XML Data Processing

ITC5202 – Project 1

Submitted By : Sandeep Das and Rohan Vasudev Patel

2/24/2022

Under the Guidance of: Prof Shahdad Shariatmadari

This document explains how to process Customer/Order XML data

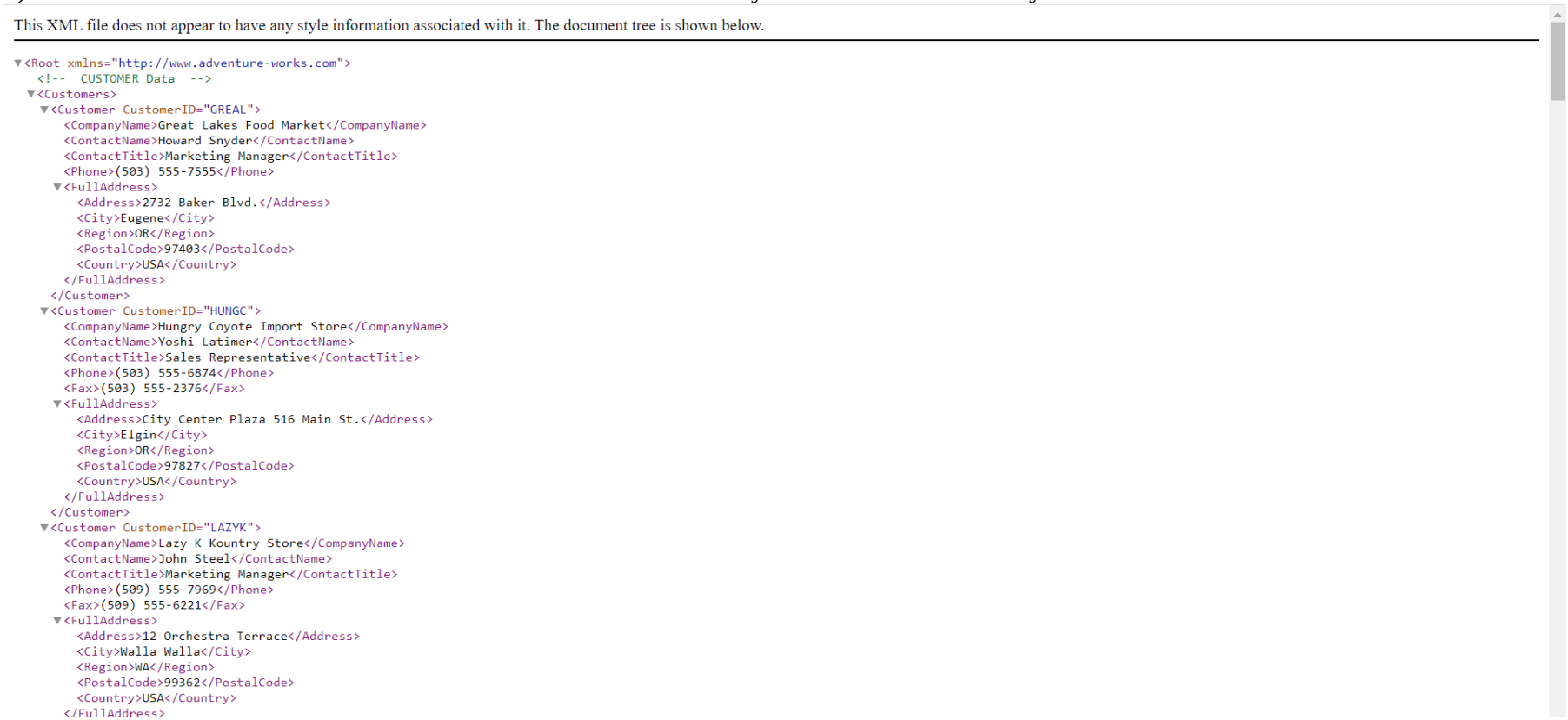
Table of Contents

Step 1 :

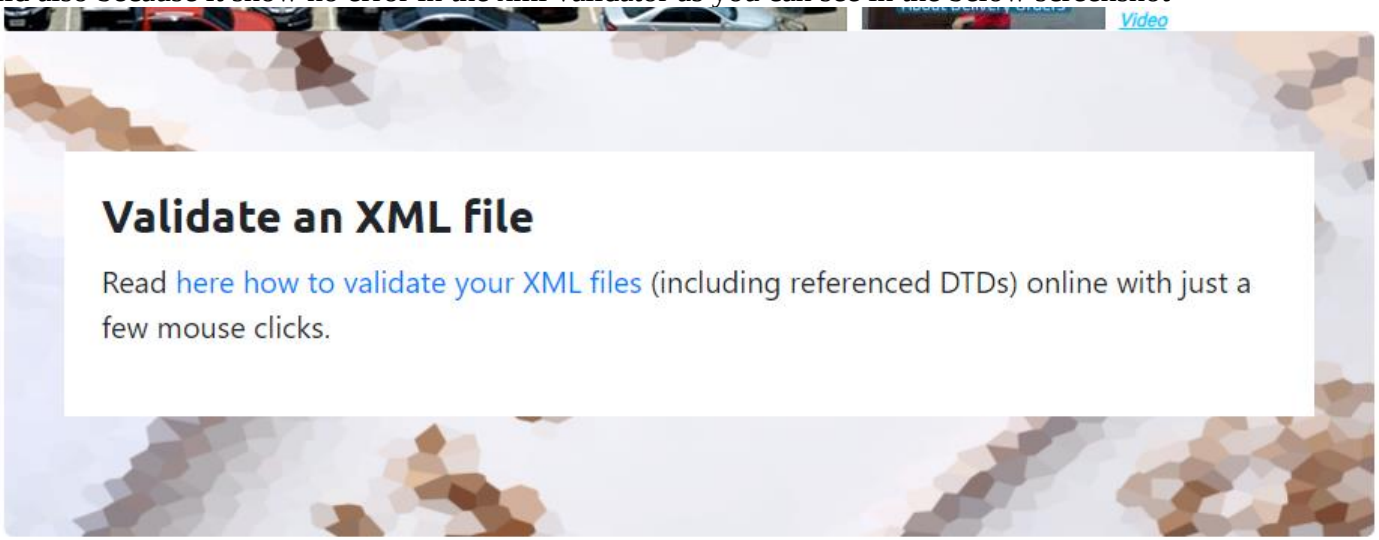
(Describe you answer. How did you prove that the document is well-formed and valid? Add screenshots)

Answer->

1) -this document is well formed because it runs without an any error in the browser as you can see in the below screenshot



2) - this document is valid also because it show no error in the xml validator as you can see in the below screenshot



No errors were found

The following files have been uploaded so far:
XML document:
Click on any file name if you want to edit the file.

3) - is there any namespace?
Answer-> yes there is a one default namespace which is written below

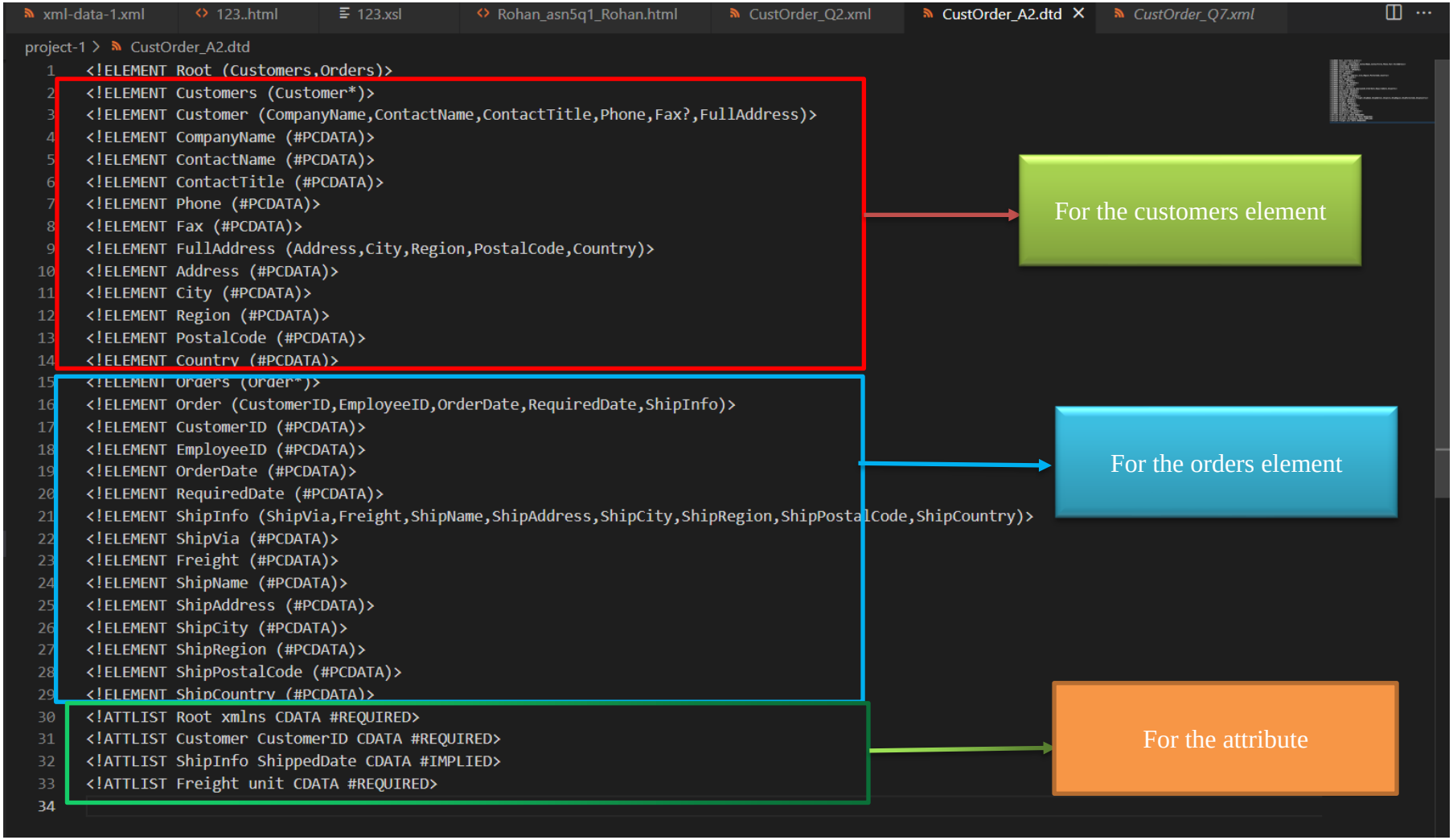
```
xmlns="http://www.adventure-works.com">
```

