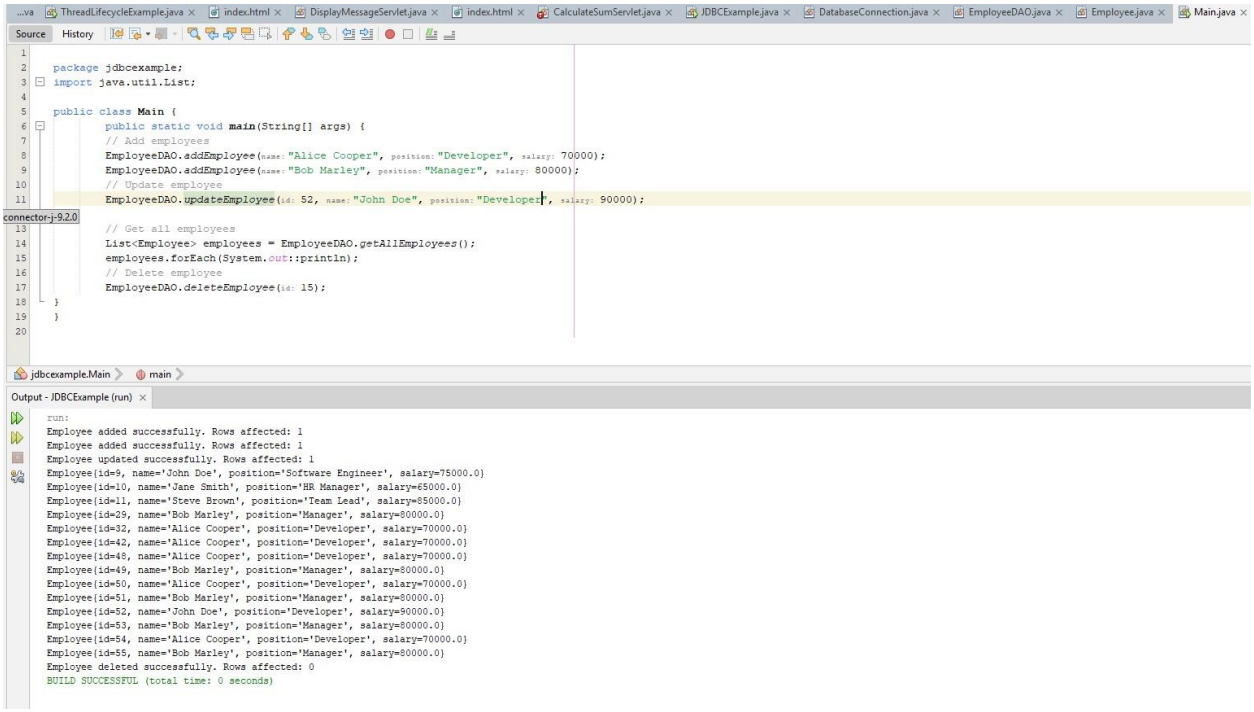
2.5 Create Employee.java Class



```java
package jdbcexample;


public class Employee {
    private int id;
    private String name;
    private String position;
    private double salary;
    public Employee(int id, String name, String position, double salary) {
        this.id = id;
        this.name = name;
        this.position = position;
        this.salary = salary;
    }
    // Getters and setters
    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getPosition() { return position; }
    public void setPosition(String position) { this.position = position; }
    public double getSalary() { return salary; }
    public void setSalary(double salary) { this.salary = salary; }
    @Override
    public String toString() {
        return "Employee{id=" + id + ", name='" + name + "', position='" + position + "', salary=" + salary + '}';
    }
}
```

2.6 Test the Application



```java
package jdbcexample;
import java.util.List;

public class Main {
        public static void main(String[] args) {
        // Add employees
        EmployeeDAO.addEmployee(name: "Alice Cooper", position: "Developer", salary: 70000);
        EmployeeDAO.addEmployee(name: "Bob Marley", position: "Manager", salary: 80000);
        // Update employee
        EmployeeDAO.updateEmployee(id: 52, name: "John Doe", position: "Developer", salary: 90000);

        // Get all employees
        List<Employee> employees = EmployeeDAO.getAllEmployees();
        employees.forEach(System.out::println);
        // Delete employee
        EmployeeDAO.deleteEmployee(id: 15);
    }
}
```

```
run:
Employee added successfully. Rows affected: 1
Employee added successfully. Rows affected: 1
Employee updated successfully. Rows affected: 1
Employee{id=9, name='John Doe', position='Software Engineer', salary=75000.0}
Employee{id=10, name='Jane Smith', position='HR Manager', salary=65000.0}
Employee{id=11, name='Steve Brown', position='Team Lead', salary=85000.0}
Employee{id=29, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=32, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=42, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=48, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=49, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=50, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=51, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=52, name='John Doe', position='Developer', salary=90000.0}
Employee{id=53, name='Bob Marley', position='Manager', salary=80000.0}
Employee{id=54, name='Alice Cooper', position='Developer', salary=70000.0}
Employee{id=55, name='Bob Marley', position='Manager', salary=80000.0}
Employee deleted successfully. Rows affected: 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

2.7 Run the Application