Roll no : 122

Practical No: 9

Aim: Assignment based Spring JDBC

Q.1 Write a program to insert, update and delete records from the given table.

Student.java

Code:

```
package experiment8;
public class Student {
  private int id;
  private String name;
  private int age;
  // parameterized constructor
  public Student(int id, String name, int age) {
    this.id = id;
    this.name = name;
    this.age = age;
  }
  // 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 int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    // method to display data
    public void show() {
        System.out.println("Id = " + id + ", Name = " + name + ", Age = " + age);
    }
}
```

StudentDao.java:

Code:

```
package experiment8;
import experiment8.Student;
public interface StudentDao {
    public int save(Student s) throws Exception;
    public int update(Student s) throws Exception;
    public int delete(int id) throws Exception;
}
```

Studentdaoimpl.java

Code:

```
package experiment8;

import java.sql.Connection;
import java.sql.PreparedStatement;
import javax.sql.DataSource;
import experiment8.Student;
import experiment8.StudentDao;
public class StudentDaoImpl implements StudentDao {

    private DataSource ds;
    private Connection con;

    public void setDs(DataSource ds) {
        this.ds = ds;
    }
    @Override
    public int save(Student s) throws Exception {
        con = ds.getConnection();
        String sql = "insert into student values(?,?,?)";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setInt(1, s.getId());
        ps.setString(2, s.getName());
        ps.setInt(3, s.getAge());

        return ps.executeUpdate();
    }
    @Override
    public int update(Student s) throws Exception {
        con = ds.getConnection();
        String sql = "update student set name=?, age=? where id=?";
        PreparedStatement ps = con.prepareStatement(sql);
        ps.setString(1, s.getName());
        ps.setInt(2, s.getAge());
        ps.setInt(3, s.getId());
        return ps.executeUpdate();
    }
    @Override
    public int delete(int id) throws Exception {
        con = ds.getConnection();
        String sql = "delete from student where id=?";
        PreparedStatement ps = con.prepareStatement(sql);

        ps.setInt(1, id);

        return ps.executeUpdate();
    }
}
```

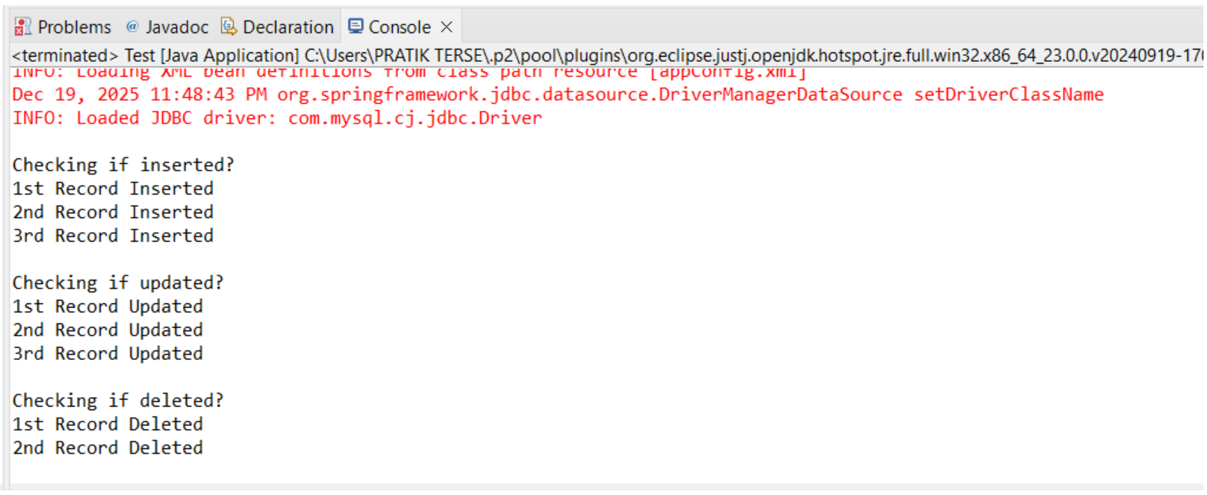
appConfig.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bds"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/student" />
        <property name="username" value="root" />
```

```
      <property name="password" value="Shubham@29" />
   </bean>


   <bean id="dao" class="experiment8.StudentDaoImpl">
      <property name="ds" ref="bds" />
   </bean>


