Practical 05

Customer and Order Tables
Customer Table: customer_id, customer_name, email
Order Table: order_id, order_date, customer_id, order_amt

1) Create a DAO (Data Access Object) classes that provides CRUD (Create, Read, Update, Delete) operations for the first table mentioned above.

Customer.java
```
package Question_01;

import java.util.ArrayList;
import java.util.List;

import question2.Order;

public class Customer {
    private int customerId;
    private String customerName;
    private String email;
    private List<Order> orders; // List of orders for the customer
    public Customer(int customerId, String customerName, String email) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.email = email;
        this.orders = new ArrayList<>(); // Initialize the orders list
    }
    public Customer() {
        this.orders = new ArrayList<>(); // Initialize the orders list
    }
    public int getCustomerId() {
        return customerId;
    }
    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }
    public String getCustomerName() {
        return customerName;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public List<Order> getOrders() {
        return orders;
    }
    public void setOrders(List<Order> orders) {
        this.orders = orders;
    }
    public void addOrder(Order order) {
        this.orders.add(order);
    }
    @Override
    public String toString() {
        return "Customer [customerId=" + customerId + ", customerName=" +
customerName + ", email=" + email + "]";
    }

}
```

CustomerDAO.java

```
package Question_01;

import org.springframework.jdbc.core.JdbcTemplate;

public class CustomerDAO {
    JdbcTemplate jdbcTemplate;

    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }
      public int addCustomer(int customerId, String customerName, String email) {
        String query = "INSERT INTO Customer (customer_id, customer_name, email)
VALUES (?, ?, ?)";
        return jdbcTemplate.update(query, customerId, customerName, email);
    }
    public void getAllRecords() {
        String query = "SELECT * FROM Customer";
        System.out.println(jdbcTemplate.queryForList(query));
    }

        public int updateCustomer(int customerId, String customerName, String email) {
        String query = "UPDATE Customer SET customer_name = ?, email = ? WHERE
customer_id = ?";
        return jdbcTemplate.update(query, customerName, email, customerId);
    }
       public int deleteCustomer(int customerId) {
        String query = "DELETE FROM Customer WHERE customer_id = ?";
        return jdbcTemplate.update(query, customerId);
    }
}
```

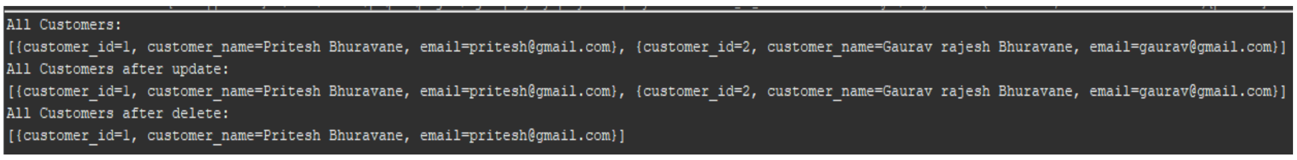
CustomerMain.java

```
package Question_01;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class CustomerMain {
    public static void main(String[] args) {

        ClassPathXmlApplicationContext appContext = new
ClassPathXmlApplicationContext("question1.xml");
        CustomerDAO customerDAO = (CustomerDAO)
appContext.getBean("customerDAO");
        customerDAO.addCustomer(1, "Pritesh Bhuravane", "pritesh@gmail.com");
        customerDAO.addCustomer(2, "Gaurav rajesh Bhuravane", "gaurav@gmail.com");
        System.out.println("All Customers:");
        customerDAO.getAllRecords();


        customerDAO.updateCustomer(3, "Pritesh Suresh Bhuravane",
"pritesh@gmail.com");
        System.out.println("All Customers after update:");
        customerDAO.getAllRecords();
        customerDAO.deleteCustomer(2);
        System.out.println("All Customers after delete:");
        customerDAO.getAllRecords();
        appContext.close();
    }
}
```

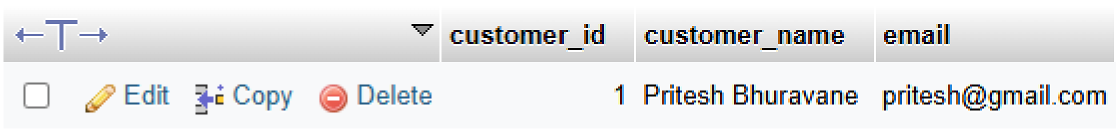
Question1.xml

```
<?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="ds" class="org.apache.commons.dbcp2.BasicDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/company"/>
        <property name="username" value="root"/>
    </bean>
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="ds"/>
    </bean>
    <bean id="customerDAO" class="Question_01.CustomerDAO">
        <property name="jdbcTemplate" ref="jdbcTemplate"/>
    </bean>
</beans>
```