Step 2 and 3 : XML Structure

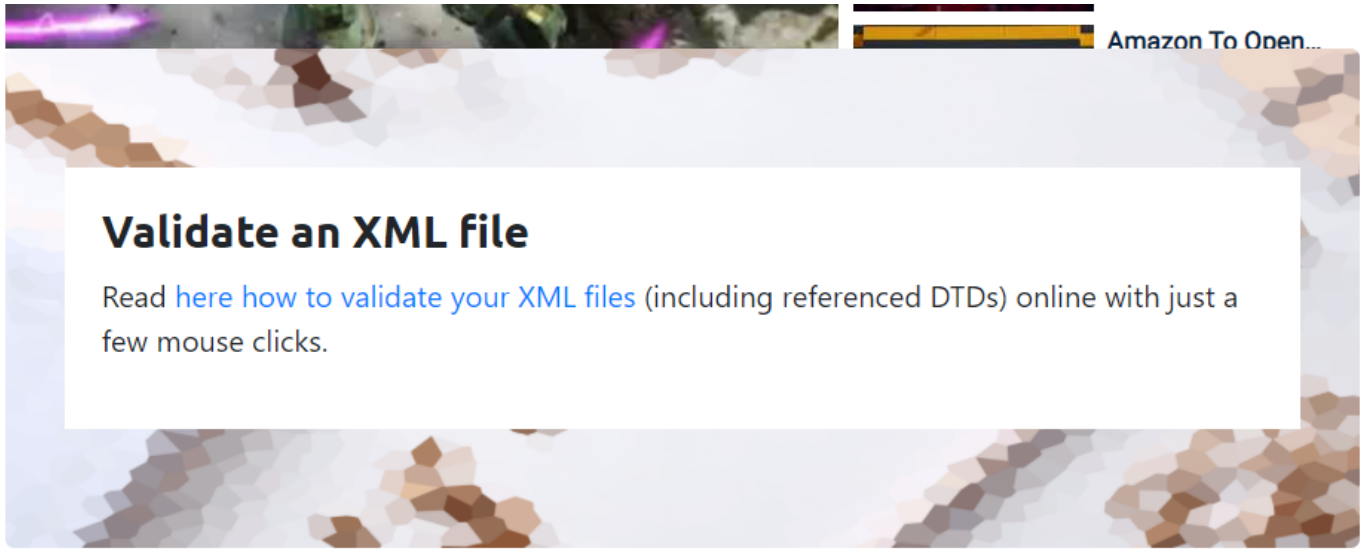
(1) Explain the major steps that you take to create DTD. Did you create a .dtd file, or you keep the DTD declaration inside the XML file? Why?
(2) Explain the major steps that you take to create XML Schema.
(3) How did you validate them? Add screenshots.
(4) Compare the DTD and Schema and show how DTD declaration are matched with Schema.

Answer-1)
I have created the external dtd which is shown below and the reason I choose external dtd between external and internal dtd is you can link the external dtd to any file apart from this the actual file is already big and if you add internal then it will become more long and when you are doing the other question then there is chance that you edit the dtd unintentionally so it is better to use external and it also look clear.so that's the reason I have used external dtd
Code
Link to the external dtd

EXTERNAL DTD



Output



No errors were found

The following files have been uploaded so far:

XML document:

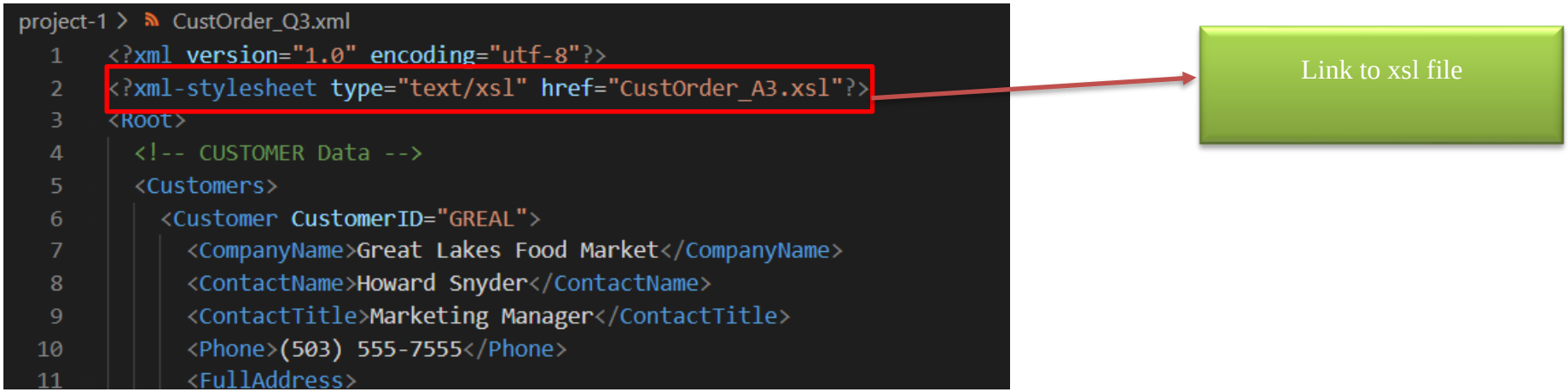
CustOrder_A2.dtd

Click on any file name if you want to edit the file.

2) Explain the major steps that you take to create XML Schema.

