

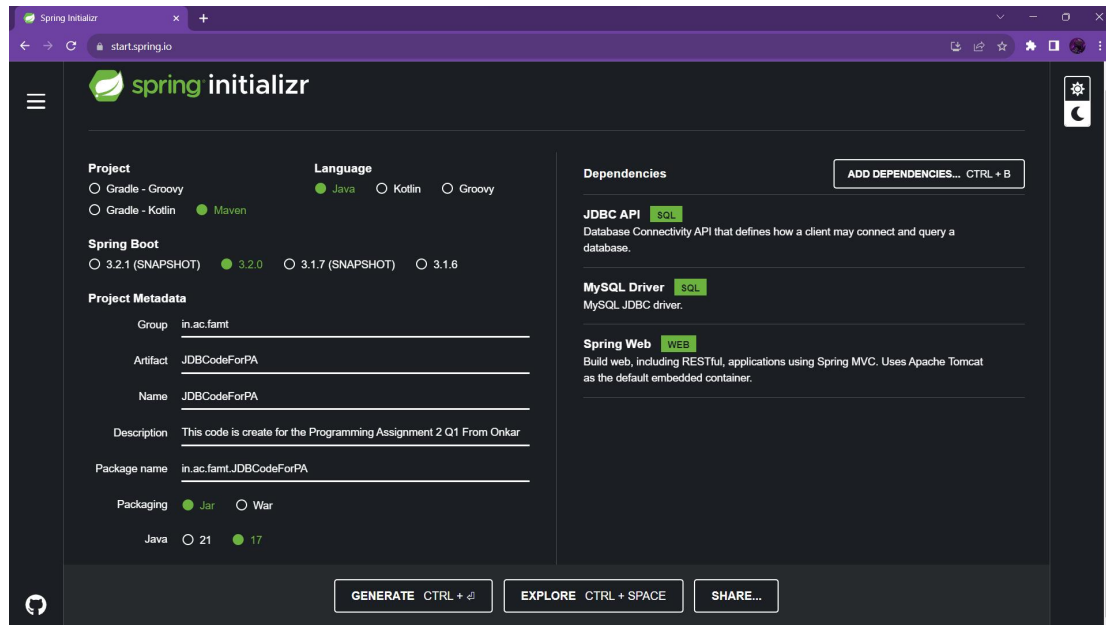
Programming Assignment No. 2

JDBC Data Access with Spring

Q.1. Write a Java application for executing stored procedure that selects particular record from database where POHeader consist of information like poId, supplierNm, podt, poAmt and PODetails table consist of information like poId, srNo, prodCode, quantity, rate.

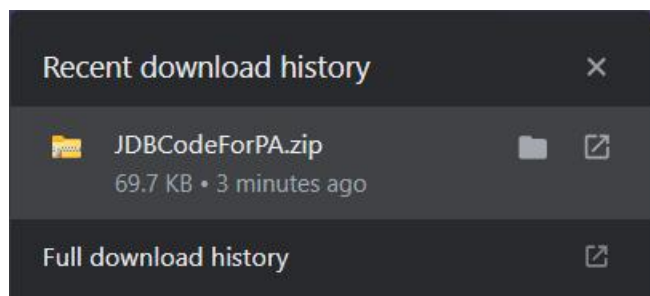
Program 1 :

Step 1: Create Spring Boot Project

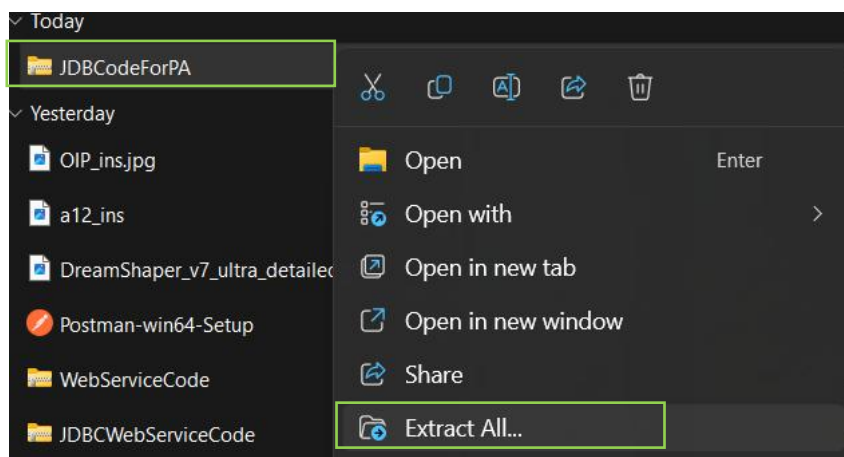


Step 2: Click on the “GENERATE”