Output:



```
All Customers:
[{customer_id=1, customer_name=Pritesh Bhuravane, email=pritesh@gmail.com}, {customer_id=2, customer_name=Gaurav rajesh Bhuravane, email=gaurav@gmail.com}]
All Customers after update:
[{customer_id=1, customer_name=Pritesh Bhuravane, email=pritesh@gmail.com}, {customer_id=2, customer_name=Gaurav rajesh Bhuravane, email=gaurav@gmail.com}]
All Customers after delete:
[{customer_id=1, customer_name=Pritesh Bhuravane, email=pritesh@gmail.com}]
```



←T→	customer_id	customer_name	email
☐ Edit Copy Delete	1	Pritesh Bhuravane	pritesh@gmail.com

2.Create a DAO (Data Access Object) classes that provides CRUD (Create, Read, Update, Delete) operations for the second table mentioned above.

Order.java

```
package question2;
```

```
import java.util.Date;

public class Order {
    private int orderId;
    private Date orderDate;
    private int customerId;
    private double orderAmt;
    public Order(int orderId, Date orderDate, int customerId, double orderAmt) {
        this.orderId = orderId;
        this.orderDate = orderDate;
        this.customerId = customerId;
        this.orderAmt = orderAmt;
    }
    public int getOrderId() {
        return orderId;
    }
    public void setOrderId(int orderId) {
        this.orderId = orderId;
    }
    public Date getOrderDate() {
        return orderDate;
    }
    public void setOrderDate(Date orderDate) {
        this.orderDate = orderDate;
    }
    public int getCustomerId() {
        return customerId;
    }
    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }
    public double getOrderAmt() {
        return orderAmt;
    }
    public void setOrderAmt(double orderAmt) {
        this.orderAmt = orderAmt;
    }
}
```

OrderDAO.java

```
package question2;
import org.springframework.jdbc.core.JdbcTemplate;
```

```
public class OrderDAO {
    JdbcTemplate jdbcTemplate;
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }
    public int addOrder(int orderId, String orderDate, int customerId, double orderAmt) {
        String query = "INSERT INTO `Order` (order_id, order_date, customer_id,
order_amt) VALUES (?, ?, ?, ?)";
        return jdbcTemplate.update(query, orderId, orderDate, customerId, orderAmt);
    }
    public void getAllOrders() {
        String query = "SELECT * FROM `Order`";
        System.out.println(jdbcTemplate.queryForList(query));
    }
    public int updateOrder(int orderId, String orderDate, int customerId, double orderAmt)
{
        String query = "UPDATE `Order` SET order_date = ?, customer_id = ?, order_amt =
? WHERE order_id = ?";
        return jdbcTemplate.update(query, orderDate, customerId, orderAmt, orderId);
    }
    public int deleteOrder(int orderId) {
        String query = "DELETE FROM `Order` WHERE order_id = ?";
        return jdbcTemplate.update(query, orderId);
    }}
```

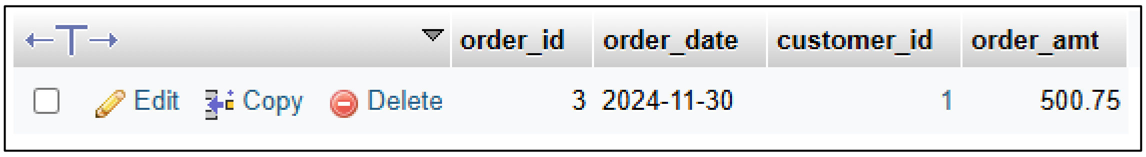
OrderMain.java

```
package question2;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class OrderMain {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("question2.xml");
        OrderDAO orderDAO = (OrderDAO) context.getBean("orderDAO");
        orderDAO.addOrder(105, "2024-11-30", 2, 500.75);
        orderDAO.getAllOrders();
        orderDAO.deleteOrder(101);
        context.close();
    }
}
```