Answer->

Link to xsl file

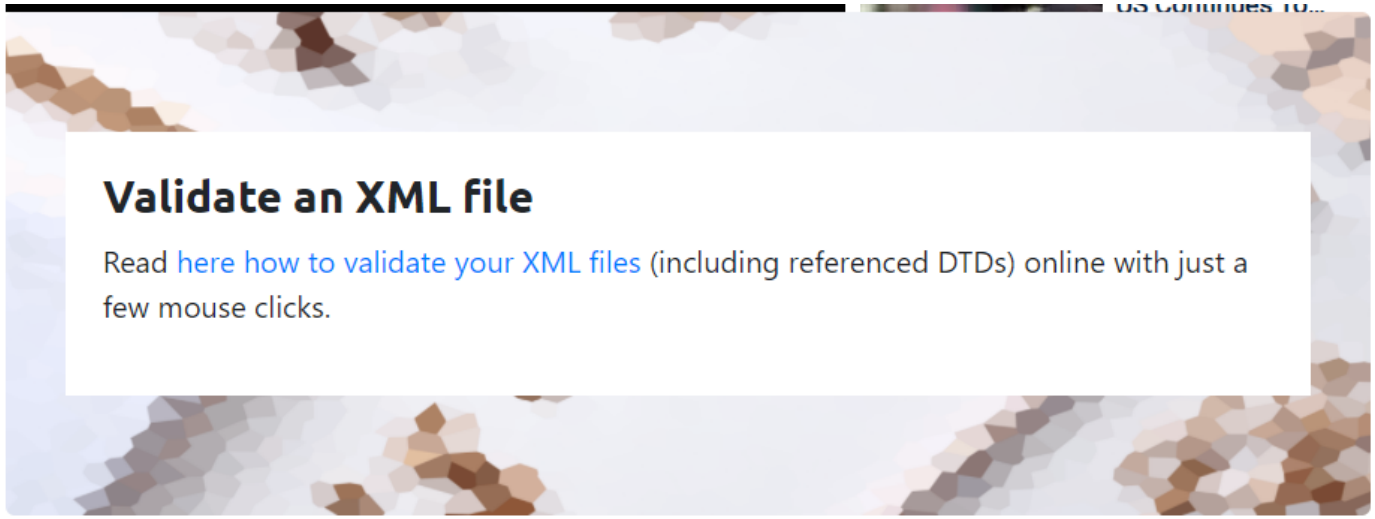


Xsl file

project-1 > CustOrder_A3.xsl

```xml
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Root">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Customers">
          <xs:annotation>
            <xs:documentation>CUSTOMER Data</xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Customer" maxOccurs="unbounded" minOccurs="0">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element type="xs:string" name="CompanyName"/>
                    <xs:element type="xs:string" name="ContactName"/>
                    <xs:element type="xs:string" name="ContactTitle"/>
                    <xs:element type="xs:string" name="Phone"/>
                    <xs:element type="xs:string" name="Fax" minOccurs="0"/>
                    <xs:element name="FullAddress">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element type="xs:string" name="Address"/>
                          <xs:element type="xs:string" name="City"/>
                          <xs:element type="xs:string" name="Region"/>
                          <xs:element type="xs:int" name="PostalCode"/>
                          <xs:element type="xs:string" name="Country"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                  <xs:attribute type="xs:string" name="CustomerID" use="optional"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="Orders">
```

3) How did you validate them? Add screenshots.



No errors were found

The following files have been uploaded so far:
XML document:
XML schema:
Click on any file name if you want to edit the file.

4)

5) Compare the DTD and Schema and show how DTD declaration are matched with Schema.
Answer-> first lets compare the element how both dtd and xsl declare element
In dtd elements are declared like this
<!ELEMENT ELEMENT NAME (CHILD1,CHILD2)>
For example here we have declared the customer element like this in dtd

```
<!ELEMENT Customer (CompanyName,ContactName,ContactTitle,Phone,Fax?,FullAddress)>
<!ELEMENT CompanyName (#PCDATA)>
<!ELEMENT ContactName (#PCDATA)>
<!ELEMENT ContactTitle (#PCDATA)>
<!ELEMENT Phone (#PCDATA)>
<!ELEMENT Fax (#PCDATA)>
<!ELEMENT FullAddress (Address,City,Region,PostalCode,Country)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Region (#PCDATA)>
<!ELEMENT PostalCode (#PCDATA)>
<!ELEMENT Country (#PCDATA)>
```

Where as in xsl we have declared the same element like this

```
<xs:element name="Customers">
  <xs:annotation>
    <xs:documentation>CUSTOMER Data</xs:documentation>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Customer" maxOccurs="unbounded" minOccurs="0">
        <xs:complexType>
          <xs:sequence>
            <xs:element type="xs:string" name="CompanyName"/>
            <xs:element type="xs:string" name="ContactName"/>
            <xs:element type="xs:string" name="ContactTitle"/>
            <xs:element type="xs:string" name="Phone"/>
            <xs:element type="xs:string" name="Fax" minOccurs="0"/>
            <xs:element name="FullAddress">
              <xs:complexType>
                <xs:sequence>
                  <xs:element type="xs:string" name="Address"/>
                  <xs:element type="xs:string" name="City"/>
                  <xs:element type="xs:string" name="Region"/>
                  <xs:element type="xs:int" name="PostalCode"/>
                  <xs:element type="xs:string" name="Country"/>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
```

In the xsl we specify the element one by one by using the xs:sequence here fax is used only one time so in xsl they use minOccurs where as in dtd we add one questionmark after that as it is seen in the above dtd screenshot

Step 4 : Design XSLT

Write an XSLT document to transform order.xml into an HTML document that displays all "**Customer's Data**" in a HTML table format.

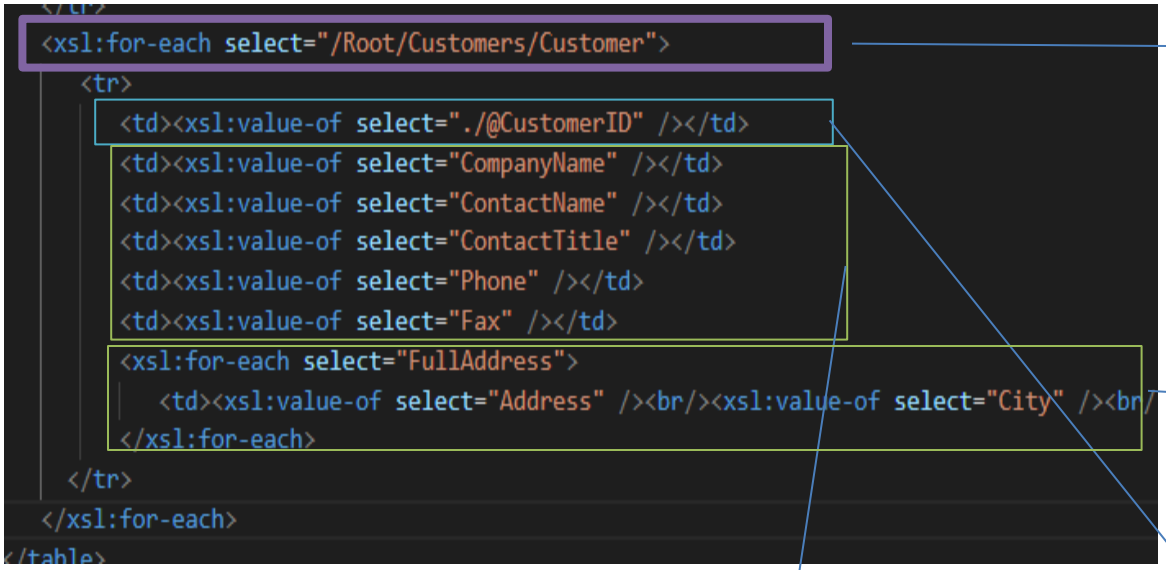
CustOrder_Q4.xsl

CustOrder_Q4.xsl

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>Companies</title>
      </head>
      <body>
      <h1>Customer</h1>
          <table border="1" >
            <tr>
              <th>Customer Id</th>
              <th>Company Name</th>
              <th>Contact Name</th>
              <th>Contact Title</th>
              <th>Phone</th>
              <th>Fax</th>
              <th>Address</th>
            </tr>
            <xsl:for-each select="/Root/Customers/Customer">
              <tr>
                <td><xsl:value-of select="./@CustomerID" /></td>
                <td><xsl:value-of select="CompanyName" /></td>
                <td><xsl:value-of select="ContactName" /></td>
                <td><xsl:value-of select="ContactTitle" /></td>
                <td><xsl:value-of select="Phone" /></td>
                <td><xsl:value-of select="Fax" /></td>
                <xsl:for-each select="FullAddress">
                  <td><xsl:value-of select="Address" /><br/><xsl:value-of select="City" /><br/><xsl:value-of select="Region" /><br/><xsl:value-of select="PostalCode'
                </xsl:for-each>
              </tr>
            </xsl:for-each>
          </table>
      </body>
    </html>
  </xsl:template>