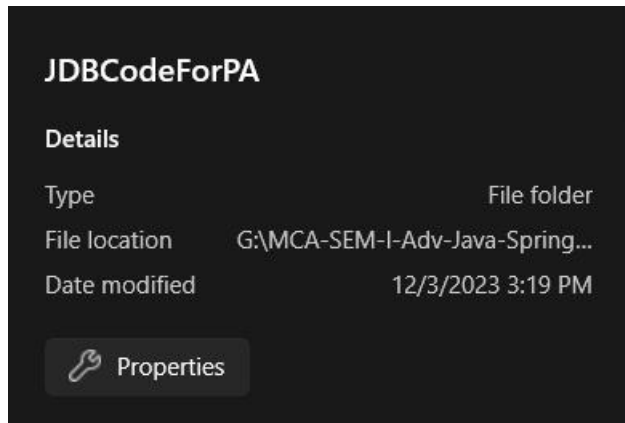
Step 3: It download Zip file from the Browser like given below :



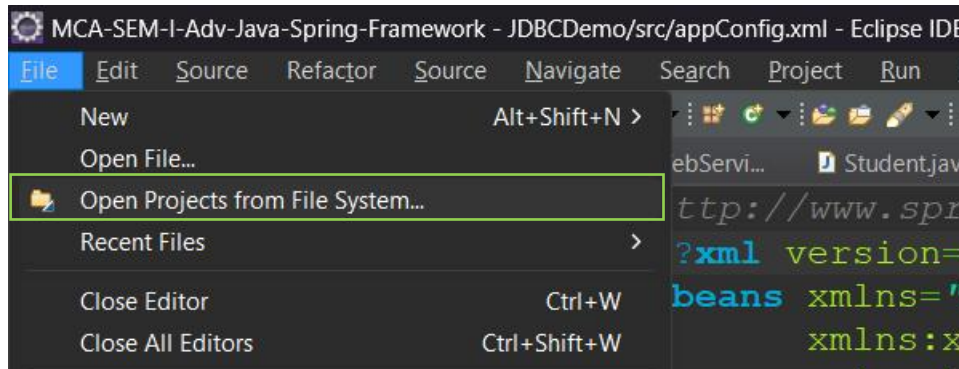
Step 4: Extract File and Open it.



