CSCI 3901 Assignment 5

Name : Yogish Honnadevipura Gopalakrishna(B00928029)

Overview

This program creates an XML file from the extracted summary information from the database.

This program extracts a summary of information about customers, Product and stores over a specific time frame taken as the input. The csci3901 database contains the information. The collected data is transformed into an XML file.

Files and external data

There are two main files:

- Main.java -> This program takes information about start and end dates and the name of the xml file to be created.
- XmlCreator -> This program interacts with main.java to get the information about dates and then creates an XML document by getting the data from the database.

Choices

I have grouped by product_name instead of product_id as there were multiple IDs for the same product names.

Key algorithms and design elements

This program has two files:

Main.java is used to take information about start date, end date and file name. The taken information is used as a parameter to call CreateDocument method in the XmlCreator file.

XmlCreator takes the information from the start date and then interacts with the database to get the customer, product and store information. The results obtained from the database are then used to build the XML document. The Xml document would not get built if any error in input regarding the format of the date.

Customer Information details

Query:

```
“SELECT c.customer_id, CONCAT(c.first_name,' ', c.last_name) as customer_name,
c.street as street_address, c.city, c.state, c.zip_code,
sum((oi.list_price*oi.quantity) - ((oi.quantity*oi.list_price) * oi.discount)) as
order_value, sum(oi.quantity) as bicycles_purchased
FROM customers c
JOIN orders o ON c.customer_id = o.customer_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN products p ON oi.product_id = p.product_id
        where o.order_date BETWEEN '2016-01-01' AND '2017-01-01' and
        c.customer_id NOT IN (
          SELECT customer_id
          FROM orders
          GROUP BY customer_id, order_date
          HAVING order_date > (SELECT MIN(order_date) FROM orders LIMIT 1)
AND order_date < '2016-01-01' )
GROUP BY c.customer_id;”
```

Here ‘2016-01-01’ represents the start date and ‘2017-01-01’ represents the end date.This query extracts all customer name, address, value of purchases and number of bicycles purchased within the given interval of time.

Product Information Details

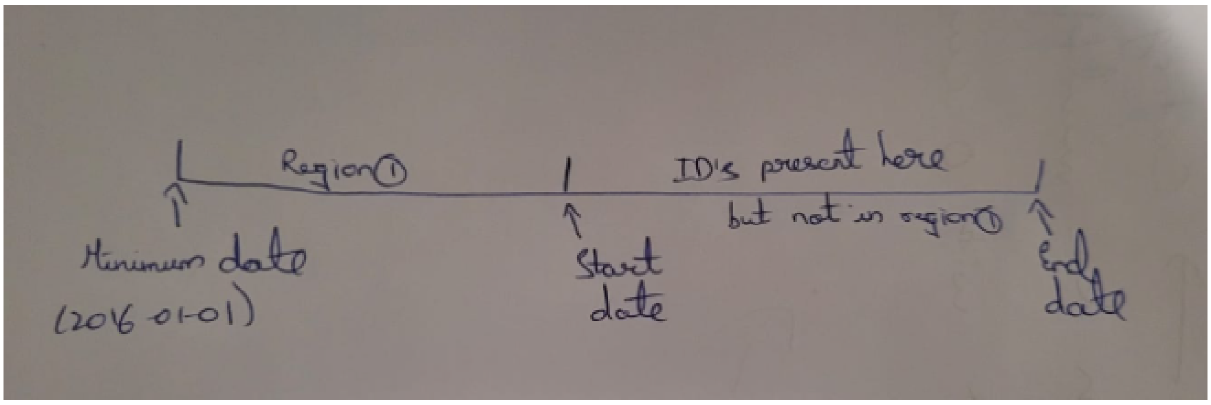
Query:

```
“SELECT p.product_name, b.brand_name as brand,
group_concat(distinct(c.category_name)) as category, s.store_name,
sum(oi.quantity) as units_sold
FROM products p
JOIN categories c ON p.category_id = c.category_id
JOIN brands b ON p.brand_id = b.brand_id
JOIN order_items oi ON p.product_id = oi.product_id
JOIN orders o ON oi.order_id = o.order_id
JOIN stores s ON o.store_id = s.store_id
```

```
WHERE o.order_date BETWEEN '2016-01-01' AND '2016-01-05'
and p.product_id NOT IN (
        SELECT p.product_id
   FROM products p
   JOIN order_items oi ON p.product_id = oi.product_id
   JOIN orders o ON oi.order_id = o.order_id
   GROUP BY p.product_id, o.order_date
   HAVING o.order_date > (SELECT MIN(order_date) FROM orders LIMIT 1) AND
o.order_date < '2016-01-01')
GROUP BY p.product_name, s.store_id;”
```

Here ‘2016-01-01’ represents the start date and ‘2017-01-01’ represents the end date.

Design - The **first** and **second queries** use the below logic to get the IDs of customers who had their first order and products which were first sold in the given time period.



The name of the product, its brand is printed only once in this query, and all the categories and store sales that come under the product will belong to the same new product tag.

Store Information Details

To get the Store information I used 2 queries. The first query is used to get the number of customers served in each store.

Query**:**

```
“SELECT COUNT(DISTINCT(c.customer_id)) as customer_served
from stores s
JOIN orders o ON s.store_id = o.store_id
```

JOIN staffs st ON s.store_id = st.store_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN customers c ON o.customer_id = c.customer_id
WHERE o.order_date BETWEEN '2016-01-01' AND '2017-01-01'
GROUP BY s.store_id;"

The result is stored in the array list namely '**customerList'** in the retrieved order. Then other information of the stores are extracted with minor modifications to the above query.

"SELECT s.store_name, s.city as store_city, count(DISTINCT(st.staff_id)) as employee_count, CONCAT(c.first_name,' ', c.last_name) as customer_name, sum((oi.list_price*oi.quantity) - ((oi.quantity*oi.list_price) * oi.discount)) as customer_sales_value
from stores s
JOIN orders o ON s.store_id = o.store_id
JOIN staffs st ON s.store_id = st.store_id
JOIN order_items oi ON o.order_id = oi.order_id
JOIN customers c ON o.customer_id = c.customer_id
WHERE o.order_date BETWEEN '2016-01-01' AND '2017-01-01'
GROUP BY s.store_id, c.customer_id;"

Design - As the same stores would have served multiple customers, even though there are multiple rows of the same store names, the store names are used only once and the number of customers is retrieved by indexing **customerList.** Then by using a for loop the customer name and their sales value are extracted for each store.

The Document is built using **Document builder** and **Document builder factory** through which the Xml tags are being created. Finally by using the **transformer.setOutputProperty,** XML file is formatted using the pretty print which makes it easily readable.

References

https://www.w3schools.com/xml/
https://mkyong.com/java/how-to-create-xml-file-in-java-dom/
https://www.tutorialspoint.com/java_xml/java_dom_create_document.htm

Argument

This program is ready to be deployed due to the the following reasons:

- The program is flexible with dates. Even when the date changes, the program is flexible and gives an output according to that data modified
- The program is rigorously tested by using multiple dates. Even when the dates are reversed, the program just creates empty tags which says that no data can be generated between those dates
- The code is easy to read and understand. It is simple to follow and the future modifications can be done easily to it.
- The XML is printed in a way that it is very easily understandable and readable.
- As the program won't import any external packages or libraries, the program can be run on any system which has Java and JDBC driver installed in it.
- As the program is easily readable, maintainable, flexible and is compatible with other devices I believe that the program is ready to be deployed to the production server.