</xsl:stylesheet>

```

CustOrder_Q4.xml

CustOrder_Q4.xsl



We will loop through all customers here by using this XPath inside for-each statement

Display internal tag values of Address attributes.

Display other tag values inside loop

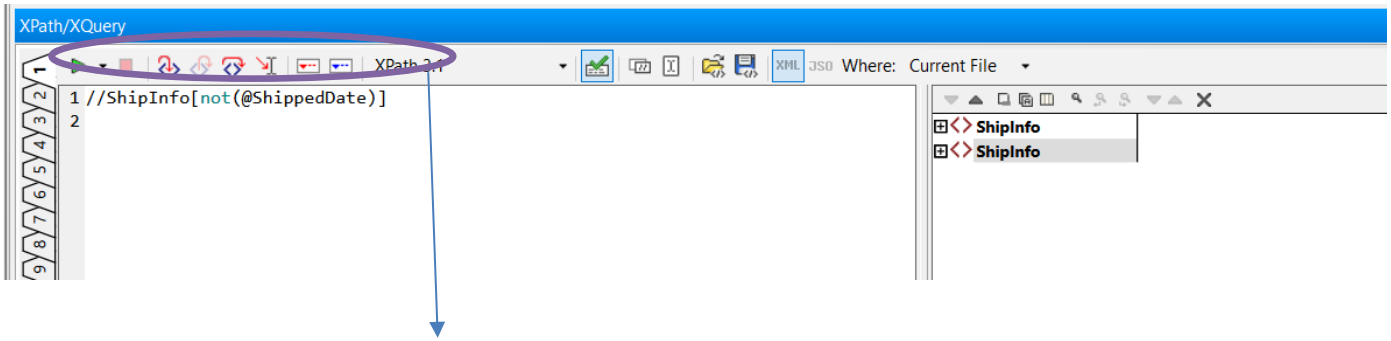
To get the customer ID attribute

Customer

Customer Id	Company Name	Contact Name	Contact Title	Phone	Fax	Address
GREAL	Great Lakes Food Market	Howard Snyder	Marketing Manager	(503) 555-7555		2732 Baker Blvd. Eugene OR 97403 USA
HUNGC	Hungry Coyote Import Store	Yoshi Latimer	Sales Representative	(503) 555-6874	(503) 555-2376	City Center Plaza 516 Main St. Elgin OR 97827 USA
LAZYK	Lazy K Kountry Store	John Steel	Marketing Manager	(509) 555-7969	(509) 555-6221	12 Orchestra Terrace Walla Walla WA 99362 USA
LETSS	Let's Stop N Shop	Jaime Yorres	Owner	(415) 555-5938		87 Polk St. Suite 5 San Francisco CA 94117 USA

Step 5 : XPath and XSLT

Write an XPATH query to show all orders that have not shipped yet (don't have "ShippedDate")



Omit the ShipInfo elements that don’t have the ShippedDate attribute

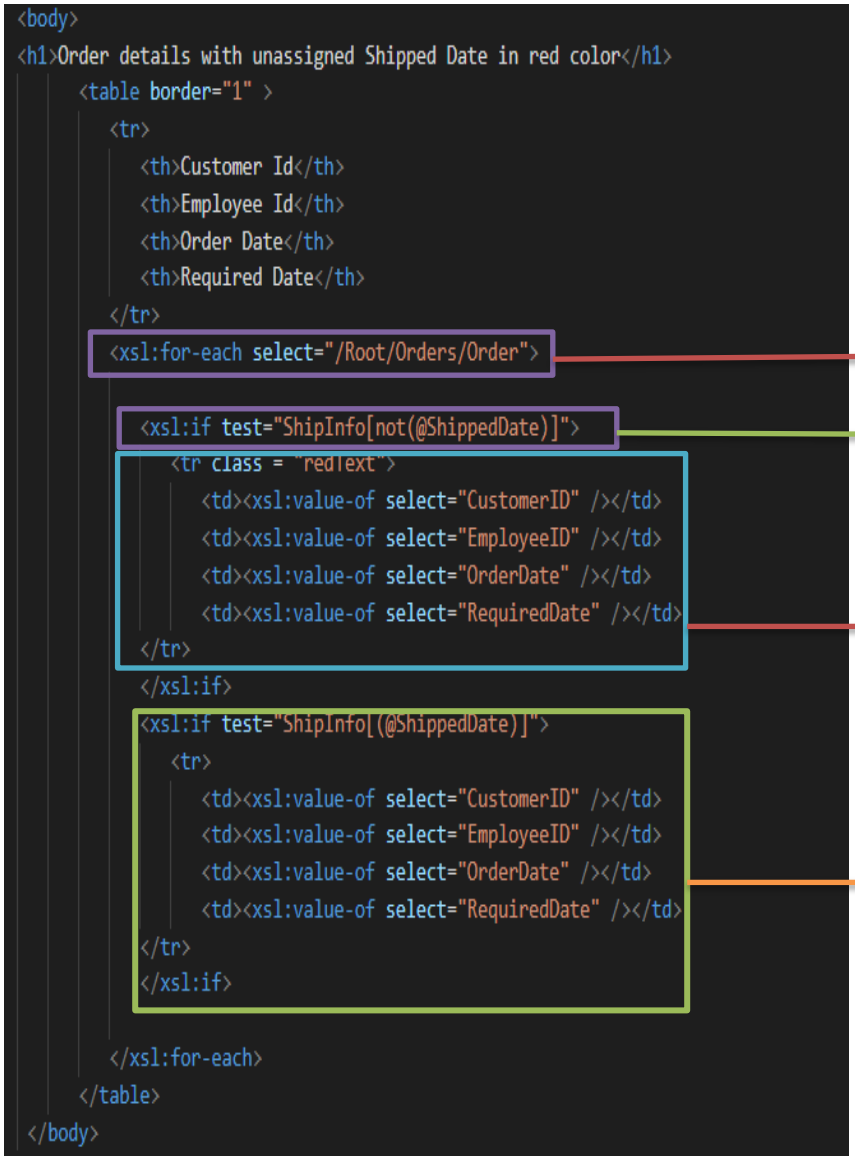
Design a new XSLT file that displays all "**Orders**" info (CustomerID, EmployeeID, OrderDate and RequiredDate), but displays these items (which you selected with XPATH) in red color.

CustOrder_Q5.xsl

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <html>
            <head>
                <title>Companies</title>
            </head>
            <style>
                .redText{
                    color: red;
                }
            </style>
            <body>
            <h1>Order details with unassigned Shipped Date in red color</h1>
                    <table border="1" >
                        <tr>
                            <th>Customer Id</th>
                            <th>Employee Id</th>
                            <th>Order Date</th>
                            <th>Required Date</th>
                        </tr>
                        <xsl:for-each select="/Root/Orders/Order">

                            <xsl:if test="ShipInfo[not(@ShippedDate)]">
                                <tr class = "redText">
                                    <td><xsl:value-of select="CustomerID" /></td>
                                    <td><xsl:value-of select="EmployeeID" /></td>
                                    <td><xsl:value-of select="OrderDate" /></td>
                                    <td><xsl:value-of select="RequiredDate" /></td>
                            </tr>
                            </xsl:if>
                            <xsl:if test="ShipInfo[(@ShippedDate)]">
                                <tr>
                                    <td><xsl:value-of select="CustomerID" /></td>
                                    <td><xsl:value-of select="EmployeeID" /></td>
                                    <td><xsl:value-of select="OrderDate" /></td>
                                    <td><xsl:value-of select="RequiredDate" /></td>
                            </tr>
                            </xsl:if>

                        </xsl:for-each>
                    </table>
                </body>
        </html>
    </xsl:template>

</xsl:stylesheet>