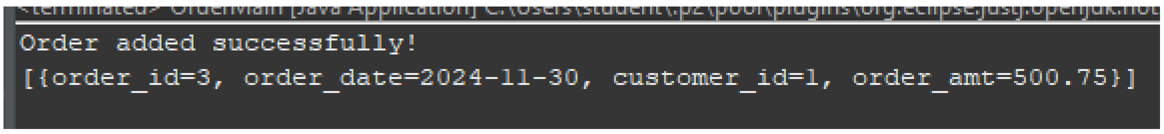
question2.xml

```
<?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="ds" class="org.apache.commons.dbcp2.BasicDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/company"/>
        <property name="username" value="root"/>
    </bean>
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="ds"/>
    </bean>
    <bean id="orderDAO" class="question2.OrderDAO">
        <property name="jdbcTemplate" ref="jdbcTemplate"/>
    </bean>

</beans>
```

Output:



	order_id	order_date	customer_id	order_amt
Edit Copy Delete	3	2024-11-30	1	500.75



```
<terminated> OrderMain [Java Application] C:\Users\student\.p2\pool\plugins\org.eclipse.justj.openjdk.hot
Order added successfully!
[{order_id=3, order_date=2024-11-30, customer_id=1, order_amt=500.75}]
```

3 Create a class to display all customers along with their respective order using
ResultSetExtractor Interface of Spring JDBC.

CustomerOrderDAO.java

```
package customeOrder;

public class CustomerOrderDTO {
    private int customerId;
    private String customerName;
    private String email;
    private int orderId;
    private String orderDate;
    private double orderAmount;
    public CustomerOrderDTO(int customerId, String customerName, String email,
                            int orderId, String orderDate, double orderAmount) {
        this.customerId = customerId;
        this.customerName = customerName;
        this.email = email;
        this.orderId = orderId;
        this.orderDate = orderDate;
        this.orderAmount = orderAmount;
    }
    public int getCustomerId() {
        return customerId;
    }
    public void setCustomerId(int customerId) {
        this.customerId = customerId;
    }
    public String getCustomerName() {
        return customerName;
    }
    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public int getOrderId() {
        return orderId;
    }
    public void setOrderId(int orderId) {
        this.orderId = orderId;
    }
    public String getOrderDate() {
        return orderDate;
    }
    public void setOrderDate(String orderDate) {
        this.orderDate = orderDate;
    }
    public double getOrderAmount() {
        return orderAmount;
    }
    public void setOrderAmount(double orderAmount) {
        this.orderAmount = orderAmount;
    }
    @Override
    public String toString() {
        return "Customer ID: " + customerId +
                ", Name: " + customerName +
                ", Email: " + email +
                ", Order ID: " + orderId +
                ", Order Date: " + orderDate +
                ", Order Amount: " + orderAmount;
    }
}
```

CustomerOderDAO.java

```
package customeOrder;
import org.springframework.jdbc.core.JdbcTemplate;
import Question_01.*;

import java.util.Map;

public class CustomerOrderDAO {
    private JdbcTemplate jdbcTemplate;
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }
    public Map<Integer, Customer> getAllCustomersWithOrders() {
        String sql = "SELECT c.customer_id, c.customer_name, c.email, o.order_id,
o.order_date, o.order_amt " +
                     "FROM Customer c " +
                     "LEFT JOIN `Order` o ON c.customer_id = o.customer_id";
        return jdbcTemplate.query(sql, new CustomerOrderExtractor());
    }
}
```

CustomerOrderExtractor.java

```
package customeOrder;
```

```
import org.springframework.jdbc.core.ResultSetExtractor;
import Question_01.*;
import question2.Order;

import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;

public class CustomerOrderExtractor implements ResultSetExtractor<Map<Integer,
Customer>> {
    @Override
    public Map<Integer, Customer> extractData(ResultSet rs) throws SQLException {
        Map<Integer, Customer> customers = new HashMap<>();

        while (rs.next()) {
            int customerId = rs.getInt("customer_id");
            Customer customer = customers.get(customerId);
            if (customer == null) {
                customer = new Customer(
                        customerId,
                        rs.getString("customer_name"),
                        rs.getString("email")
                );
                customers.put(customerId, customer);
            }
            Order order = new Order(
                    rs.getInt("order_id"),
                    rs.getDate("order_date"),
                    customerId,
                    rs.getDouble("order_amt")
            );
            customer.addOrder(order);
        }
        return customers;
    }
}
```

