# JDBC

1) **Design a Java program to create a simple employee management system using JDBC and MySQL Connector/J. The program should allow users to perform the following operations**:

    a) **Add a new employee: The user can enter details like employee ID, name, department, and salary, and the program should add the employee to the database.**
    b) **Update employee details: The user can update the name, department, or salary of an existing employee based on their employee ID.**
    c) **Delete an employee: The user can delete an employee from the database based on their employee ID.**
    d) **Display all employees: The program should retrieve and display a list of all employees and their details from the database.**
    e) **Requirements:**
        i) **Use JDBC and MySQL Connector/J to connect to the MySQL database and perform CRUD (Create, Read, Update, Delete) operations.**
        ii) **Implement exception handling to handle possible errors during database interactions.**
        iii) **Provide a user-friendly console interface for the user to interact with the employee management system.**
        iv) **Cover Java topics such as classes, methods, user input and output (I/O), and exception handling.**
    f) **Note: Before running the program, make sure you have MySQL installed, create a database named "employee_management," and a table named "employees" with columns: "id" (INT, PRIMARY KEY), "name" (VARCHAR), "department" (VARCHAR), and "salary" (DOUBLE).**

**Mysql**

```
CREATE DATABASE employee_management;
USE employee_management;

CREATE TABLE employees (
  id INT PRIMARY KEY,
  name VARCHAR(50),
  department VARCHAR(50),
  salary DOUBLE
);
```

EmployeeManagementSystem.java

```
package com.example.assignment;

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

public class EmployeeManagementSystem {

    // MySQL Database connection details
//    static final String JDBC_URL = "jdbc:mysql://localhost:3306/employee_management";
//    static final String JDBC_USER = "root";  // Replace with your MySQL username
//    static final String JDBC_PASSWORD = "root";  // Replace with your MySQL password

    static String jdbcURL =
"jdbc:mysql://localhost:3306/employee_management?useSSL=false&serverTimezone=UTC";
    static String username = "root";
    static String password = "root";
    Connection connection = DriverManager.getConnection(jdbcURL, username, password);

    public EmployeeManagementSystem() throws SQLException {
    }


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

```
    try (Connection conn = DriverManager.getConnection(jdbcURL, username, password)) {
        System.out.println("Connected to the database!");

        while (true) {
            System.out.println("\nEmployee Management System");
            System.out.println("1. Add Employee");
            System.out.println("2. Update Employee");
            System.out.println("3. Delete Employee");
            System.out.println("4. Display All Employees");
            System.out.println("5. Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            switch (choice) {
                case 1:
                    addEmployee(conn, scanner);
                    break;
                case 2:
                    updateEmployee(conn, scanner);
                    break;
                case 3:
                    deleteEmployee(conn, scanner);
                    break;
                case 4:
                    displayAllEmployees(conn);
                    break;
                case 5:
                    System.out.println("Exiting...");
                    return;
                default:
                    System.out.println("Invalid choice. Try again.");
            }
        }
    } catch (SQLException e) {
        System.out.println("Database connection failed.");
        e.printStackTrace();
    }
}

// Method to add a new employee
private static void addEmployee(Connection conn, Scanner scanner) {
    try {
        System.out.print("Enter employee ID: ");
        int id = scanner.nextInt();
```

```
        scanner.nextLine();  // Consume the newline

        System.out.print("Enter employee name: ");
        String name = scanner.nextLine();

        System.out.print("Enter department: ");
        String department = scanner.nextLine();

        System.out.print("Enter salary: ");
        double salary = scanner.nextDouble();

        String query = "INSERT INTO employees (id, name, department, salary) VALUES (?, ?, ?, ?)";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
            stmt.setInt(1, id);
            stmt.setString(2, name);
            stmt.setString(3, department);
            stmt.setDouble(4, salary);
            stmt.executeUpdate();
            System.out.println("Employee added successfully.");
        }
    } catch (SQLException e) {
        System.out.println("Error adding employee.");
        e.printStackTrace();
    }
}

// Method to update employee details
private static void updateEmployee(Connection conn, Scanner scanner) {
    try {
        System.out.print("Enter employee ID to update: ");
        int id = scanner.nextInt();
        scanner.nextLine();  // Consume the newline

        System.out.print("Enter new name (or leave blank to keep current): ");
        String name = scanner.nextLine();

        System.out.print("Enter new department (or leave blank to keep current): ");
        String department = scanner.nextLine();

        System.out.print("Enter new salary (or enter -1 to keep current): ");
        double salary = scanner.nextDouble();

