

## JDBS

Lab sheet -03

```
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 some sample data
```

```
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);
```

Code for DatabaseConnection.java:

```
package jdbcexample;
```

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
```

```
import java.sql.SQLException;
```

```
/**
```

```
 *
```

```
 * @author student
```

```
 */
```

```
public class DatabaseConnection {
```

```
    private static final String URL = "jdbc:mysql://localhost:3306/employee_db"; // Database URL
```

```
    private static final String USER = "root";
```

```

private static final String PASSWORD = "";

public static Connection getConnection() throws SQLException {

    try {

        Class.forName("com.mysql.cj.jdbc.Driver");

        return DriverManager.getConnection(URL, USER, PASSWORD);
    }

    catch (ClassNotFoundException | SQLException e) {
        System.out.println("Connection failed:" + e.getMessage());
        throw new SQLException("Failed to establish connection.");
    }
}
}

```

1. Open NetBeans IDE 8.2.

2. Create a new Java application:

- Go to File > New Project.
- Select Java as the project type, and choose Java Application.
- Name your project JDBCExample.
- 3. Add MySQL JDBC Driver to your project:
- Right-click on the project in the Projects pane.
- Select Properties.
- In the Libraries tab, click Add JAR/Folder.
- Navigate to the location of your mysql-connector-java-x.x.xx.jar file and add it.

Code for EmployeeDAO.java:

```

package jdbcexample;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

/**

```

```

* @author student
*/
public class DatabaseConnection {
    private static final String URL ="jdbc:mysql://localhost:3306/employee_db"; // Database URL
    private static final String USER = "root"; // Your MySQL username
    private static final String PASSWORD = ""; // Your MySQL password
    public static Connection getConnection() throws SQLException {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            return DriverManager.getConnection(URL, USER, PASSWORD);
        }
        catch (ClassNotFoundException | SQLException e) {
            System.out.println("Connection failed:" + e.getMessage());
            throw new SQLException("Failed to establish connection.");
        }
    }
}

```

Code for EmployeeDAO.java:

```

package jdbcexample;

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

/**
 *
 * @author student
 */
public class EmployeeDAO {
    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)) {

package jdbcexample;

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

/**
 *
 * @author student
 */
public class EmployeeDAO {

    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 (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);
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();
}
}
}

```

Code for Employee.java:

```

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 + "}";
}
}

```

Code for JDBCExample.java:

```

package jdbcexample;

import java.util.List;

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

    /**
     * @param args the command line arguments

```

\*/

```
public static void main(String[] args) {
```

```
    EmployeeDAO.addEmployee("Alice Cooper", "Developer", 70000);
```

```
    EmployeeDAO.addEmployee("Bob Marley", "Manager", 80000);
```

```
    EmployeeDAO.updateEmployee(1, "John Doe", "Senior Software Engineer", 90000);
```

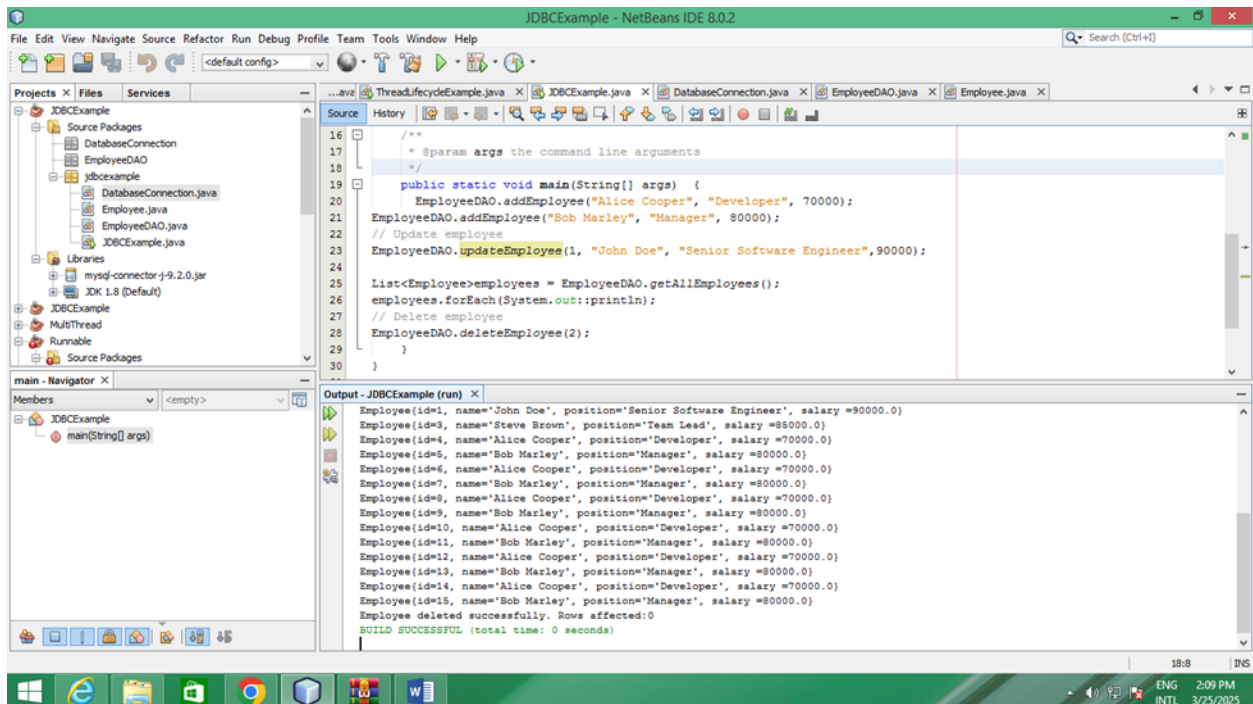
```
    List<Employee>employees = EmployeeDAO.getAllEmployees();
```