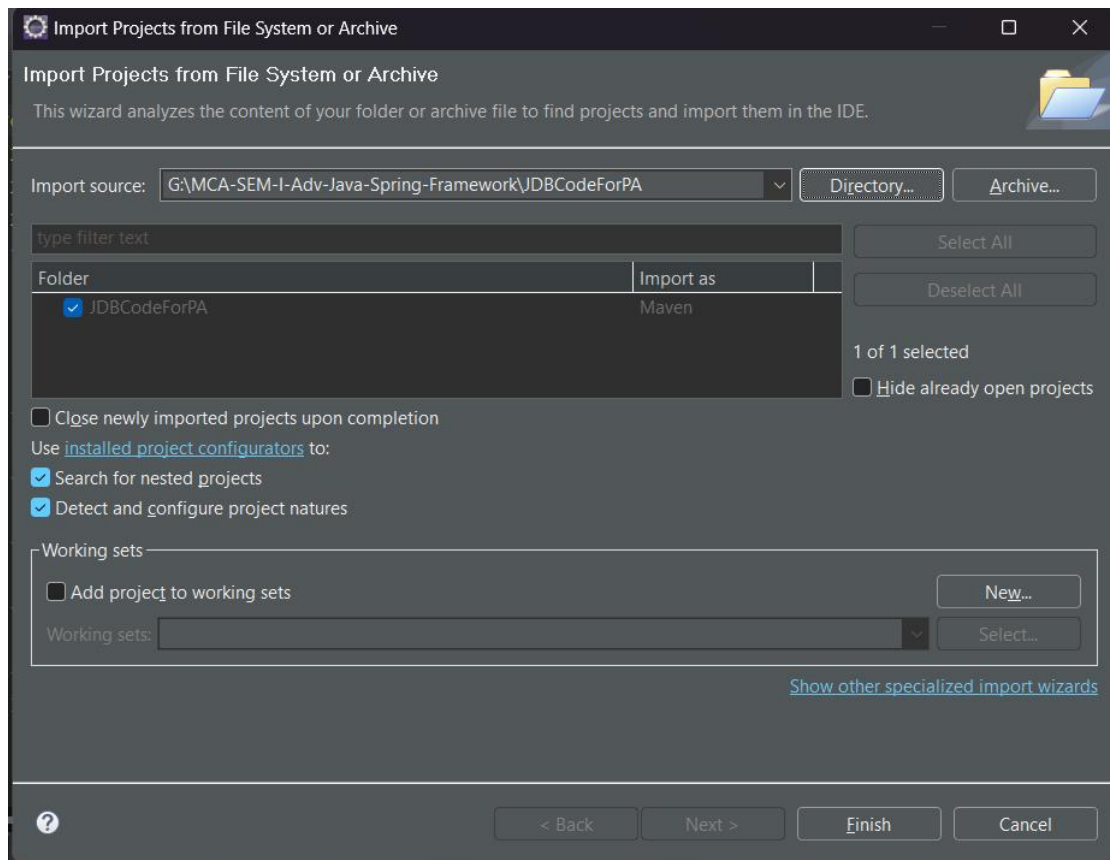
Step 5: Extracted file open and paste to your work-space(eclipse).



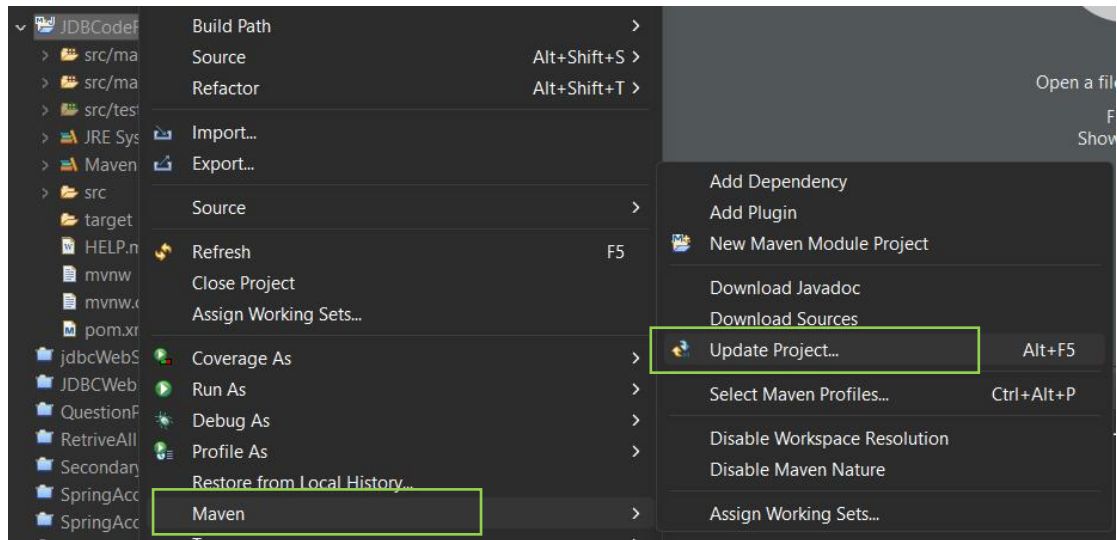
Step 6: Open Eclipse IDE and Click on the Open Projects from File System...



Step 7: Select folder in it and Click on the Finish.

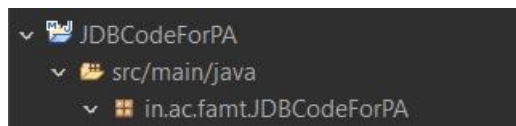


Step 8: Right Click to the Project Follow the given steps:



Step 9 : Your Project is Updated Successfully in the Work-Space.