</beans>
```

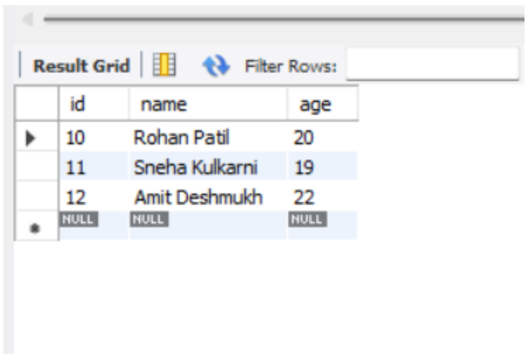
Code:

Output:

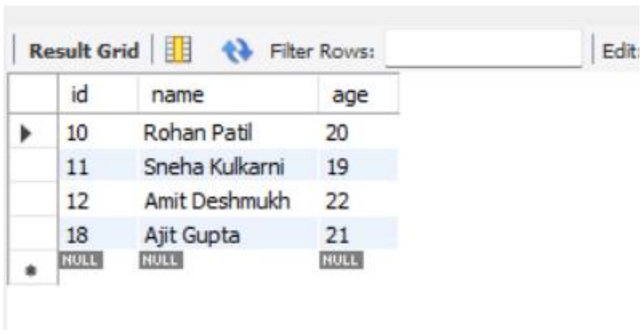


```
<terminated> Test [Java Application] C:\Users\PRATIK TERSE\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-17
INFO: Loading XML bean definitions from class path resource [appConfig.xml]
Dec 19, 2025 11:48:43 PM org.springframework.jdbc.datasource.DriverManagerDataSource setDriverClassName
INFO: Loaded JDBC driver: com.mysql.cj.jdbc.Driver

Checking if inserted?
1st Record Inserted
2nd Record Inserted
3rd Record Inserted

Checking if updated?
1st Record Updated
2nd Record Updated
3rd Record Updated

Checking if deleted?
1st Record Deleted
2nd Record Deleted
```

OUTPUT: Before and after insertion of data



id	name	age
10	Rohan Patil	20
11	Sneha Kulkarni	19
12	Amit Deshmukh	22
NULL	NULL	NULL



id	name	age
10	Rohan Patil	20
11	Sneha Kulkarni	19
12	Amit Deshmukh	22
18	Ajit Gupta	21
NULL	NULL	NULL

2. Write a program to demonstrate Prepared Statement in Spring Jdbc Template.

Employee.java

Code:

```
package spring;
public class Employee {
  private int id;
  private String name;
  private float salary;
  public Employee(int id, String name, float salary) {
    this.id = id;
    this.name = name;
    this.salary = salary;
  }
  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 float getSalary() {
    return salary;
  }
```

```
    public void setSalary(float salary) {
        this.salary = salary;
    }
    public void show() {
        System.out.println("Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]");
    }
}
```

EmployeeDao.java

Code:

```
package spring;
import spring.Employee;
public interface EmployeeDao {
    public int saveEmployee(Employee e);
    public int updateEmployee(Employee e);
}
```

EmployeeDaoImpl.java

Code:

```
package spring;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;
public class EmployeeDaoImpl implements EmployeeDao {
    private JdbcTemplate jdbcTempl;
```

```
public void setJdbcTempl(JdbcTemplate jdbcTempl) {
   this.jdbcTempl = jdbcTempl;
}
@Override
public int saveEmployee(Employee e) {
   String query = "insert into employee values(?,?,?)";
   return jdbcTempl.execute(query, new PreparedStatementCallback<Integer>() {
      @Override
      public Integer doInPreparedStatement(PreparedStatement ps)
            throws SQLException, DataAccessException {
         ps.setInt(1, e.getId());
         ps.setString(2, e.getName());
         ps.setFloat(3, e.getSalary());
         return ps.executeUpdate();
      }
   });
}
@Override
public int updateEmployee(Employee e) {
   String query = "update employee set name=?, salary=? where id=?";
   return jdbcTempl.execute(query, new PreparedStatementCallback<Integer>() {
      @Override
      public Integer doInPreparedStatement(PreparedStatement ps)
            throws SQLException, DataAccessException {
         ps.setString(1, e.getName());
         ps.setFloat(2, e.getSalary());
         ps.setInt(3, e.getId());
         return ps.executeUpdate();
      }
```