```
    employees.forEach(System.out::println);
```

```
    EmployeeDAO.deleteEmployee(2);
```

```
}}
```

## OUT PUT

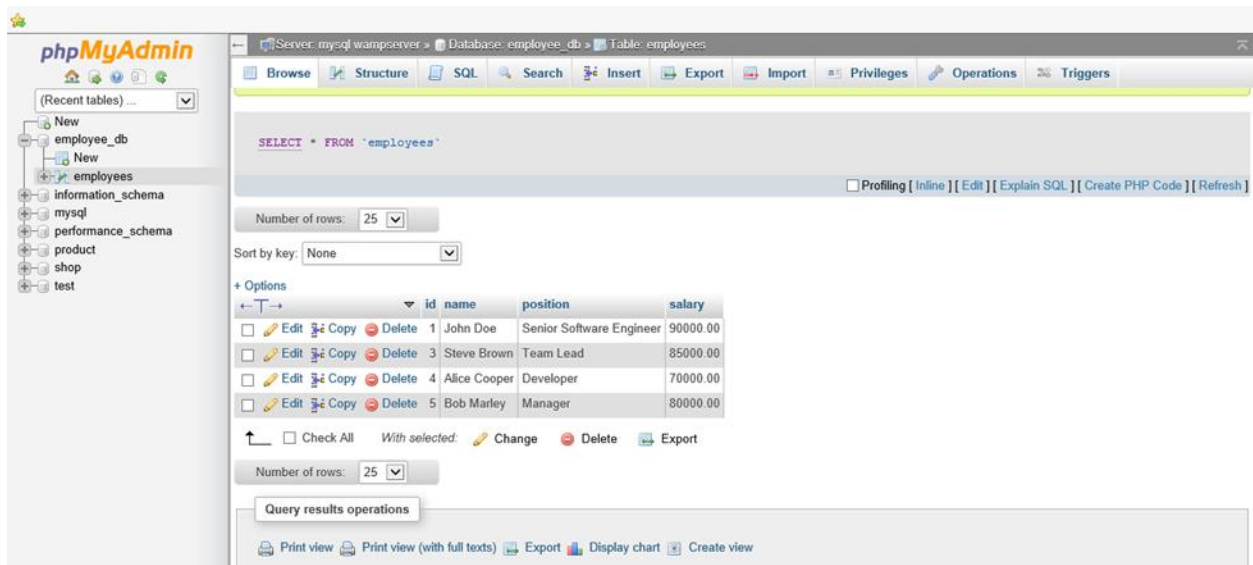


The screenshot displays the NetBeans IDE interface. The main editor shows the `main` method of `EmployeeDAO.java`, which performs several database operations: adding two employees, updating an employee's name and salary, retrieving all employees, and deleting an employee. The `Output` window at the bottom shows the results of these operations, listing 15 employees with their IDs, names, positions, and salaries. The output also indicates that the build was successful and that 0 rows were affected by the delete operation.

```
Employee(id=1, name='John Doe', position='Senior Software Engineer', salary =90000.0)
Employee(id=3, name='Steve Brown', position='Team Lead', salary =85000.0)
Employee(id=4, name='Alice Cooper', position='Developer', salary =70000.0)
Employee(id=5, name='Bob Marley', position='Manager', salary =80000.0)
Employee(id=6, name='Alice Cooper', position='Developer', salary =70000.0)
Employee(id=7, name='Bob Marley', position='Manager', salary =80000.0)
Employee(id=8, name='Alice Cooper', position='Developer', salary =70000.0)
Employee(id=9, name='Bob Marley', position='Manager', salary =80000.0)
Employee(id=10, name='Alice Cooper', position='Developer', salary =70000.0)
Employee(id=11, name='Bob Marley', position='Manager', salary =80000.0)
Employee(id=12, name='Alice Cooper', position='Developer', salary =70000.0)
Employee(id=13, name='Bob Marley', position='Manager', salary =80000.0)
Employee(id=14, name='Alice Cooper', position='Developer', salary =70000.0)
Employee(id=15, name='Bob Marley', position='Manager', salary =80000.0)
Employee deleted successfully. Rows affected:0
BUILD SUCCESSFUL (total time: 0 seconds)
```



## DATABASE UPDATE



Server: mysql wampserver » Database: employee\_db » Table: employees

Browse Structure SQL Search Insert Export Import Privileges Operations Triggers

`SELECT * FROM `employees``

☐ Profiling [\[ Inline \]](#) [\[ Edit \]](#) [\[ Explain SQL \]](#) [\[ Create PHP Code \]](#) [\[ Refresh \]](#)

Number of rows: 25

Sort by key: None

+ Options

	id	name	position	salary
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	1	John Doe	Senior Software Engineer	90000.00
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	3	Steve Brown	Team Lead	85000.00
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	4	Alice Cooper	Developer	70000.00
<input type="checkbox"/> <a href="#">Edit</a> <a href="#">Copy</a> <a href="#">Delete</a>	5	Bob Marley	Manager	80000.00

☐ Check All With selected: [Change](#) [Delete](#) [Export](#)

Number of rows: 25

Query results operations

[Print view](#) [Print view \(with full texts\)](#) [Export](#) [Display chart](#) [Create view](#)