```

CustOrder_Q5.xml

CustOrder_Q5.xsl

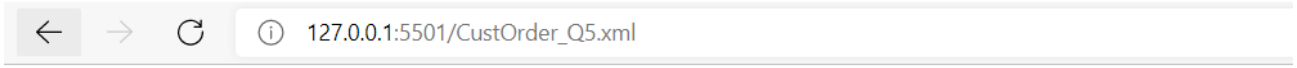


Go to the xpath /Root/Orders/Order tag and loop on each element

Check if the ShipInfo tag does not have the ShippedDate attribute

Apply class redText to the selective elements to display them in red font

If the elements have ShipDate attribute, then display as a normal font

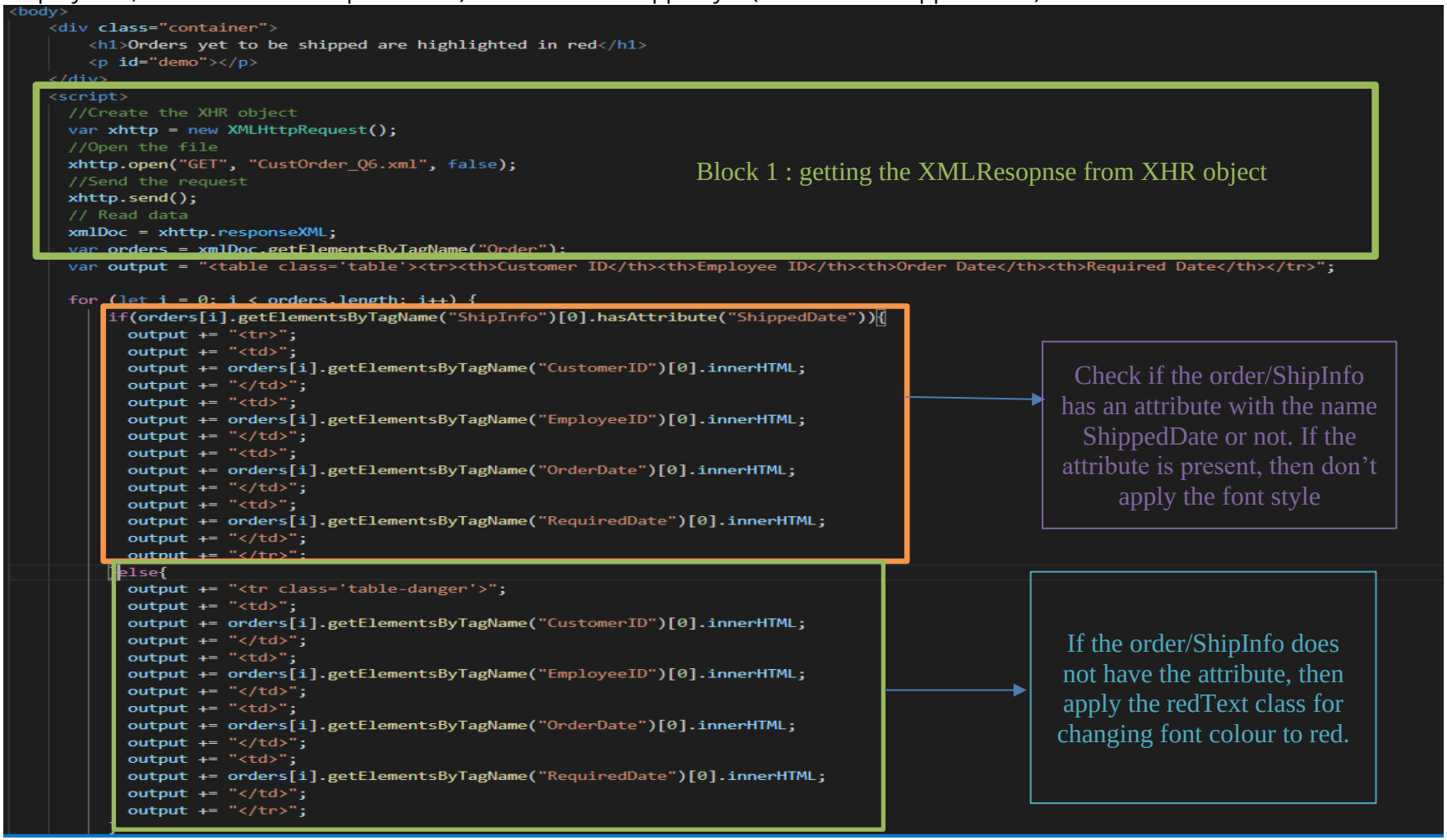


Order details with unassigned Shipped Date in red color

Customer Id	Employee Id	Order Date	Required Date
GREAL	6	1997-05-06T00:00:00	1997-05-20T00:00:00
GREAL	8	1997-07-04T00:00:00	1997-08-01T00:00:00
GREAL	1	1997-07-31T00:00:00	1997-08-28T00:00:00
GREAL	4	1997-07-31T00:00:00	1997-08-28T00:00:00
GREAL	6	1997-09-04T00:00:00	1997-10-02T00:00:00
GREAL	3	1997-09-25T00:00:00	1997-10-23T00:00:00
GREAL	4	1998-01-06T00:00:00	1998-02-03T00:00:00
GREAL	3	1998-03-09T00:00:00	1998-04-06T00:00:00
GREAL	3	1998-04-07T00:00:00	1998-05-05T00:00:00
GREAL	4	1998-04-22T00:00:00	1998-05-20T00:00:00
GREAL	4	1998-04-30T00:00:00	1998-06-11T00:00:00
HUNGC	3	1996-12-06T00:00:00	1997-01-03T00:00:00
HUNGC	1	1996-12-25T00:00:00	1997-01-22T00:00:00
HUNGC	3	1997-01-15T00:00:00	1997-02-12T00:00:00
HUNGC	4	1997-07-16T00:00:00	1997-08-13T00:00:00
HUNGC	8	1997-09-08T00:00:00	1997-10-06T00:00:00
LAZYK	1	1997-03-21T00:00:00	1997-04-18T00:00:00
LAZYK	8	1997-05-22T00:00:00	1997-06-19T00:00:00
LETSS	1	1997-06-25T00:00:00	1997-07-23T00:00:00
LETSS	8	1997-10-27T00:00:00	1997-11-24T00:00:00
LETSS	6	1997-11-10T00:00:00	1997-12-08T00:00:00
LETSS	4	1998-02-12T00:00:00	1998-03-12T00:00:00

Step 6: Use JavaScript to process XML data

You are going to redesign Step5 using JavaScript. You need Develop a proper JavaScript that shows all orders' info (CustomerID, EmployeeID, OrderDate and RequiredDate) that have not shipped yet (don't have "ShippedDate")




```
<body>
    <div class="container">
        <h1>Orders yet to be shipped are highlighted in red</h1>
        <p id="demo"></p>
    </div>
    <script>
      //Create the XHR object
      var xhttp = new XMLHttpRequest();
      //Open the file
      xhttp.open("GET", "CustOrder_Q6.xml", false);
      //Send the request
      xhttp.send();
      // Read data
      xmlDoc = xhttp.responseXML;
      var orders = xmlDoc.getElementsByTagName("Order");
      var output = "<table class='table'><tr><th>Customer ID</th><th>Employee ID</th><th>Order Date</th><th>Required Date</th></tr>";

      for (let i = 0; i < orders.length; i++) {
        if(orders[i].getElementsByTagName("ShipInfo")[0].hasAttribute("ShippedDate")){
          output += "<tr>";
          output += "<td>";
          output += orders[i].getElementsByTagName("CustomerID")[0].innerHTML;
          output += "</td>";
          output += "<td>";
          output += orders[i].getElementsByTagName("EmployeeID")[0].innerHTML;
          output += "</td>";
          output += "<td>";
          output += orders[i].getElementsByTagName("OrderDate")[0].innerHTML;
          output += "</td>";
          output += "<td>";
          output += orders[i].getElementsByTagName("RequiredDate")[0].innerHTML;
          output += "</td>";
          output += "</tr>";
        }else{
          output += "<tr class='table-danger'>";
          output += "<td>";
          output += orders[i].getElementsByTagName("CustomerID")[0].innerHTML;
          output += "</td>";
          output += "<td>";
          output += orders[i].getElementsByTagName("EmployeeID")[0].innerHTML;
          output += "</td>";
          output += "<td>";
          output += orders[i].getElementsByTagName("OrderDate")[0].innerHTML;
          output += "</td>";
          output += "<td>";
          output += orders[i].getElementsByTagName("RequiredDate")[0].innerHTML;
          output += "</td>";
          output += "</tr>";
        }
```

CustOrder_Q6.html

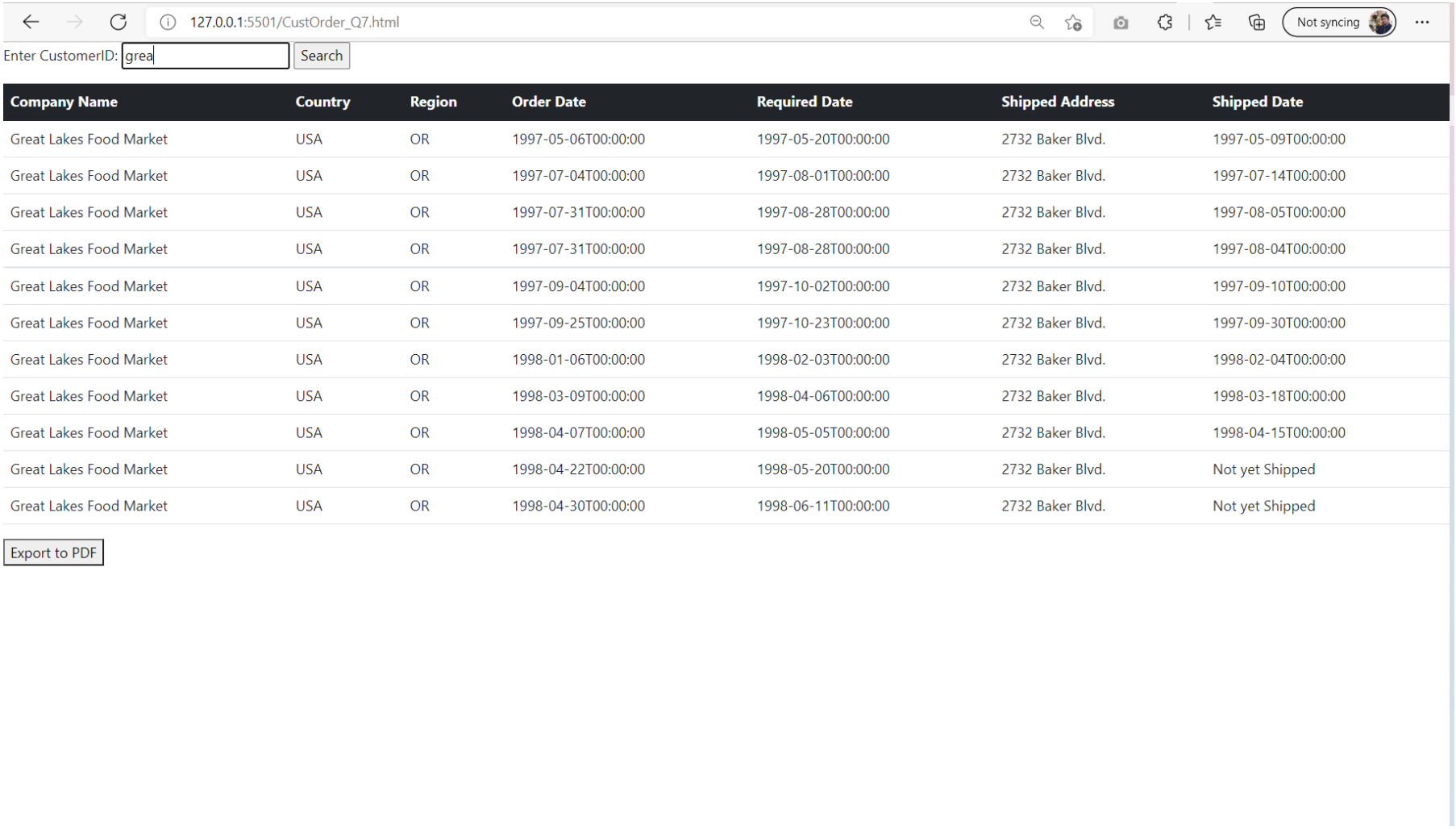
Orders yet to be shipped are highlighted in red