```
        });
    }
}
```

applicationContext.xml

Code:

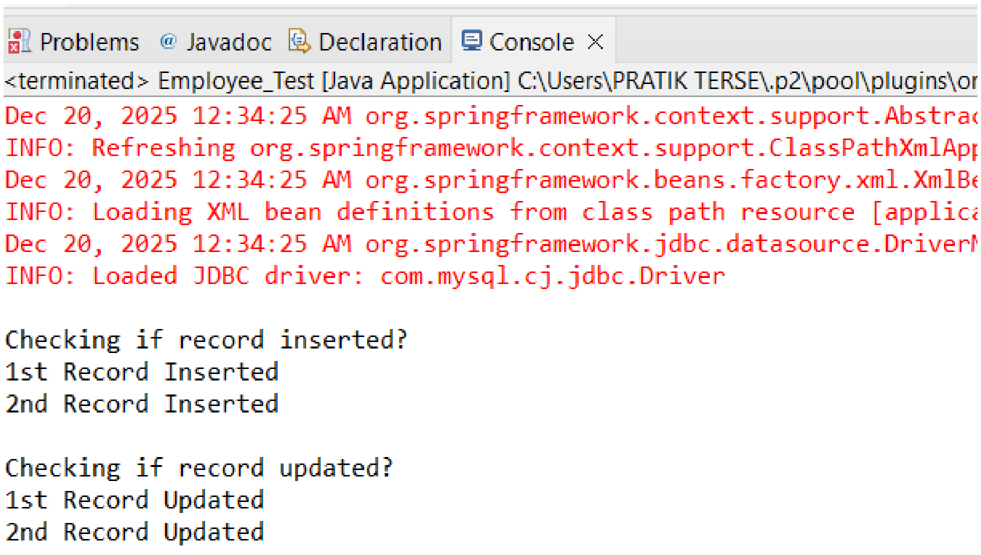
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">
    <bean id="ds"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/employee" />
        <property name="username" value="root" />
        <property name="password" value="Shubham@29" />
    </bean>
    <bean id="jdbcTempl"
          class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="ds" />
    </bean>
    <bean id="edao" class="spring.EmployeeDaoImpl">
        <property name="jdbcTempl" ref="jdbcTempl" />
    </bean>
</beans>
```

Employee_Test.java

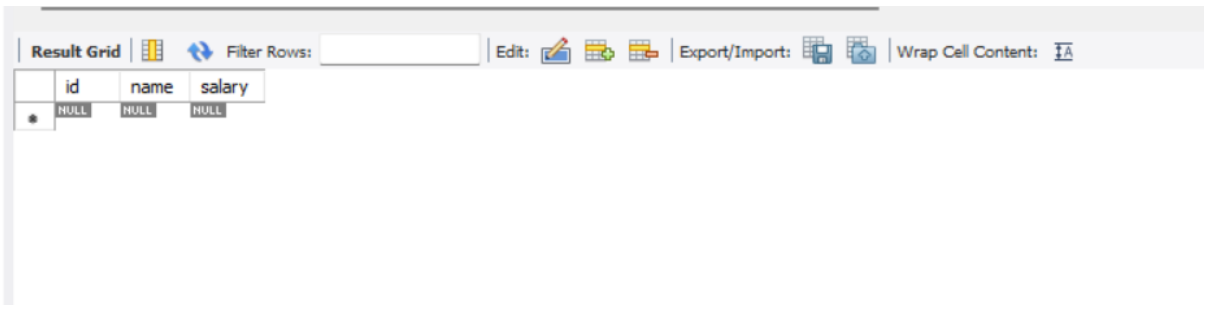
Code:

```
package spring;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Employee_Test {
   public static void main(String[] args) throws Exception {
      // Loading bean configuration file
      ApplicationContext ctx =
            new ClassPathXmlApplicationContext("applicationContext.xml");
      EmployeeDao dao = (EmployeeDao) ctx.getBean("edao");
      // Inserting records
      int insert1 = dao.saveEmployee(new Employee(101, "Deepak", 45000));
      int insert2 = dao.saveEmployee(new Employee(102, "Rahul", 35000));
      System.out.println("\nChecking if record inserted?");
      System.out.println((insert1 > 0) ? "1st Record Inserted" : "1st Record Not Inserted");
      System.out.println((insert2 > 0) ? "2nd Record Inserted" : "2nd Record Not Inserted");
      // Updating records
      int update1 = dao.updateEmployee(new Employee(101, "Divya Mhatre", 45000));
      int update2 = dao.updateEmployee(new Employee(102, "Poonam Naik", 35000));
      System.out.println("\nChecking if record updated?");
      System.out.println((update1 > 0) ? "1st Record Updated" : "1st Record Not Updated");
      System.out.println((update2 > 0) ? "2nd Record Updated" : "2nd Record Not Updated");
   }
}
```

