Database Systems Lab(MCSE506P)

EXERCISE 9: XML, XSD, XQUERY

Name: Mahesh Jagtap Reg No:24MCS1017 Date: 04/10/2024

- 1. Create three XML documents for:
 - a. DEPARTMENT (DEPT_ID, DEPT_NAME).
 - b. PROJECT (PROJECT_ID, PROJECT_NAME, DID).
 - c. EMPLOYEE (EMP_ID, NAME, GENDER, SALARY, DEPT_ID, PID, DOJ).

Each XML should have 5 nodes at least.

```
department.xml->
```

```
<?xml version="1.0" encoding="UTF-8"?>
<DEPARTMENT MCS1017>
   <DEPARTMENT>
      <DEPT ID>D001
T ID>
      <DEPT NAME>Human Resources
   </DEPARTMENT>
   <DEPARTMENT>
      <DEPT ID>D002
T ID>
      <DEPT NAME>Finance
   <DEPARTMENT>
      <DEPT ID>D003
/DEPT ID>
      <DEPT NAME>IT Support
   </DEPARTMENT>
   <DEPARTMENT>
      <DEPT ID>D004/DEPT ID>
      <DEPT NAME>Marketing/DEPT NAME>
   </DEPARTMENT>
   <DEPARTMENT>
      <DEPT ID>D005/DEPT ID>
      <DEPT NAME>Research and Development
```

```
<?xml version="1.0" encoding="UTF-8"?>
employee.xml->
<EMPLOYEE MCS1017>
   <EMPLOYEE>
       <EMP ID>E2000/EMP ID>
       <NAME>John Doe</NAME>
       <GENDER>Male</GENDER>
       <SALARY>75000</SALARY>
       <DEPT ID>D001
T ID>
       <PID>1001</PID>
       <DOJ>2022-05-15</DOJ>
    </EMPLOYEE>
    <EMPLOYEE>
       <EMP ID>E2001/EMP ID>
       <NAME>Jane Smith</NAME>
       <GENDER>Female</GENDER>
       <SALARY>65000</SALARY>
       <DEPT ID>D002
T ID>
       <PID>1002</PID>
       <DOJ>2023-04-10</DOJ>
    </EMPLOYEE>
    <EMPLOYEE>
       <EMP ID>E2002/EMP ID>
       <NAME>Mike Brown</NAME>
       <GENDER>Male</GENDER>
       <SALARY>55000</SALARY>
       <DEPT ID>D003 ID>
       <PID>1003</PID>
       <DOJ>2023-07-20</DOJ>
    </EMPLOYEE>
    <EMPLOYEE>
       <EMP ID>E2003/EMP ID>
       <NAME>Emily Wilson</NAME>
       <GENDER>Female</GENDER>
       <SALARY>72000</SALARY>
       <DEPT ID>D004 ID>
```

```
<PID>1004</PID>
       <DOJ>2022-09-05</DOJ>
    </EMPLOYEE>
    <EMPLOYEE>
       <EMP ID>E2004/EMP ID>
       <NAME>Chris Johnson</NAME>
       <GENDER>Male</GENDER>
       <SALARY>80000</SALARY>
       <DEPT ID>D005/DEPT ID>
       <PID>1005</PID>
       <DOJ>2021-12-30</DOJ>
    </EMPLOYEE>
</EMPLOYEE MCS1017>
project.xml
<?xml version="1.0" encoding="UTF-8"?>
<PROJECT MCS1017>
    <PROJECT>
       <PROJECT ID>P101/project_ID>
       <PROJECT NAME>GITHUB
       <DID>1</DID>
    </PROJECT>
    <PROJECT>
       <PROJECT ID>P102/PROJECT ID>
       <PROJECT NAME>CODEMASTER
       <DID>2</DID>
    </PROJECT>
    <PROJECT>
       project id>p103/project_id>
       <PROJECT NAME>VSCODE
       <DID>4</DID>
    </PROJECT>
    <PROJECT>
       <PROJECT ID>P104/PROJECT ID>
       <PROJECT NAME>AWS/PROJECT NAME>
       <DID>3</DID>
```

2. Create suitable XSD files for the three XML documents and validate the same.

schema.xsd

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="DEPARTMENT_MCS1017">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="DEPARTMENT" maxOccurs="unbounded">
         <xs:complexType>
          <xs:sequence>
            <xs:element name="DEPT_ID" type="xs:string"/>
            <xs:element name="DEPT_NAME" type="xs:string"/>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
 <xs:element name="PROJECT_MCS1017">
   <xs:complexType>
     <xs:sequence>
       <xs:element name="PROJECT" maxOccurs="unbounded">
         <xs:complexType>
           <xs:sequence>
            <xs:element name="PROJECT_ID" type="xs:string"/>
            <xs:element name="PROJECT_NAME" type="xs:string"/>
            <xs:element name="DID" type="xs:string"/>
           </xs:sequence>
         </xs:complexType>
       </xs:element>
```

