

ECLIPSE EXP-5 Question1;

```
package Mysql;
```

```
import java.sql.*;
```

```
import java.util.Scanner;
```

```
public class Mydb {  
    private static final String URL = "jdbc:mysql://localhost:3306/Employee_rec";  
    private static final String USER = "root";  
    private static final String PASSWORD = "Mysql123";  
    private Connection connection;  
    private Scanner scanner;  
  
    // Constructor: Connect to Database  
    public Mydb() {  
        try {  
            Class.forName("com.mysql.cj.jdbc.Driver");  
            connection = DriverManager.getConnection(URL, USER, PASSWORD);  
            System.out.println("Database Connected Successfully!");  
        } catch (ClassNotFoundException e) {  
            System.err.println("JDBC Driver not found! Add MySQL Connector JAR.");  
            e.printStackTrace();  
        } catch (SQLException e) {  
            System.err.println("Database Connection Failed! Check Credentials.");  
            e.printStackTrace();  
        }  
        scanner = new Scanner(System.in);  
    }  
  
    // Fetch all employees  
    public void getAllEmployees() {  
        String query = "SELECT * FROM Employees"; // Fixed table name  
        try (Statement stmt = connection.createStatement();  
             ResultSet rs = stmt.executeQuery(query)) {  
            System.out.println("\n Employee List:");  
            if (!rs.isBeforeFirst()) {  
                System.out.println("No employees found!");  
                return;  
            }  
            while (rs.next()) {  
                System.out.println(" ID: " + rs.getInt("id") +  
                                   " | Name: " + rs.getString("name") +  
                                   " | Salary: " + rs.getDouble("salary"));  
            }  
        } catch (SQLException e) {  
            System.err.println("Error fetching employees: " + e.getMessage());  
            e.printStackTrace();  
        }  
    }  
  
    // Insert an employee  
    public void addEmployee(String name, double salary) {
```

```

        String query = "INSERT INTO Employees (name, salary) VALUES (?, ?)";
        try (PreparedStatement stmt = connection.prepareStatement(query)) {
            stmt.setString(1, name);
            stmt.setDouble(2, salary);
            stmt.executeUpdate();
            System.out.println(" Employee added successfully!");
        } catch (SQLException e) {
            System.err.println(" Error adding employee: " + e.getMessage());
            e.printStackTrace();
        }
    }

    // Remove duplicate employees based on name and salary
    public void removeDuplicateEmployees() {
        String query = "DELETE e1 FROM Employees e1 " +
            "INNER JOIN Employees e2 " +
            "ON e1.name = e2.name AND e1.salary = e2.salary " +
            "WHERE e1.id > e2.id";
        try (Statement stmt = connection.createStatement()) {
            int rowsDeleted = stmt.executeUpdate(query);
            System.out.println(" + rowsDeleted + " duplicate records removed.");
        } catch (SQLException e) {
            System.err.println(" Error removing duplicates: " + e.getMessage());
            e.printStackTrace();
        }
    }

    // Close connection
    public void closeConnection() {
        if (connection != null) {
            try {
                connection.close();
                System.out.println(" Database Connection Closed.");
            } catch (SQLException e) {
                System.err.println(" Error closing connection: " + e.getMessage());
                e.printStackTrace();
            }
        }
    }

    // Start the menu-driven system
    public void startMenu() {
        while (true) {
            System.out.println("\n===== Employee Management System =====");
            System.out.println("1 Add Employee");
            System.out.println("2 Display All Employees");
            System.out.println("3 Remove Duplicate Employees");
            System.out.println("4 Exit");
            System.out.print(" Choose an option: ");
            int choice = scanner.nextInt();
            scanner.nextLine(); // Consume the newline

            switch (choice) {
                case 1:
                    addEmployeeMenu();