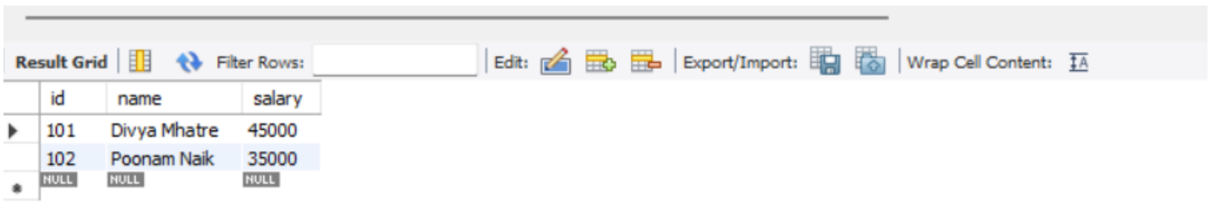
Output:



```
<terminated> Employee_Test [Java Application] C:\Users\PRATIK TERSE\.p2\pool\plugins\or
Dec 20, 2025 12:34:25 AM org.springframework.context.support.Abstrac
INFO: Refreshing org.springframework.context.support.ClassPathXmlApp
Dec 20, 2025 12:34:25 AM org.springframework.beans.factory.xml.XmlBe
INFO: Loading XML bean definitions from class path resource [applica
Dec 20, 2025 12:34:25 AM org.springframework.jdbc.datasource.DriverM
INFO: Loaded JDBC driver: com.mysql.cj.jdbc.Driver

Checking if record inserted?
1st Record Inserted
2nd Record Inserted

Checking if record updated?
1st Record Updated
2nd Record Updated
```



	id	name	salary
*	NULL	NULL	NULL



	id	name	salary
▶	101	Divya Mhatre	45000
	102	Poonam Naik	35000
*	NULL	NULL	NULL

3. Write a program in Spring JDBC to demonstrate ResultSetExtractor Interface

Model Class

Student.java

Code:

```
package com.mca;
public class Student {
   private int id;
   private String name;
   private int age;
   // 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 int getAge() { return age; }
   public void setAge(int age) { this.age = age; }
}
```

StudentRowExtractor.java

Code:

```
package com.mca;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.springframework.jdbc.core.ResultSetExtractor;
public class StudentRowExtractor implements ResultSetExtractor<List<Student>> {
   @Override
   public List<Student> extractData(ResultSet rs) throws SQLException {
        List<Student> list = new ArrayList<>();
        while (rs.next()) {
            Student s = new Student();
            s.setId(rs.getInt("id"));
            s.setName(rs.getString("name"));
            s.setAge(rs.getInt("age"));
            list.add(s);
        }
        return list;
    }
}
```

StudentDao.java

Code:

```
package com.mca;
import java.util.List;
import org.springframework.jdbc.core.JdbcTemplate;
public class StudentDao {
    private JdbcTemplate jdbcTemplate;
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }
    public List<Student> getAllStudents() {
        String sql = "SELECT * FROM student";
        return jdbcTemplate.query(sql, new StudentRowExtractor());
    }
}
```