```
</xs:sequence>
   </xs:complexType>
 </xs:element>
 <xs:element name="EMPLOYEE_MCS1017">
   <xs:complexType>
     <xs:sequence>
      <xs:element name="EMPLOYEE" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
           <xs:element name="EMP_ID" type="xs:string"/>
           <xs:element name="NAME" type="xs:string"/>
           <xs:element name="GENDER" type="xs:string"/>
           <xs:element name="SALARY" type="xs:decimal"/>
           <xs:element name="DEPT_ID" type="xs:string"/>
           <xs:element name="PID" type="xs:string"/>
           <xs:element name="DOJ" type="xs:date"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
     </xs:sequence>
   </xs:complexType>
 </xs:element>
</xs:schema>
validate.java
import java.io.File;
import javax.xml.XMLConstants;
import javax.xml.transform.stream.StreamSource;
import javax.xml.validation.Schema;
import javax.xml.validation.SchemaFactory;
import javax.xml.validation.Validator;
public class XSDValidator {
    public static void main(String[] args) {
         String xsdPath = "C:/Users/Jagta/OneDrive/Desktop/VIT
SEM1/DBMS/LAB ASSIGNMENTS/schema.xsd";
         String[] xmlPaths = {
              "C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB
ASSIGNMENTS/department.xml",
```

```
"C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB
ASSIGNMENTS/employee.xml",
            "C:/Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS/LAB
ASSIGNMENTS/project.xml"
        };
        for (String xmlPath : xmlPaths) {
            validateXMLSchema(xsdPath, xmlPath);
        }
    }
   public static void validateXMLSchema (String xsdPath, String
xmlPath) {
        try {
            SchemaFactory factory =
SchemaFactory.newInstance(XMLConstants.W3C XML SCHEMA NS URI);
            Schema schema = factory.newSchema(new File(xsdPath));
            Validator validator = schema.newValidator();
            validator.validate(new StreamSource(new
File(xmlPath)));
            System.out.println(xmlPath + " is valid against " +
xsdPath);
        } catch (Exception e) {
            System.out.println(xmlPath + " is not valid. Error: "
+ e.getMessage());
        }
    }
}
```

OUTPUT:

```
C:\Users\Jagta\OneDrive\Desktop\VIT SEM1\DBMS\LAB ASSIGNMENTS>javac XSDValidator.java

C:\Users\Jagta\OneDrive\Desktop\VIT SEM1\DBMS\LAB ASSIGNMENTS>java XSDValidator

C:\Users/Jagta/OneDrive/Desktop/VIT SEM1/DBMS\LAB ASSIGNMENTS/department.xml is valid against C:\Users/Jagta/OneDrive/Desktop/VIT SEM1/

DBMS\LAB ASSIGNMENTS\schema.xsd

C:\Users\Jagta\OneDrive\Desktop\VIT SEM1\DBMS\LAB ASSIGNMENTS\employee.xml is valid against C:\Users\Jagta\OneDrive\Desktop\VIT SEM1/DB

MS\LAB ASSIGNMENTS\schema.xsd

C:\Users\Jagta\OneDrive\Desktop\VIT SEM1\DBMS\LAB ASSIGNMENTS\project.xml is valid against C:\Users\Jagta\OneDrive\Desktop\VIT SEM1\DBM

S\LAB ASSIGNMENTS\schema.xsd
```

- 3. Perform the following operations:
- a. Retrieve the department name for department id D001.

for \$dept in //DEPARTMENT[DEPT_ID='D001'] return \$dept/DEPT_NAME



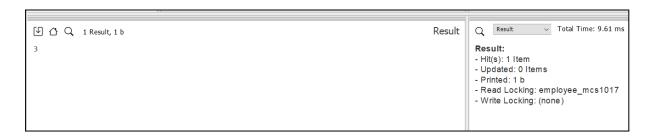
b. Retrieve the project name and department id of project id P102.

for \$proj in //PROJECT[PROJECT_ID='P102']
return (concat(\$proj/PROJECT_NAME, ' - ', \$proj/DEPT_ID))



c. Retrieve the count of male employees.

count(//EMPLOYEE[GENDER='Male'])



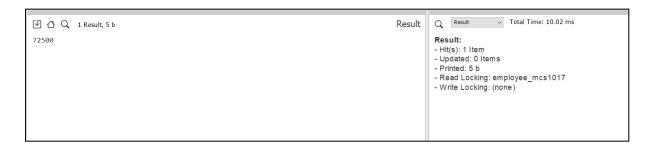
d. Retrieve the names and dept ids of employees who joined after 01-Jan-2020.

for \$emp in //EMPLOYEE[xs:date(DOJ) > xs:date('2020-01-01')] return concat(\$emp/NAME, ' - ', \$emp/DEPT_ID)

☑ △ Q 5 Results, 96 b	Result	Q Result V Total Time: 13.03 ms
John Doe - D001 Jane Smith - D002 Mike Brown - D003 Emily Wilson - D004 Chris Johnson - D005		Result: - Hit(s): 5 Items - Updated: 0 Items - Printed: 96 b - Read Locking: (none) - Write Locking: (none)

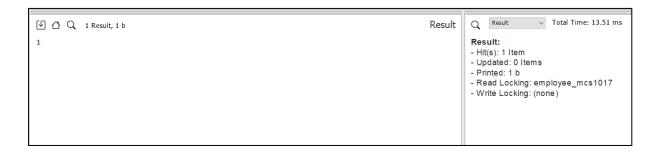
e. Find the average salary of employees belonging to PID P1002.

let \$salaries := //EMPLOYEE[PID='P1002']/SALARY
return avg(\$salaries)



f. How many male employees work for dept id D001?

count(//EMPLOYEE[GENDER='Male' and DEPT_ID='D001'])



g. List the names of employees whose salary ranges between 60000 to 70000.

for \$emp in //EMPLOYEE[SALARY >= 60000 and SALARY <= 70000] return \$emp/NAME

 	Result, 23 b	esult	Q	Result	~	Total Time: 59.	.52 ms
<name>Jane Sm</name>	mith		- Up - Prii - Re	(s): 1 Item dated: 0 Ite nted: 23 b	ems g:emp	ployee_mcs10 ne))17