```

```

        break;
    case 2:
        getAllEmployees();
        break;
    case 3:
        removeDuplicateEmployees();
        break;
    case 4:
        System.out.println(" Exiting Employee Management System...");
        closeConnection();
        scanner.close();
        System.exit(0);
        break;
    default:
        System.out.println(" Invalid choice! Please enter 1, 2, 3, or
4.");
    }
}

// Menu for Adding Employee
private void addEmployeeMenu() {
    System.out.print(" Enter Employee Name: ");
    String name = scanner.nextLine();
    System.out.print(" Enter Employee Salary: ");
    double salary = scanner.nextDouble();
    addEmployee(name, salary);
}

// Main method
public static void main(String[] args) {
    Mydb db = new Mydb();
    db.startMenu();
}
}

```

MY SQL \_Code:

CREATE DATABASE Employee\_rec;

USE Employee\_rec;

CREATE TABLE Employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

name VARCHAR(100) NOT NULL,

salary DECIMAL(10,2) NOT NULL

);

select \* from Employees;

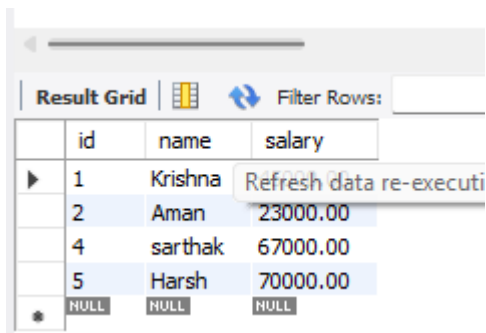
#OUTPUT:

```
Mydb (1) [Java Application] C:\Users\Asus\p2\pool\plugins\org.eclipse
[☒] Database Connected Successfully!

===== Employee Management System =====
1 Add Employee
2 Display All Employees
3 Remove Duplicate Employees
4 Exit
Choose an option: 2

Employee List:
ID: 1 | Name: Krishna | Salary: 45000.0
ID: 2 | Name: Aman | Salary: 23000.0
ID: 4 | Name: sarthak | Salary: 67000.0
ID: 5 | Name: Harsh | Salary: 70000.0

===== Employee Management System =====
1 Add Employee
2 Display All Employees
3 Remove Duplicate Employees
4 Exit
Choose an option:
```



The screenshot shows the Eclipse IDE's 'Result Grid' window. It displays a table with three columns: 'id', 'name', and 'salary'. The table contains five rows of data, with the first four rows highlighted in blue. A tooltip 'Refresh data re-executi' is visible over the 'salary' column. At the bottom of the table, there is a row with three 'NULL' values.

	id	name	salary
▶	1	Krishna	45000.00
	2	Aman	23000.00
	4	sarthak	67000.00
	5	Harsh	70000.00
*	NULL	NULL	NULL

EXP\_5 QUESTION:2

```
package Auto;

import java.util.*;

public class Autoboxing {
    public static void main(String[] args) {
        List<String> numberStrings = Arrays.asList("10", "20", "30", "40", "50");
        List<Integer> numbers = new ArrayList<>();

        for (String numStr : numberStrings) {
            numbers.add(parseToInteger(numStr));
        }
    }
}
```

```
    }

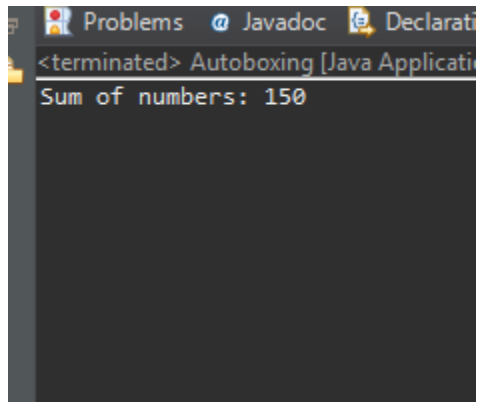
    int sum = calculateSum(numbers);

    System.out.println("Sum of numbers: " + sum);
}

private static Integer parseToInteger(String str) {
    return Integer.parseInt(str);
}

private static int calculateSum(List<Integer> numbers) {
    int sum = 0;
    for (Integer num : numbers) {
        sum += num;
    }
    return sum;
}
}
```

OUTPUT:



The screenshot shows an IDE window with tabs for 'Problems', 'Javadoc', and 'Declarations'. The active tab is 'Problems', which displays the message '<terminated> Autoboxing [Java Application]'. Below this, the output of the program is shown as 'Sum of numbers: 150'.