Customer ID	Employee ID	Order Date	Required Date
GREAL	6	1997-05-06T00:00:00	1997-05-20T00:00:00
GREAL	8	1997-07-04T00:00:00	1997-08-01T00:00:00
GREAL	1	1997-07-31T00:00:00	1997-08-28T00:00:00
GREAL	4	1997-07-31T00:00:00	1997-08-28T00:00:00
GREAL	6	1997-09-04T00:00:00	1997-10-02T00:00:00
GREAL	3	1997-09-25T00:00:00	1997-10-23T00:00:00
GREAL	4	1998-01-06T00:00:00	1998-02-03T00:00:00
GREAL	3	1998-03-09T00:00:00	1998-04-06T00:00:00
GREAL	3	1998-04-07T00:00:00	1998-05-05T00:00:00
GREAL	4	1998-04-22T00:00:00	1998-05-20T00:00:00
GREAL	4	1998-04-30T00:00:00	1998-06-11T00:00:00
HUNGC	3	1996-12-06T00:00:00	1997-01-03T00:00:00
HUNGC	1	1996-12-25T00:00:00	1997-01-22T00:00:00
HUNGC	3	1997-01-15T00:00:00	1997-02-12T00:00:00
HUNGC	4	1997-07-16T00:00:00	1997-08-13T00:00:00
HUNGC	8	1997-09-08T00:00:00	1997-10-06T00:00:00
LAZYK	1	1997-03-21T00:00:00	1997-04-18T00:00:00
LAZYK	8	1997-05-22T00:00:00	1997-06-19T00:00:00
LETSS	1	1997-06-25T00:00:00	1997-07-23T00:00:00
LETSS	8	1997-10-27T00:00:00	1997-11-24T00:00:00
LETSS	6	1997-11-10T00:00:00	1997-12-08T00:00:00
LETSS	4	1998-02-12T00:00:00	1998-03-12T00:00:00

Step 7: Using Step7.html, you need to develop a proper JavaScript function which will be invoked when user clicks on the "Search" button. The JavaScript function, reads the CustomerID from webform, and processes the CustOrder.xml data and displays all Customer info and related orders that their CustomerID is matched with the given CustomerID. (display these fields: CompanyName, Country, Region, OrderDate, RequiredDate, ShippedDate, ShipAddress)

- **Use your creativity to display Customer and Order data properly to look like a report.**

- **You need to add partial search to this program, so, if user enters "L", the program should displays CustomerID's has 'L' (or starts with 'L').**



Company Name	Country	Region	Order Date	Required Date	Shipped Address	Shipped Date
Great Lakes Food Market	USA	OR	1997-05-06T00:00:00	1997-05-20T00:00:00	2732 Baker Blvd.	1997-05-09T00:00:00
Great Lakes Food Market	USA	OR	1997-07-04T00:00:00	1997-08-01T00:00:00	2732 Baker Blvd.	1997-07-14T00:00:00
Great Lakes Food Market	USA	OR	1997-07-31T00:00:00	1997-08-28T00:00:00	2732 Baker Blvd.	1997-08-05T00:00:00
Great Lakes Food Market	USA	OR	1997-07-31T00:00:00	1997-08-28T00:00:00	2732 Baker Blvd.	1997-08-04T00:00:00
Great Lakes Food Market	USA	OR	1997-09-04T00:00:00	1997-10-02T00:00:00	2732 Baker Blvd.	1997-09-10T00:00:00
Great Lakes Food Market	USA	OR	1997-09-25T00:00:00	1997-10-23T00:00:00	2732 Baker Blvd.	1997-09-30T00:00:00
Great Lakes Food Market	USA	OR	1998-01-06T00:00:00	1998-02-03T00:00:00	2732 Baker Blvd.	1998-02-04T00:00:00
Great Lakes Food Market	USA	OR	1998-03-09T00:00:00	1998-04-06T00:00:00	2732 Baker Blvd.	1998-03-18T00:00:00
Great Lakes Food Market	USA	OR	1998-04-07T00:00:00	1998-05-05T00:00:00	2732 Baker Blvd.	1998-04-15T00:00:00
Great Lakes Food Market	USA	OR	1998-04-22T00:00:00	1998-05-20T00:00:00	2732 Baker Blvd.	Not yet Shipped
Great Lakes Food Market	USA	OR	1998-04-30T00:00:00	1998-06-11T00:00:00	2732 Baker Blvd.	Not yet Shipped

Export to PDF

Code:

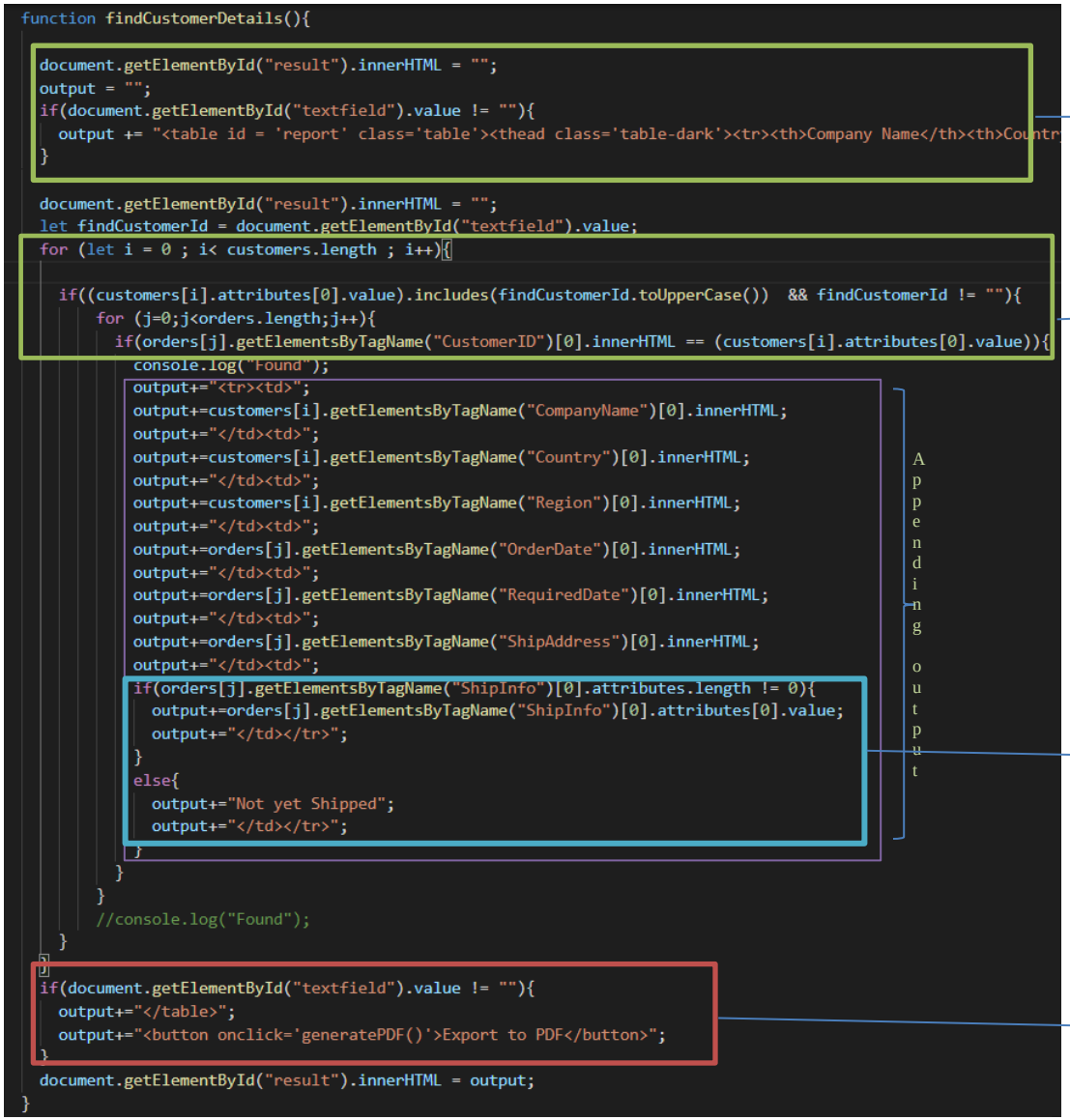
```
<p>
  <label for="textfield">Enter CustomerID:</label>
  <input type="text" name="textfield" id="textfield" onkeyup="findCustomerDetails()">
  <input type="button" name="button" id="button" value="Search" onclick="findCustomerDetails()">
</p>
<p id="result"></p>
```