        StringBuilder query = new StringBuilder("UPDATE employees SET ");
        boolean firstField = true;
```

```
        if (!name.isEmpty()) {
            query.append("name = ?");
            firstField = false;
        }
        if (!department.isEmpty()) {
            if (!firstField) query.append(", ");
            query.append("department = ?");
            firstField = false;
        }
        if (salary != -1) {
            if (!firstField) query.append(", ");
            query.append("salary = ?");
        }
        query.append(" WHERE id = ?");

        try (PreparedStatement stmt = conn.prepareStatement(query.toString())) {
            int paramIndex = 1;
            if (!name.isEmpty()) stmt.setString(paramIndex++, name);
            if (!department.isEmpty()) stmt.setString(paramIndex++, department);
            if (salary != -1) stmt.setDouble(paramIndex++, salary);
            stmt.setInt(paramIndex, id);

            int rowsUpdated = stmt.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Employee updated successfully.");
            } else {
                System.out.println("Employee not found.");
            }
        }
    } catch (SQLException e) {
        System.out.println("Error updating employee.");
        e.printStackTrace();
    }
}

// Method to delete an employee
private static void deleteEmployee(Connection conn, Scanner scanner) {
    try {
        System.out.print("Enter employee ID to delete: ");
        int id = scanner.nextInt();

        String query = "DELETE FROM employees WHERE id = ?";
        try (PreparedStatement stmt = conn.prepareStatement(query)) {
```

```
            stmt.setInt(1, id);
            int rowsDeleted = stmt.executeUpdate();
            if (rowsDeleted > 0) {
                System.out.println("Employee deleted successfully.");
            } else {
                System.out.println("Employee not found.");
            }
        }
    } catch (SQLException e) {
        System.out.println("Error deleting employee.");
        e.printStackTrace();
    }
}

// Method to display all employees
private static void displayAllEmployees(Connection conn) {
    String query = "SELECT * FROM employees";
    try (Statement stmt = conn.createStatement(); ResultSet rs = stmt.executeQuery(query)) {
        System.out.println("\nEmployee List:");
        while (rs.next()) {
            System.out.println("ID: " + rs.getInt("id") +
                    ", Name: " + rs.getString("name") +
                    ", Department: " + rs.getString("department") +
                    ", Salary: " + rs.getDouble("salary"));
        }
    } catch (SQLException e) {
        System.out.println("Error retrieving employees.");
        e.printStackTrace();
    }
}
}
```

OUTPUT:

```
D:\Users\Bhushan\Downloads\bin\java.exe ...
Connected to the database!

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 1
Enter employee ID: 102
Enter employee name: Tushar Pawar
Enter department: BE IT
Enter salary: 50,000
Employee added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 4

Employee List:
ID: 102, Name: Tushar Pawar, Department: BE IT, Salary: 50000.0
```

```
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 1
Enter employee ID: 107
Enter employee name: Rutuja Patil
Enter department: Cloud Developer
Enter salary: 35000
Employee added successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 4

Employee List:
ID: 101, Name: Harish Ghodke, Department: Comp, Salary: 50000.0
ID: 102, Name: Tushar Pawar, Department: BE IT, Salary: 50000.0
ID: 105, Name: Rutik Tambe, Department: Bsc -DS, Salary: 49999.0
ID: 107, Name: Rutuja Patil, Department: Cloud Developer, Salary: 35000.0
ID: 108, Name: Yash More, Department: Developer, Salary: 80000.0

Employee Management System
```

```
ID: 107, Name: Rutuja Patil, Department: Cloud Developer, Salary: 35000.0
ID: 108, Name: Yash More, Department: Developer, Salary: 80000.0

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 2
Enter employee ID to update: 102
Enter new name (or leave blank to keep current):
Enter new department (or leave blank to keep current): Trainee
Enter new salary (or enter -1 to keep current): 25000
Employee updated successfully.

Employee Management System
1. Add Employee
2. Update Employee
3. Delete Employee
4. Display All Employees
5. Exit
Enter your choice: 4

Employee List:
ID: 101, Name: Harish Ghodke, Department: Comp, Salary: 50000.0
ID: 102, Name: Tushar Pawar, Department: Trainee, Salary: 25000.0
ID: 105, Name: Rutik Tambe, Department: Bsc -DS, Salary: 49999.0
ID: 107, Name: Rutuja Patil, Department: Cloud Developer, Salary: 35000.0
ID: 108, Name: Yash More, Department: Developer, Salary: 80000.0

Employee Management System
1. Add Employee
```

Build completed successfully in 4 sec, 778 ms (56 minutes ago)