Open Folder Structure -> Like Given below:



Step 10 : Create Class POHeader.java

```
package in.ac.famt.JDBCCodeForPA;
```

```
import java.util.Date;
```

```
public class POHeader {  
    private int poId;  
    private String supplNm;  
    private Date podt;  
    private int poAmt;  
    public int getPoId() {  
        return poId;  
    }  
    public void setPoId(int poId) {  
        this.poId = poId;  
    }  
    public String getSupplNm() {  
        return supplNm;  
    }  
    public void setSupplNm(String supplNm) {  
        this.supplNm = supplNm;  
    }  
    public Date getPodt() {  
        return podt;  
    }  
    public void setPodt(Date podt) {  
        this.podt = podt;  
    }  
    public int getPoAmt() {  
        return poAmt;  
    }  
    public void setPoAmt(int poAmt) {  
        this.poAmt = poAmt;  
    }  
}
```

```
}  
}
```

Step 10 : Create Class PODetails.java

```
package in.ac.famt.JDBCCodeForPA;  
  
public class PODetails {  
    private int srNo;  
    private String prodCode;  
    private int quantity;  
    private int rate;  
    public int getSrNo() {  
        return srNo;  
    }  
    public void setSrNo(int srNo) {  
        this.srNo = srNo;  
    }  
    public String getProdCode() {  
        return prodCode;  
    }  
    public void setProdCode(String prodCode) {  
        this.prodCode = prodCode;  
    }  
    public int getQuantity() {  
        return quantity;  
    }  
    public void setQuantity(int quantity) {  
        this.quantity = quantity;  
    }  
    public int getRate() {  
        return rate;  
    }  
    public void setRate(int rate) {  
        this.rate = rate;  
    }  
}
```

Step 11 : Create Class POHeaderRowMapper.java

```
package in.ac.famt.JDBCCodeForPA;  
  
import org.springframework.jdbc.core.RowMapper;  
import java.sql.ResultSet;  
import java.sql.SQLException;  
  
public class POHeaderRowMapper implements RowMapper<POHeader> {  
    @Override  
    public POHeader mapRow(ResultSet rs, int rowNum) throws SQLException {  
        POHeader poHeader = new POHeader();  
        poHeader.setPoId(rs.getInt("poId"));  
        poHeader.setSupplNm(rs.getString("supplNm"));  
        poHeader.setPodt(rs.getDate("podt"));  
        poHeader.setPoAmt(rs.getInt("poAmt"));  
        return poHeader;  
    }  
}
```

Step 12 : Create Class PODetailsRowMapper.java

```
package in.ac.famt.JDBCCodeForPA;

import org.springframework.jdbc.core.RowMapper;
import java.sql.ResultSet;
import java.sql.SQLException;

public class PODetailsRowMapper implements RowMapper<PODetails> {
    @Override
    public PODetails mapRow(ResultSet rs, int rowNum) throws SQLException {
        PODetails poDetails = new PODetails();
        poDetails.setSrNo(rs.getInt("srNo"));
        poDetails.setProdCode(rs.getString("prodCode"));
        poDetails.setQuantity(rs.getInt("quantity"));
        poDetails.setRate(rs.getInt("rate"));
        return poDetails;
    }
}
```

Step 13 : Create Class POController.java

```
package in.ac.famt.JDBCCodeForPA;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("/po")
public class POController {

    @Autowired
    private POService poService;

    @GetMapping("/details")
    public List<POHeader> getPODetails() {
        return poService.getPODetails();
    }
}
```

Step 14 : Create Class POService.java

```
package in.ac.famt.JDBCCodeForPA;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class POService {

    @Autowired
    private PORepository poRepository;
```

```

public List<POHeader> getPODetails() {
return poRepository.getPODetails();
}
// You can add more methods as needed for other business logic
}

```

Step 15 : Create Class PORepository.java

```

package in.ac.famt.JDBCCodeForPA;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public class PORepository {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    public List<POHeader> getPODetails() {
    String sql = "SELECT POHeader.poId, POHeader.supplNm, POHeader.podt,
    POHeader.poAmt, PODetails.srNo, PODetails.prodCode, PODetails.quantity,
    PODetails.rate " +
    "FROM POHeader INNER JOIN PODetails ON POHeader.poId = PODetails.poId";

    return jdbcTemplate.query(sql, new POHeaderRowMapper());
    }

    // You can add more methods as needed for other data access operations
    }