Creating The Skeleton And Adding Callback Functions To Generate Report

```
<script>
  //Create the XHR object
  var xhttp = new XMLHttpRequest();
  //Open the file
  xhttp.open("GET", "CustOrder_Q7.xml", false);
  //Send the request
  xhttp.send();
  // Read data
  xmlDoc = xhttp.responseXML;

  var customers = xmlDoc.getElementsByTagName("Customer");
  var orders = xmlDoc.getElementsByTagName("Order");

  var output = "";
```

Fetching the XML file and storing its data into variables

Functions:



```
function findCustomerDetails(){
  document.getElementById("result").innerHTML = "";
  output = "";
  if(document.getElementById("textfield").value != ""){
    output += "<table id = 'report' class='table'><thead class='table-dark'><tr><th>Company Name</th><th>Countr
  }
  document.getElementById("result").innerHTML = "";
  let findCustomerId = document.getElementById("textfield").value;
  for (let i = 0 ; i< customers.length ; i++){
    if((customers[i].attributes[0].value).includes(findCustomerId.toUpperCase())  && findCustomerId != ""){
      for (j=0;j<orders.length;j++){
        if(orders[j].getElementsByTagName("CustomerID")[0].innerHTML == (customers[i].attributes[0].value)){
          console.log("Found");
          output+="<tr><td>";
          output+=customers[i].getElementsByTagName("CompanyName")[0].innerHTML;
          output+="</td><td>";
          output+=customers[i].getElementsByTagName("Country")[0].innerHTML;
          output+="</td><td>";
          output+=customers[i].getElementsByTagName("Region")[0].innerHTML;
          output+="</td><td>";
          output+=orders[j].getElementsByTagName("OrderDate")[0].innerHTML;
          output+="</td><td>";
          output+=orders[j].getElementsByTagName("RequiredDate")[0].innerHTML;
          output+="</td><td>";
          output+=orders[j].getElementsByTagName("ShipAddress")[0].innerHTML;
          output+="</td><td>";
          if(orders[j].getElementsByTagName("ShipInfo")[0].attributes.length != 0){
            output+=orders[j].getElementsByTagName("ShipInfo")[0].attributes[0].value;
            output+="</td></tr>";
          }
          else{
            output+="Not yet Shipped";
            output+="</td></tr>";
          }
        }
      }
      //console.log("Found");
    }
  if(document.getElementById("textfield").value != ""){
    output+="</table>";
    output+="<button onclick='generatePDF()'>Export to PDF</button>";
  }
  document.getElementById("result").innerHTML = output;
}
```

Appending output

Set the element with result id to blank whenever the function is called.
Set the output text to blank whenever the function is called.
If the search input is not empty, only then add table element into output field

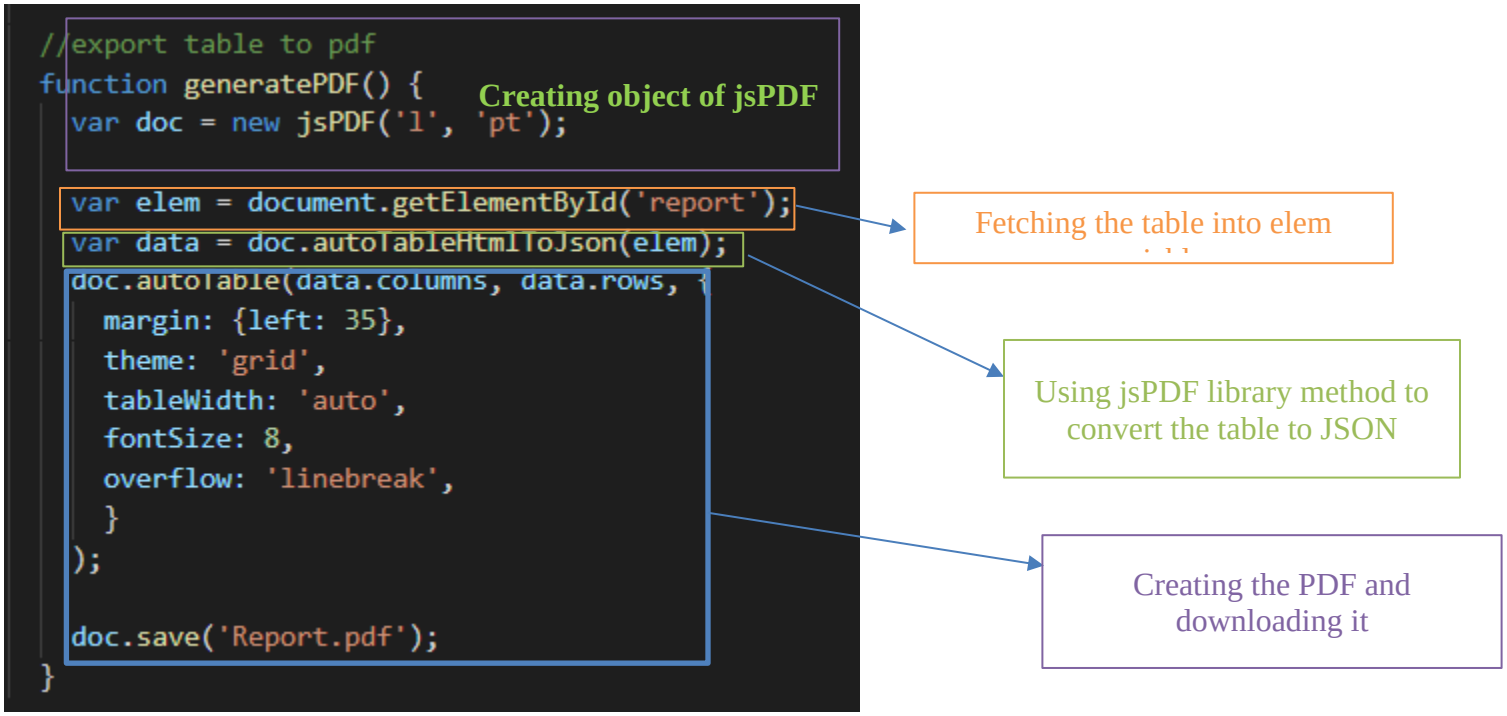
Loop through the customers array where customer data is saved
If the customerId attribute of the current customer in loop includes the search id and if the search id is not blank
Then,
Inner loop orders array and check if any order has a customerId tag whose value matches to outer loop current customer, if yes then populate table by appending text to output string.

Check if order has a shipped date by checking length of attributes of ShipInfo tag inside Order tag, if not then populate table with Not yet shipped.

If there is a search string present, only then append the option for export report to pdf

GeneratePDF function:

References : https://codingshiksha.com/javascript/jspdf-tutorial-to-export-html-table-to-excel-and-pdf-document-using-jspdf-autotable-library-in-javascript-full-tutorial-for-beginners/



Step 8:

Modify the CustOrder.xml by removing the element (as a child of) and add it as attribute to the element. save it as order_modified.xml:

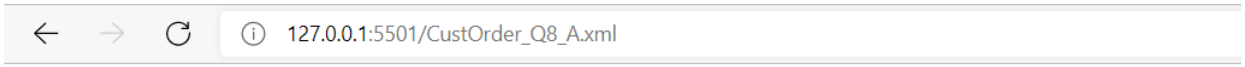
CustOrder_Q8_A.xml

Step 5 on the modified XML

```
1  <xsl:for-each select="/Root/Orders/Order">
2
3                     <xsl:if test="ShipInfo[not(@ShippedDate)]">
4                        <tr class = "redText">
5                           <td><xsl:value-of select="CustomerID" /></td>
6                           <td><xsl:value-of select="EmployeeID" /></td>
7                           <td><xsl:value-of select="OrderDate" /></td>
8                           <td><xsl:value-of select="RequiredDate" /></td>
9                     </tr>
10                    </xsl:if>
11                    <xsl:if test="ShipInfo[(@ShippedDate)]">
12                       <tr>
13                          <td><xsl:value-of select="CustomerID" /></td>
14                          <td><xsl:value-of select="EmployeeID" /></td>
15                          <td><xsl:value-of select="OrderDate" /></td>
16                          <td><xsl:value-of select="RequiredDate" /></td>
17                    </tr>
18                    </xsl:if>
19
20                </xsl:for-each>
```

```
1  <xsl:for-each select="/Root/Orders/Order">
2
3                     <xsl:if test="ShipInfo[not(@ShippedDate)]">
4                        <tr class = "redText">
5                           <td><xsl:value-of select="./@CustomerID" /></td>
6                           <td><xsl:value-of select="EmployeeID" /></td>
7                           <td><xsl:value-of select="OrderDate" /></td>
8                           <td><xsl:value-of select="RequiredDate" /></td>
9                     </tr>
10                    </xsl:if>
11                    <xsl:if test="ShipInfo[(@ShippedDate)]">
12                       <tr>
13                          <td><xsl:value-of select="./@CustomerID" /></td>
14                          <td><xsl:value-of select="EmployeeID" /></td>
15                          <td><xsl:value-of select="OrderDate" /></td>
16                          <td><xsl:value-of select="RequiredDate" /></td>
17                    </tr>
18                    </xsl:if>
19
20                </xsl:for-each>
```