CustomerOrderMain.java

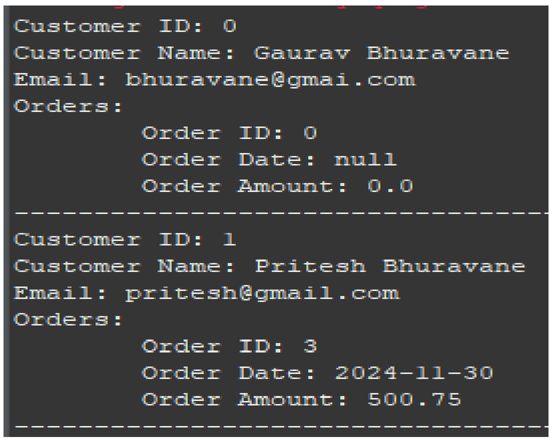
```
package customeOrder;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import Question_01.*;
import java.util.Map;
public class CustomerOrderMain {
    public static void main(String[] args) {
        ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("question3.xml");
        CustomerOrderDAO customerOrderDAO = (CustomerOrderDAO)
context.getBean("customerOrderDAO");
        Map<Integer, Customer> customers =
customerOrderDAO.getAllCustomersWithOrders();
        for (Customer customer : customers.values()) {
            System.out.println("Customer ID: " + customer.getCustomerId());
            System.out.println("Customer Name: " + customer.getCustomerName());
            System.out.println("Email: " + customer.getEmail());
            System.out.println("Orders:");
            customer.getOrders().forEach(order -> {
                System.out.println("\tOrder ID: " + order.getOrderId());
                System.out.println("\tOrder Date: " + order.getOrderDate());
                System.out.println("\tOrder Amount: " + order.getOrderAmt());
            });
            System.out.println("--------------------------------------------------");
        }
        context.close();
    }
}
```

question3.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="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver"/>
        <property name="url" value="jdbc:mysql://localhost:3306/company"/>
        <property name="username" value="root"/>
    </bean>
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource"/>
    </bean>
    <bean id="customerOrderDAO" class="customeOrder.CustomerOrderDAO">
```

```
        <property name="jdbcTemplate" ref="jdbcTemplate"/>
    </bean>
</beans>
```

Output:



```
Customer ID: 0
Customer Name: Gaurav Bhuravane
Email: bhuravane@gmai.com
Orders:
        Order ID: 0
        Order Date: null
        Order Amount: 0.0
---------------------------------
Customer ID: 1
Customer Name: Pritesh Bhuravane
Email: pritesh@gmail.com
Orders:
        Order ID: 3
        Order Date: 2024-11-30
        Order Amount: 500.75
---------------------------------
```

4 Create a method called &quot;getCustomerByName&quot; in the DAO class that takes a name as input and returns the details of the customer with that name. The method should use PreparedStatement in Spring JdbcTemplate to execute a SQL query and retrieve the employee information using RowMapper.

CustomerDAo.java

```
package question4;

import Question_01.Customer; // Importing the existing Customer class
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import java.sql.ResultSet;
import java.sql.SQLException;
public class CustomerDAO {
    private JdbcTemplate jdbcTemplate;
    public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
    }
    @SuppressWarnings("deprecation")
         public Customer getCustomerByName(String customerName) {
        String query = "SELECT * FROM Customer WHERE customer_name = ?";
        try {
            return jdbcTemplate.queryForObject(query, new Object[]{customerName}, new
RowMapper<Customer>() {
```

```
                @Override
                public Customer mapRow(ResultSet rs, int rowNum) throws SQLException {
                    return new Customer(
                        rs.getInt("customer_id"),
                        rs.getString("customer_name"),
                        rs.getString("email")
                    );
                }
            });
        } catch (org.springframework.dao.EmptyResultDataAccessException e) {
            System.out.println("No customer found with the name: " + customerName);
            return null;
        }
    }
}
```

CustomerMian.java

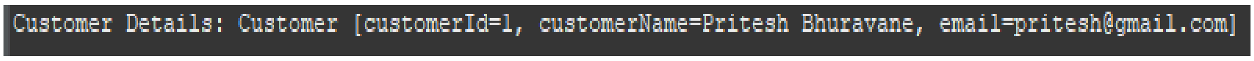
```
package question4;

import Question_01.*;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class CustomerMain {
    public static void main(String[] args) {
        ApplicationContext context =
new ClassPathXmlApplicationContext("question4.xml");
        CustomerDAO customerDAO = (CustomerDAO) context.getBean("customerDAO");
        String customerName = "Pritesh Bhuravane";
        Customer customer = customerDAO.getCustomerByName(customerName);
        if (customer != null) {
            System.out.println("Customer Details: " + customer);
        } else {
            System.out.println("Customer not found.");
        }
    }
}
```

question4.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="dataSource"
class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/company" />
        <property name="username" value="root" />
    </bean>
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource" />
    </bean>
    <bean id="customerDAO" class="question4.CustomerDAO">
        <property name="jdbcTemplate" ref="jdbcTemplate" />
    </bean>
</beans>
```