```

Step 16 : Create Class JdbCodeForPaApplication.java

```

package in.ac.famt.JDBCCodeForPA;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class JdbCodeForPaApplication {
    public static void main(String[] args) {
    SpringApplication.run(JdbCodeForPaApplication.class, args);
    }
}

```

Step 17 : Create Class application.properties

```

# Database Configuration
spring.datasource.url=jdbc:mysql://localhost:3306/podb
spring.datasource.username=root
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

```

Note:

Before Running the Project Create Database With Given Query:

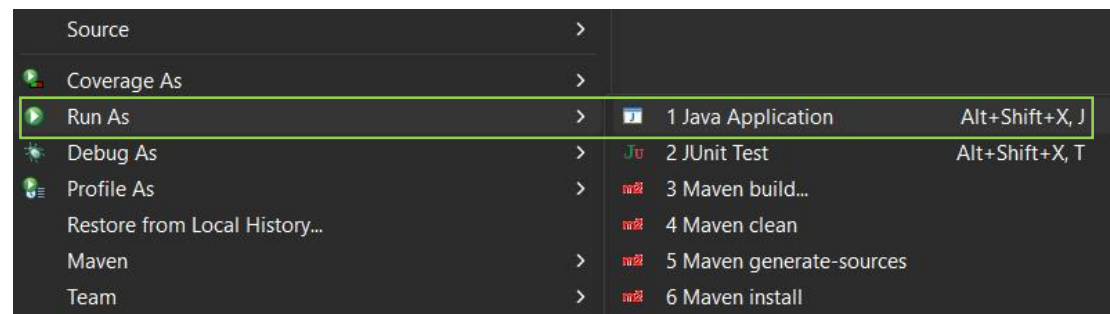
-- Create POHeader table

```
CREATE TABLE POHeader (  
    poId INT PRIMARY KEY,  
    supplNm VARCHAR(255),  
    podt DATE,  
    poAmt INT  
);
```

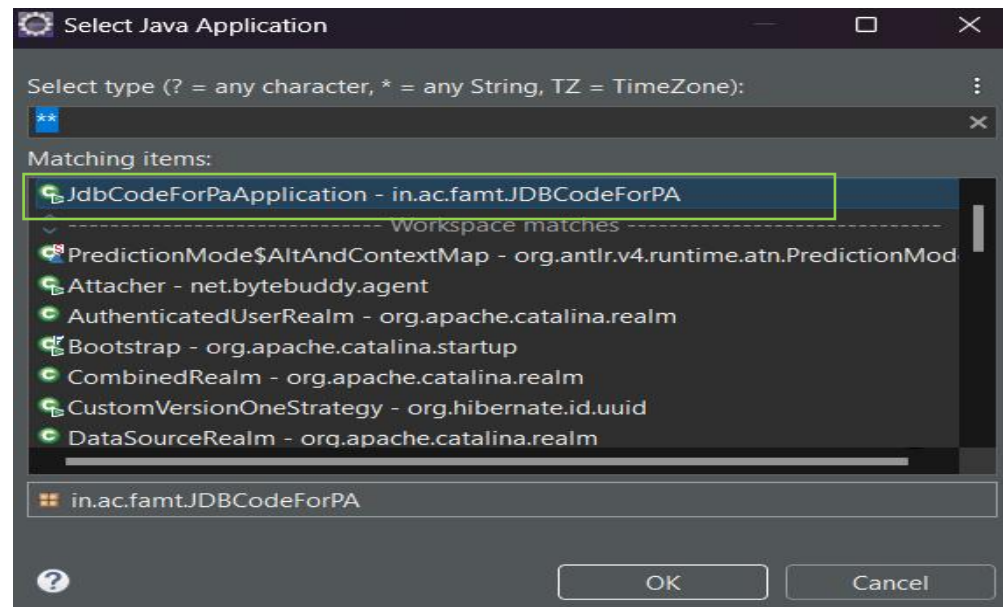
-- Create PODetails table

```
CREATE TABLE PODetails (  
    poId INT,  
    srNo INT,  
    prodCode VARCHAR(255),  
    quantity INT,  
    rate INT,  
    PRIMARY KEY (poId, srNo),  
    FOREIGN KEY (poId) REFERENCES POHeader(poId)  
);
```