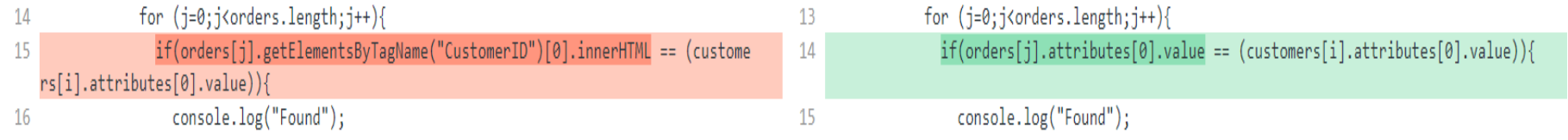
The modified xsl is the one on the right hand side in the image above. These changes were made to display the content when CustomerID is made an attribute of Order.



Order details with unassigned Shipped Date in red color

Customer Id	Employee Id	Order Date	Required Date
GREAL	6	1997-05-06T00:00:00	1997-05-20T00:00:00
GREAL	8	1997-07-04T00:00:00	1997-08-01T00:00:00
GREAL	1	1997-07-31T00:00:00	1997-08-28T00:00:00
GREAL	4	1997-07-31T00:00:00	1997-08-28T00:00:00
GREAL	6	1997-09-04T00:00:00	1997-10-02T00:00:00
GREAL	3	1997-09-25T00:00:00	1997-10-23T00:00:00
GREAL	4	1998-01-06T00:00:00	1998-02-03T00:00:00
GREAL	3	1998-03-09T00:00:00	1998-04-06T00:00:00
GREAL	3	1998-04-07T00:00:00	1998-05-05T00:00:00
GREAL	4	1998-04-22T00:00:00	1998-05-20T00:00:00
GREAL	4	1998-04-30T00:00:00	1998-06-11T00:00:00
HUNGC	3	1996-12-06T00:00:00	1997-01-03T00:00:00
HUNGC	1	1996-12-25T00:00:00	1997-01-22T00:00:00
HUNGC	3	1997-01-15T00:00:00	1997-02-12T00:00:00
HUNGC	4	1997-07-16T00:00:00	1997-08-13T00:00:00
HUNGC	8	1997-09-08T00:00:00	1997-10-06T00:00:00
LAZYK	1	1997-03-21T00:00:00	1997-04-18T00:00:00
LAZYK	8	1997-05-22T00:00:00	1997-06-19T00:00:00
LETSS	1	1997-06-25T00:00:00	1997-07-23T00:00:00
LETSS	8	1997-10-27T00:00:00	1997-11-24T00:00:00
LETSS	6	1997-11-10T00:00:00	1997-12-08T00:00:00
LETSS	4	1998-02-12T00:00:00	1998-03-12T00:00:00

Step 7 on the modified XML



The changes that were made was to replace getElementsByTagName("CustomerID")[0].innerHTML to attributes[0].value

Our opinion

Attribute vs Element:
While parsing the XML we have used both attributes and elements to fetch our data. Our mutual understanding is that element tag names are much handy and robust to use while parsing XML. These are JS functions that help us find an element object irrespective of its cascading. Contrary to that, while fetching attributes, we can get errors if the attribute position is not mentioned correctly. If the attribute is not mandatory and is missed for an element, a proper exception handling must be done to prevent runtime errors.

JS vs XSLT:
Definitely JS has the winner point here, provided the ease of syntax usage. JS also offers multiple internal libraries that makes data manipulation much easier. Added to that, we also have the flexibility to add external libraries easily. Though libraries can be added in an XSL as well but the less amount of code and syntactical ethics makes JS stand out in terms of accessibility.

Enter CustomerID: grea Search

Company Name	Country	Region	Order Date	Required Date	Shipped Address	Shipped Date
Great Lakes Food Market	USA	OR	1997-05-06T00:00:00	1997-05-20T00:00:00	2732 Baker Blvd.	1997-05-09T00:00:00
Great Lakes Food Market	USA	OR	1997-07-04T00:00:00	1997-08-01T00:00:00	2732 Baker Blvd.	1997-07-14T00:00:00
Great Lakes Food Market	USA	OR	1997-07-31T00:00:00	1997-08-28T00:00:00	2732 Baker Blvd.	1997-08-05T00:00:00
Great Lakes Food Market	USA	OR	1997-07-31T00:00:00	1997-08-28T00:00:00	2732 Baker Blvd.	1997-08-04T00:00:00
Great Lakes Food Market	USA	OR	1997-09-04T00:00:00	1997-10-02T00:00:00	2732 Baker Blvd.	1997-09-10T00:00:00
Great Lakes Food Market	USA	OR	1997-09-25T00:00:00	1997-10-23T00:00:00	2732 Baker Blvd.	1997-09-30T00:00:00
Great Lakes Food Market	USA	OR	1998-01-06T00:00:00	1998-02-03T00:00:00	2732 Baker Blvd.	1998-02-04T00:00:00
Great Lakes Food Market	USA	OR	1998-03-09T00:00:00	1998-04-06T00:00:00	2732 Baker Blvd.	1998-03-18T00:00:00
Great Lakes Food Market	USA	OR	1998-04-07T00:00:00	1998-05-05T00:00:00	2732 Baker Blvd.	1998-04-15T00:00:00
Great Lakes Food Market	USA	OR	1998-04-22T00:00:00	1998-05-20T00:00:00	2732 Baker Blvd.	Not yet Shipped
Great Lakes Food Market	USA	OR	1998-04-30T00:00:00	1998-06-11T00:00:00	2732 Baker Blvd.	Not yet Shipped

Export to PDF

Summary

This project is a joint effort of me and my team member Mr. Rohan Vasudev Patel.

We divided the work equally among ourselves.

The first three steps and 8th step have been solved by Rohan:

Step 1) Try to explore and understand the structure of the given XML data. Is the XML file a well-form document? Is it a valid document? Is there any namespace? .

Step 2) Design the DTD for this document and link it to the given XML file.

Step 3) Design XML Schema for this document and link it to the given XML file.

Step 8) Complete the followings:

- Modify the CustOrder.xml by removing the <CustomerID> element (as a child of <Order>) and add it as attribute to the <order> element. save it as order_modified.xml:
- Based on the new xml data file, redo the Step 5 and 7.
- From your understanding as developer, explain your idea about coding/processing XML element vs attribute. Which one is easier to develop with JS or XSLT, process CustomerID as element-data or attribute-data? Explain your answer based on your observation through this project.

The 4th, 5th, 6th and 7th step have been solved by Sandeep :

Step 4) Write an XSLT document to transform order.xml into an HTML document that displays all "Customer's Data" in a HTML table format.

Step 5) Write an XPATH query to show all orders that have not shipped yet (don't have "ShippedDate")
Design a new XSLT file that displays all "Orders" info (CustomerID, EmployeeID, OrderDate and RequiredDate), but displays these items (which you selected with XPATH) in red color.

Step 6) You are going to redesign Step5 using JavaScript. You need Develop a proper JavaScript that shows all orders' info (CustomerID, EmployeeID, OrderDate and RequiredDate) that have not shipped yet (don't have "ShippedDate")

Step 7) Using Step7.html, you need to develop a proper JavaScript function which will be invoked when user clicks on the “Search” button. The JavaScript function, reads the CustomerID from webform, and processes the CustOrder.xml data and displays all Customer info and related orders that their CustomerID is matched with the given CustomerID. (display these fields: CompanyName, Country, Region, OrderDate, RequiredDate, ShippedDate, ShipAddress)

- Use your creativity to display Customer and Order data properly to look like a report.
- You need to add partial search to this program, so, if user enters “L”, the program should displays CustomerID’s has ‘L’ (or starts with ‘L’).

```
/**************************************************************************
* ITC5202 – Project1 * We declare that this assignment is our own work in accordance with
Humber Academic * Policy.
* No part of this assignment has been copied manually or electronically from any other *
source (including web sites) or distributed to other students. *
*    Names: Sandeep Das, Rohan Vasudev Patel
*    Student ID: N01472825 , N01469929
*    Date: 28th Feb 2022
* * ************************
```