Output:



```
Customer Details: Customer [customerId=1, customerName=Pritesh Bhuravane, email=pritesh@gmail.com]
```

5 Create a class for executing stored procedure from database that selects a particular record as per the user input.

StoredProcedureExecutor.java

```
package question5;

import java.util.Map;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.simple.SimpleJdbcCall;
public class StoredProcedureExecutor {
    private JdbcTemplate jdbcTemplate;
    private SimpleJdbcCall customerJdbcCall;
    private SimpleJdbcCall orderJdbcCall;
    public StoredProcedureExecutor(JdbcTemplate jdbcTemplate) {
        this.jdbcTemplate = jdbcTemplate;
        this.customerJdbcCall =
new SimpleJdbcCall(jdbcTemplate).withProcedureName("GetCustomerById");
        this.orderJdbcCall =
new SimpleJdbcCall(jdbcTemplate).withProcedureName("GetOrderById");
    }
    public Map<String, Object> getCustomerById(int customerId) {
        return customerJdbcCall.execute(Map.of("customerId", customerId));
    }
    public Map<String, Object> getOrderById(int orderId) {
        return orderJdbcCall.execute(Map.of("orderId", orderId));
    }
}
```

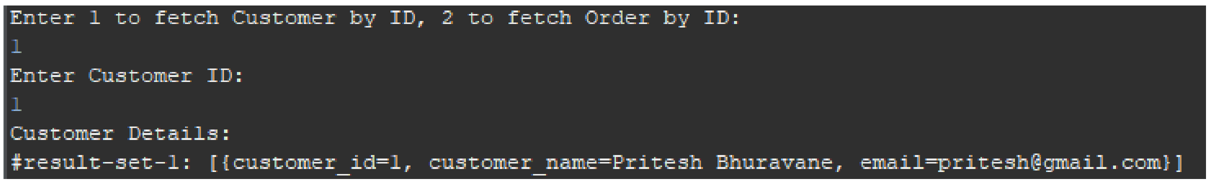
StoredProcedureMain.java

```
package question5;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import java.util.Map;
import java.util.Scanner;
public class StoredProcedureMain {
    public static void main(String[] args) {
        ApplicationContext context =
new ClassPathXmlApplicationContext("question5.xml");
        StoredProcedureExecutor executor = context.getBean("storedProcedureExecutor",
StoredProcedureExecutor.class);
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter 1 to fetch Customer by ID, 2 to fetch Order by ID:");
        int choice = scanner.nextInt();
        if (choice == 1) {
            System.out.println("Enter Customer ID:");
            int customerId = scanner.nextInt();
            Map<String, Object> customer = executor.getCustomerById(customerId);
            if (customer != null && !customer.isEmpty()) {
                System.out.println("Customer Details:");
                customer.forEach((key, value) -> System.out.println(key + ": " + value));
            } else {
                System.out.println("No customer found with ID: " + customerId);
            }
        } else if (choice == 2) {
            System.out.println("Enter Order ID:");
            int orderId = scanner.nextInt();
            Map<String, Object> order = executor.getOrderById(orderId);

            if (order != null && !order.isEmpty()) {
                System.out.println("Order Details:");
                order.forEach((key, value) -> System.out.println(key + ": " + value));
            } else {
                System.out.println("No order found with ID: " + orderId);
            }
        } else {
            System.out.println("Invalid choice.");
        }
        scanner.close();
    }
}
```

question5.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="dataSource" class="org.apache.commons.dbcp2.BasicDataSource">
        <property name="driverClassName" value="com.mysql.cj.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/company" />
        <property name="username" value="root" />
    </bean>
    <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
        <property name="dataSource" ref="dataSource" />
    </bean>
    <bean id="storedProcedureExecutor" class="question5.StoredProcedureExecutor">
        <constructor-arg ref="jdbcTemplate" />
    </bean>
</beans>
```

Output:



```
Enter 1 to fetch Customer by ID, 2 to fetch Order by ID:
1
Enter Customer ID:
1
Customer Details:
#result-set-1: [{customer_id=1, customer_name=Pritesh Bhuravane, email=pritesh@gmail.com}]
```