applicationContext.xml

Code:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!-- DataSource -->
    <bean id="dataSource"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/springjdbc"/>
        <property name="username" value="root"/>
        <property name="password">
            <value><![CDATA[Shubham@29]]></value>
        </property>
    </bean>
    <!-- JdbcTemplate -->
    <bean id="jdbcTemplate"
          class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource"/>
    </bean>
    <!-- DAO -->
    <bean id="studentDao" class="com.mca.StudentDao">
        <property name="jdbcTemplate" ref="jdbcTemplate"/>
    </bean>
</beans>
```

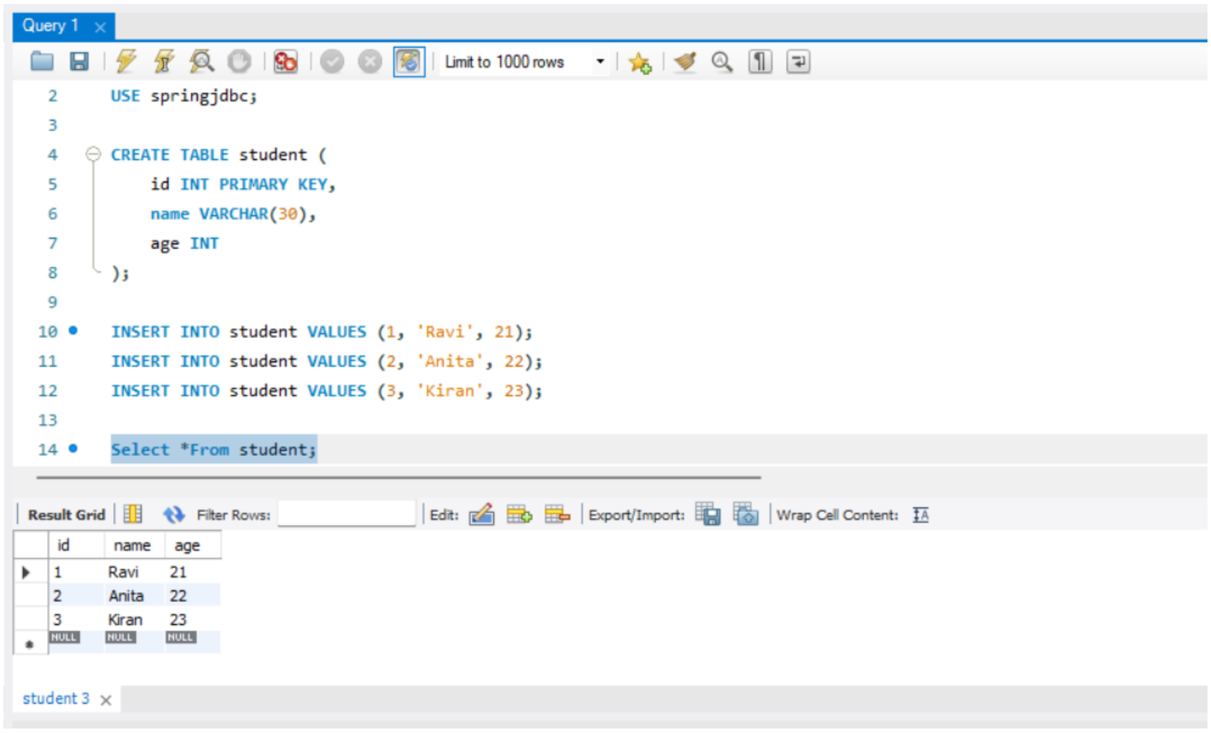
MainApp.java

Code:

```
package com.mca;

import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

  public static void main(String[] args) {

    ApplicationContext context =
        new ClassPathXmlApplicationContext("applicationContext.xml");

    StudentDao dao = context.getBean("studentDao", StudentDao.class);

    List<Student> students = dao.getAllStudents();

    for (Student s : students) {
      System.out.println(
        s.getId() + " " + s.getName() + " " + s.getAge());
    }
  }
}
```

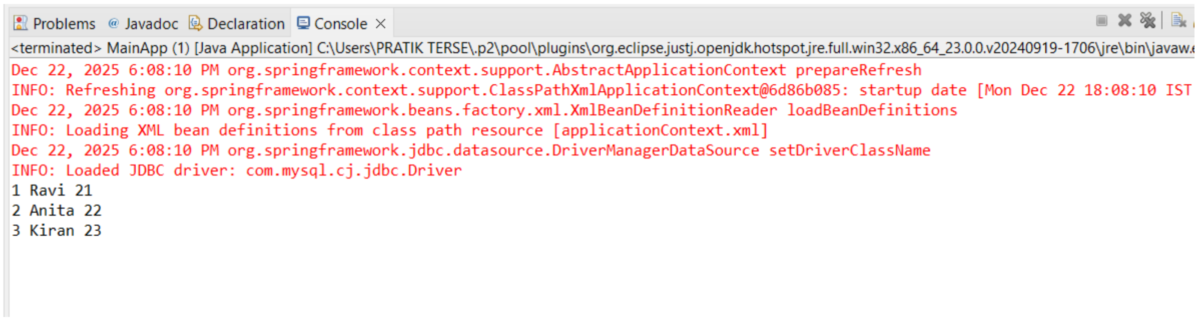
Roll no : 122

Output:



```
USE springjdbc;

CREATE TABLE student (
    id INT PRIMARY KEY,
    name VARCHAR(30),
    age INT
);

INSERT INTO student VALUES (1, 'Ravi', 21);
INSERT INTO student VALUES (2, 'Anita', 22);
INSERT INTO student VALUES (3, 'Kiran', 23);

Select *From student;
```

id	name	age
1	Ravi	21
2	Anita	22
3	Kiran	23
NULL	NULL	NULL



```
<terminated> MainApp (1) [Java Application] C:\Users\PRATIK TERSE\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706\jre\bin\javaw.
Dec 22, 2025 6:08:10 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@6d86b085: startup date [Mon Dec 22 18:08:10 IST
Dec 22, 2025 6:08:10 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path resource [applicationContext.xml]
Dec 22, 2025 6:08:10 PM org.springframework.jdbc.datasource.DriverManagerDataSource setDriverClassName
INFO: Loaded JDBC driver: com.mysql.cj.jdbc.Driver
1 Ravi 21
2 Anita 22
3 Kiran 23
```

4. Write a program to demonstrate RowMapper interface to fetch the records from the database.

Student.java (Model Class)

Code:

```
package com.mca;
public class Student {
    private int id;
    private String name;
    private int age;
    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 int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

StudentRowMapper.java (RowMapper Implementation)

Code:

```
package com.mca;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.RowMapper;
public class StudentRowMapper implements RowMapper<Student> {
   @Override
   public Student mapRow(ResultSet rs, int rowNum) throws SQLException {
      Student s = new Student();
      s.setId(rs.getInt("id"));
      s.setName(rs.getString("name"));
      s.setAge(rs.getInt("age"));
      return s;
   }
}
```

StudentDao.java (DAO Class)

Code:

```
package com.mca;
import java.util.List;
import org.springframework.jdbc.core.JdbcTemplate;
public class StudentDao {
   private JdbcTemplate jdbcTemplate;
   public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
      this.jdbcTemplate = jdbcTemplate;
   }
   public List<Student> getAllStudents() {
      String sql = "SELECT * FROM student";
```

```
        return jdbcTemplate.query(sql, new StudentRowMapper());
    }
}
```

applicationContext.xml (Spring Configuration)