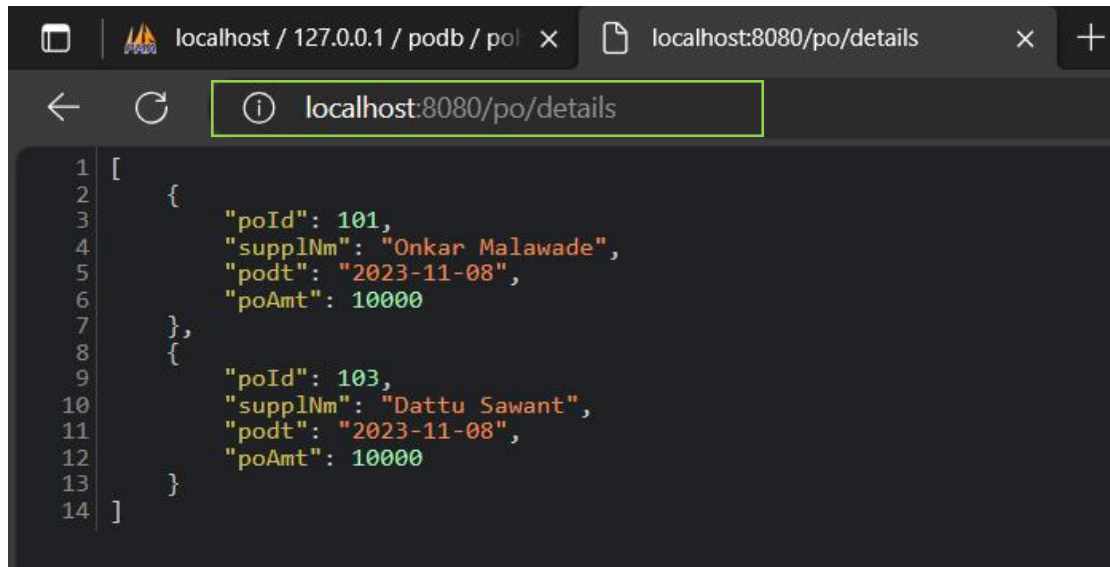
Step 18 : Run the Program with following steps:



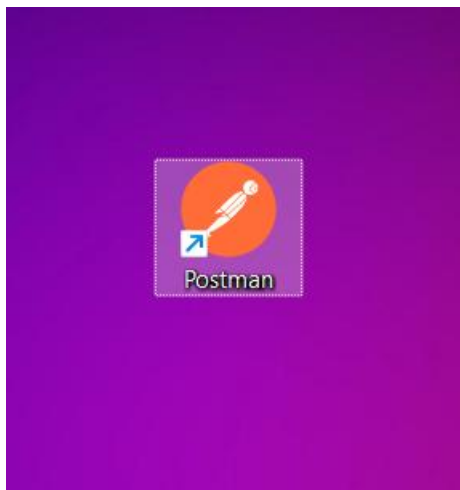
Step 19 : Select Our Application we want to Run and Click on it(OK).



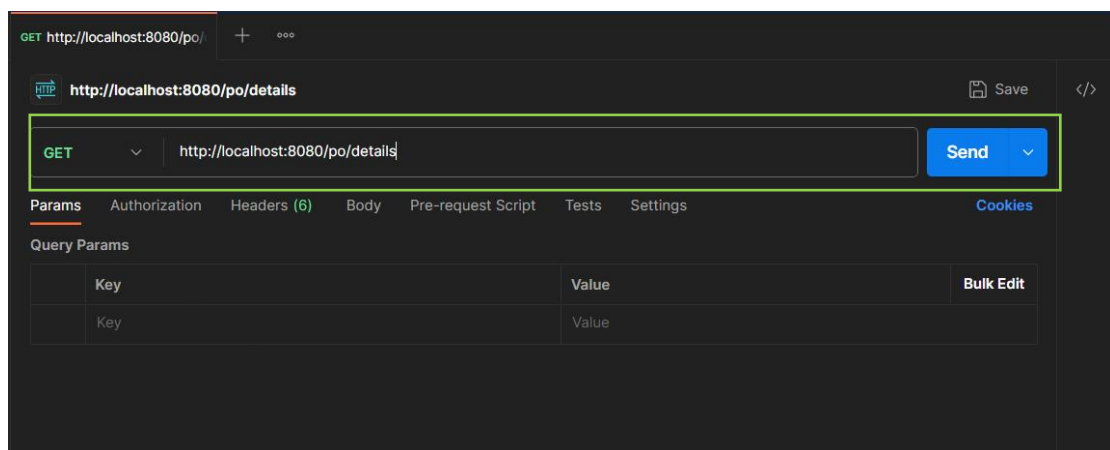
Step 20 : Open Browser: Type:



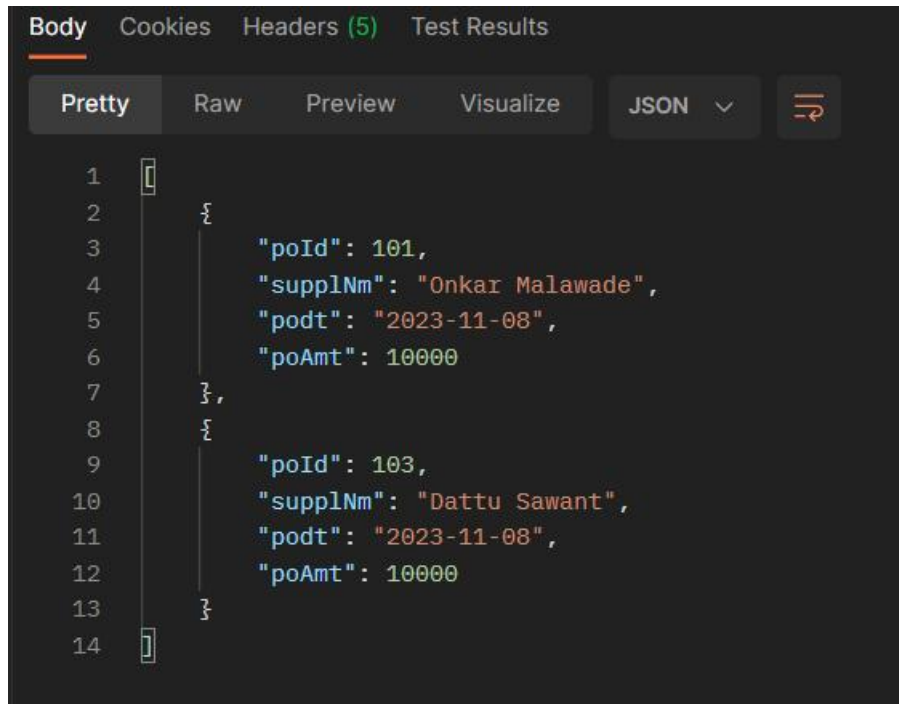
Step 21 : Another Method is Open Postman App



Step 22 : Select Get Method for fetching details from the Server as given below:



Step 23 : Press Send and Check the Details:



The screenshot shows a REST client interface with the 'Body' tab selected. The response is displayed in a 'Pretty' format, showing a JSON array of two objects. The first object has 'poId': 101, 'supplNm': 'Onkar Malawade', 'podt': '2023-11-08', and 'poAmt': 10000. The second object has 'poId': 103, 'supplNm': 'Dattu Sawant', 'podt': '2023-11-08', and 'poAmt': 10000. The interface includes tabs for 'Body', 'Cookies', 'Headers (5)', and 'Test Results'. Below the tabs are buttons for 'Pretty', 'Raw', 'Preview', 'Visualize', and a 'JSON' dropdown menu. A vertical line of numbers from 1 to 14 is on the left side of the code area.

```
1  [
2    {
3      "poId": 101,
4      "supplNm": "Onkar Malawade",
5      "podt": "2023-11-08",
6      "poAmt": 10000
7    },
8    {
9      "poId": 103,
10     "supplNm": "Dattu Sawant",
11     "podt": "2023-11-08",
12     "poAmt": 10000
13   }
14 ]
```