ITWS-4250/6250 Database Applications and Systems

Lab 9

**Instructions:**

* In this lab, you will apply the concepts of unstructured and semi-structured data to create XML files. There are a total of 4 questions, and you will use the same question setup to answer the first 3 questions.
* You will have roughly 90 minutes during the scheduled lecture period to complete it, but it’s not due on LMS until **EOD tonight**.
* Before getting started, you should have a repo setup and have both Sindhu ([mandas@rpi.edu](mailto:mandas@rpi.edu)) and me rplotka ([rplotka@rpi.edu](mailto:rplotka@rpi.edu)) as collaborators. The repos must be private.
* Create folders for each HW, Lab, and other assignments as they come up. Remember to push your repo before submitting your folder zip to LMS for credit.
* Make sure all deliverables are in the folders as instructed in the assignments and include a readme file in each folder with any information you feel may be relevant.
* You may consult with your groupmates, but you must do and submit your own work.

**Questions:**

1. Create a well-formed XML document containing student details. Students should have attributes like their id, name, address, email, and university name. You must include at least 3 records in your answer. You can make any other assumption as needed.

<?xml version="1.0" encoding="UTF-8" ?>

<student id=”112”>

<name>

<first>Aly</first>

<last>Sha</last>

</name>

<address>142 Boutin Rd., Troy, NY</address>

<email>alySha@gmai.com</email>

<university>Rensselaer Polytechnic Institute</university>

</ student >

<student id=”113”>

<name>

<first> Bab </first>

<last> Lana </last>

</name>

<address>149 Boutin Rd., Troy, NY</address>

<email>bobLan@gmai.com</email>

<university>Rensselaer Polytechnic Institute</university>

</ student >

<student id=”13”>

<name>

<first> Caren </first>

<last> Kiffin </last>

</name>

<address>12 Boutin Rd., Troy, NY</address>

<email>carenK@gmai.com</email>

<university>Rensselaer Polytechnic Institute</university>

</ student >

1. Use Document Type Definition (DTD) to define the structure, legal elements, and attributes of the student XML created in the previous question.

<!DOCTYPE Students[  
<!ELEMENT Students (Student+)>

<!ELEMENT Student (id, name, address, email, university)>  
<!ELEMENT id (#PCDATA) #REQUIRED>

<!ELEMENT name (first, last)>  
<!ELEMENT first (#PCDATA) #REQUIRED>

<!ELEMENT last (#PCDATA) #REQUIRED>  
<!ELEMENT address (#PCDATA)>  
<!ELEMENT email (#PCDATA)>  
<!ELEMENT university (#PCDATA)>  
  
]>

<student id=”112”>

<name>

<first>Aly</first>

<last>Sha</last>

</name>

<address>142 Boutin Rd., Troy, NY</address>

<email>alySha@gmai.com</email>

<university>Rensselaer Polytechnic Institute</university>

</ student >

<student id=”113”>

<name>

<first> Bab </first>

<last> Lana </last>

</name>

<address>149 Boutin Rd., Troy, NY</address>

<email>bobLan@gmai.com</email>

<university>Rensselaer Polytechnic Institute</university>

</ student >

<student id=”13”>

<name>

<first> Caren </first>

<last> Kiffin </last>

</name>

<address>12 Boutin Rd., Troy, NY</address>

<email>carenK@gmai.com</email>

<university>Rensselaer Polytechnic Institute</university>

</ student >

1. Create an XSL stylesheet to show all the student records. Use appropriate tags and elements as needed.

<?xml version="1.0"?>  
  
<xsl:stylesheet version="1.0"  
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">

<html>

<body>

<h2>Student Info</h2>

<table border="1">

<tr bgcolor="#9acd32">

<th>id</th>

<th>name</th>

<th>address</th>

<th>email</th>

<th>university</th>

</tr>

<xsl:for-each select="student">

<tr>

<td><xsl:value-of select="id"/></td>

<td><xsl:value-of select="name"/></td>

<td><xsl:value-of select=" address "/></td>

<td><xsl:value-of select=" email "/></td>

<td><xsl:value-of select=" university "/></td>

</tr>

</xsl:for-each>

</table>

</body>

</html>

</xsl: stylesheet >

1. Consider the following XML code:

<moviestore>

    <!-- a movie store database -->

    <movie id=“M101” category=“fiction”>

        <!-- a particular movie -->

        <title language=“English”>Star Wars</title>

        <price unit=“USD”>9.99</price>

    </movie>

    <movie id=“M102” category=“fiction”>

        <title language=“English”>Harry Potter</title>

        <price unit=“USD”>10.49</price>

    </movie>

    <movie id=“M103” category=“thriller”>

        <title language=“Spanish”>Prison Break</title>

        <price unit=“USD”>4.99</price>

    </movie>

    <movie id=“M104” category=“thriller”>

        <title language=“English”>Breaking Bad</title>

        <price unit=“USD”>14.99</price>

    </movie>

</moviestore>

Write XPath queries to find answers to the following questions:

1. Find the title and price of non-fiction movies with a price of more than $5

//moviestore/movie[@category!=”fiction” and price < 5]/title | // moviestore/movie [@category!=”fiction” and price < 5]/price

1. Find the average price of all fiction movies

Sum(//moviestore/movie[@category=”fiction”]/price/number(text())) div count(//moviestore/movie[@category=”fiction”]/price)

1. Find titles of all the thriller movies which are related to ‘Prison’

/moviestore/movie[@category!=” thriller” and contains(title,”Prison”)/title/text()]