Code:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
       http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">
    <!-- DataSource -->
    <bean id="dataSource"
          class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/springjdbc"/>
        <property name="username" value="root"/>
        <property name="password" value="Shubham@29"/>
    </bean>
    <!-- JdbcTemplate -->
    <bean id="jdbcTemplate"
          class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource"/>
    </bean>
    <!-- DAO -->
    <bean id="studentDao" class="com.mca.StudentDao">
        <property name="jdbcTemplate" ref="jdbcTemplate"/>
    </bean>
```

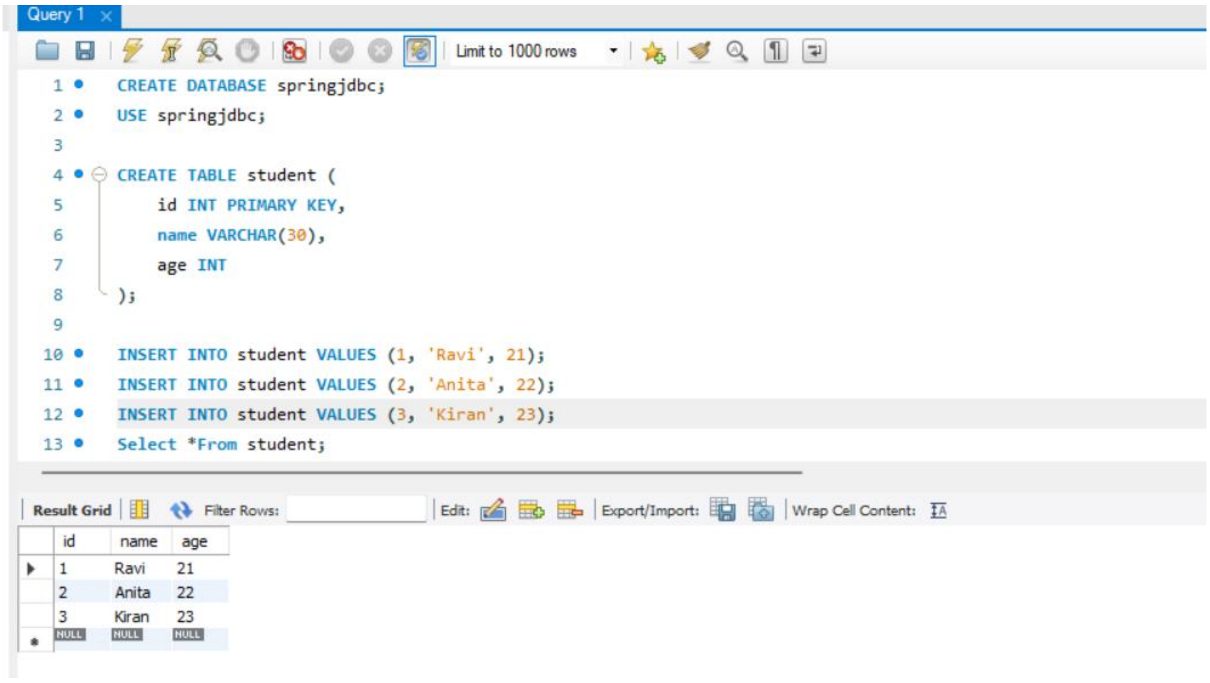
```
</beans>
```

MainApp.java (Test Class)

Code:

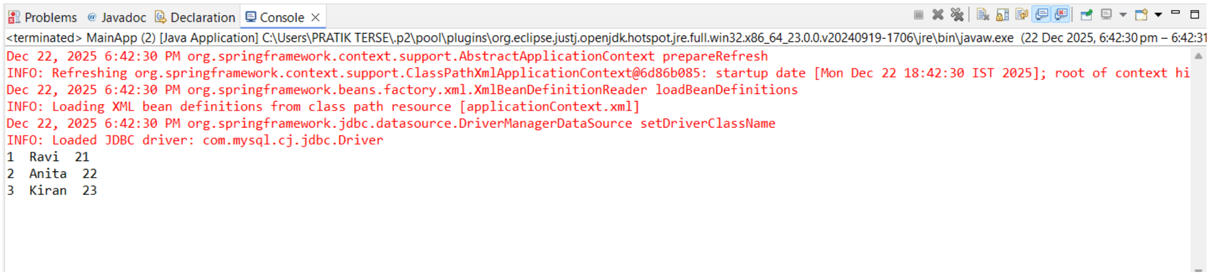
```java
package com.mca;

import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

    public static void main(String[] args) {
        ApplicationContext ctx =
            new ClassPathXmlApplicationContext("applicationContext.xml");

        StudentDao dao = (StudentDao) ctx.getBean("studentDao");

        List<Student> list = dao.getAllStudents();

        for (Student s : list) {
            System.out.println(
                s.getId() + "  " +
                s.getName() + "  " +
                s.getAge()
            );
        }
    }
}
```

Output:



```sql
CREATE DATABASE springjdbc;
USE springjdbc;

CREATE TABLE student (
    id INT PRIMARY KEY,
    name VARCHAR(30),
    age INT
);

INSERT INTO student VALUES (1, 'Ravi', 21);
INSERT INTO student VALUES (2, 'Anita', 22);
INSERT INTO student VALUES (3, 'Kiran', 23);
Select *From student;
```

id	name	age
1	Ravi	21
2	Anita	22
3	Kiran	23
NULL	NULL	NULL



```
<terminated> MainApp (2) [Java Application] C:\Users\PRATIK TERSE\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.0.v20240919-1706\jre\bin\javaw.exe  (22 Dec 2025, 6:42:30 pm – 6:42:31
Dec 22, 2025 6:42:30 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@6d86b085: startup date [Mon Dec 22 18:42:30 IST 2025]; root of context hi
Dec 22, 2025 6:42:30 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path resource [applicationContext.xml]
Dec 22, 2025 6:42:30 PM org.springframework.jdbc.datasource.DriverManagerDataSource setDriverClassName
INFO: Loaded JDBC driver: com.mysql.cj.jdbc.Driver
1  Ravi  21
2  Anita  22
3  Kiran  23
```