Problem Statement

To generate a Summary Report for Northwind food distribution company for a given time period.

The Summary Report includes the following:

1. Customer Information
    a. Customer Name, Address, Number of Orders in the period , Total Value
    b. Do not include customers with no orders
2. Product Information
    a. Include Product Name, Supplier, Units Sold and Value of products sold
    b. Do not include products with no orders
3. Supplier Information
    a. Include Supplier Name, Address, Number of Products sold and total value of business
    b. Do not include suppliers with no sales

Solution

Step 1 - Create Basic Project and Handle Input Validation

1. Input parameters include
    a. Start Date
    b. End Date
    c. Output File
2. Validation includes
    a. Checking if all the above mentioned 3 parameters are present
    b. Checking if the date is mentioned in the appropriate format
3. Configuration framework set up
    a. Initial approach included using configuration files or pre-compiled classes for obtaining configuration values for accessing the database. Hence the approach included the following steps
        i. Add an **application.conf** resource file at the project level whose properties can be read by the Java Classes. This is secure since this configuration file can be set on the deployment machine itself
        ii. The flaw observed in this approach was the fact that compiled classes could be ***decompiled***.
    b. A better and modern approach to solve the above issue included the use of environment variables
        i. The **environment variables** are set on any machine on which this application executes.
        ii. This ensures that **sensitive** and **confidential** data is not present in Source Control, Pre-Compiled Java Code or Compiled Java Classes.

Step 1 - Create SQL queries for fetching the above information

1. Customer Information Summary

```
Select customers.ContactName, " +
        "customers.Address, count(*) as \"No Of Orders\", " +
        "sum(orderdetails.UnitPrice * orderdetails.Quantity - orderdetails.Discount) as
\"Sum\"\n" +
        "from orders, orderdetails, customers\n" +
        "where orders.OrderID = orderdetails.OrderID\n" +
        "and orders.CustomerID = customers.CustomerID\n" +
        "and orders.OrderDate between " + startDate.toString() + " and " +
endDate.toString() + "\n" +
        "group by customers.CustomerID\n" +
        "having count(*) > 0;
```

2. Product Information Summary

```
Select categories.CategoryName, products.ProductName, suppliers.ContactName,
sum(orderdetails.Quantity) as \"units_sold\", sum(orderdetails.Quantity *
orderdetails.UnitPrice - orderdetails.Discount) as \"sale value\"\n" +
        "from categories, products, suppliers, orderdetails, orders\n" +
        "where categories.CategoryID = products.CategoryID\n" +
        "and suppliers.SupplierID = products.SupplierID\n" +
        "and orderdetails.OrderID = orders.OrderID\n" +
        "and orderdetails.ProductID = products.ProductID\n" +
        "and orders.OrderDate between " + startDate.toString() + " and " +
endDate.toString() + "\n" +
        "group by categories.CategoryName, products.ProductName,
suppliers.ContactName, orderdetails.ProductID\n" +
        "having count(products.ProductID) > 0;
```

3. Supplier Information Summary

```
Select suppliers.ContactName, suppliers.Address, count(products.ProductID) as
        \"num_of_products\", sum(orderdetails.Quantity * orderdetails.UnitPrice -
        orderdetails.Discount) as \"product value\"\n" +
"     from products, suppliers, orderdetails, orders\n" +
"     where suppliers.SupplierID = products.SupplierID\n" +
"     and orderdetails.OrderID = orders.OrderID\n" +
"     and orderdetails.ProductID = products.ProductID\n" +
                "and orders.OrderDate between " + startDate.toString() + " and " +
endDate.toString() + "\n" +
                "     group by products.ProductName, suppliers.ContactName,
orderdetails.ProductID\n" +
"     having count(suppliers.SupplierID) > 0
```

4. JDBC Connection

JDBC connection is established using the mysql connector made available during the course of this assignment. A connection provider pattern is established and used as a singleton object for any new db access that is required

5. XML Formatting and Conversion/Marshalling

Marshalling result set to XML is done using documents and appending child nodes appropriately.

Reasons why my code can be deployed readily:

1. Configurability of the project
    a. Since the end points and credentials required for connecting to a database are not hard coded, it would be very easy to connect to any database on demand
    b. It can be packaged as a Jar and deployed separately on any server required on demand