Bookmark this page

These exercises are a bit more difficult overall than the Course-Catalog XPath and XQuery Exercises. We suggest you work those problems first.

In these exercises, you will be working with a small XML data set about world countries. This data is adapted from the Mondial 3.0 database as hosted by the University of Washington, and was originally compiled by the Georg-August University of Goettingen Institute for Informatics. Each country has a name, population, and area (in sq. km). Some countries also list languages (with percentages of the population that speaks each language) and/or cities (with names and populations). The XML data is here.

Instructions: Each problem asks you to write a query in XPath or XQuery. When you click "Check Answer" our back-end runs your query against the sample database using Saxon. It displays the result and compares your answer against the correct one. When you're satisfied with your solution for a given problem, click the "Submit" button to check your answer.

You may perform these exercises as many times as you like, so we strongly encourage you to keep working with them until you complete the exercises with full credit.

Q1

1.0/1.0 point (graded)
Return the area of Mongolia.

Note: Your solution will need to reference *doc("countries.xml")* to access the data.
Reminder: To return the value of an attribute *attr*, you must use *data(@attr)*, although just *@attr* may be used in comparisons. You will need to remember this for some later questions as well.

```
1 doc("countries.xml")/countries/country[@name = "Mongolia"]/data(@area)
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

```
1565000
```

Expected Query Result:

```
1565000
```

Submit

Q2

1.0/1.0 point (graded)
Return the names of all cities that have the same name as the country in which they are located.

Note: Your solution will need to reference *doc("countries.xml")* to access the data.

```
1 for $ct in doc("countries.xml")/countries/country/city
2 where $ct/name = $ct/parent::*/@name
3 return $ct/name
```

```

```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

```
<name>Mexico</name>
<name>Singapore</name>
```

Expected Query Result:

```
<name>Mexico</name>
<name>Singapore</name>
```

Submit

Q3

1.0/1.0 point (graded)

Return the average population of Russian-speaking countries.

Note: Your solution will need to reference *doc("countries.xml")* to access the data.

```
avg(doc("countries.xml")/countries/country[language="Russian"]/@population)

```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

```
3.2017746666666668E7
```

Expected Query Result:

```
        3.2017746666666668E7
```

Submit

Q4

1.0/1.0 point (graded)

Return the names of all countries that have at least three cities with population greater than 3 million.

Note: Your solution will need to reference *doc("countries.xml")* to access the data.

```
doc("countries.xml")/countries/country[count(city[population>3000000]) > 2]/data(@name)
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

```
China India
```

Expected Query Result:

```
China India
```

Submit

Q5

1.0/1.0 point (graded)

Create a list of French-speaking and German-speaking countries. The result should take the form:

```
<result>
  <French>
    <country>country-name</country>
    <country>country-name</country>
    ...
  </French>
  <German>
    <country>country-name</country>
    <country>country-name</country>
    ...
  </German>
</result>
```

Note: Your solution will need to reference *doc("countries.xml")* to access the data.

```
<result>
  <French>
  { for $cf in doc("countries.xml")/countries/country[language="French"]
    return
          <country>{$cf/data(@name)}</country>
  }
  </French>
  <German>
  { for $cg in doc("countries.xml")/countries/country[language="German"]
    return
          <country>{$cg/data(@name)}</country>
  }
  </German>
</result>
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

```
<result>
  <French>
    <country>Belgium</country>
    <country>France</country>
    <country>French Guiana</country>
    <country>Guadeloupe</country>
    <country>Guinea</country>
    <country>Haiti</country>
    <country>Saint Pierre and Miquelon</country>
    <country>Switzerland</country>
  </French>
  <German>
    <country>Austria</country>
    <country>Belgium</country>
    <country>Germany</country>
    <country>Namibia</country>
    <country>Switzerland</country>
  </German>
</result>
```

Expected Query Result:

```
<result>
  <French>
    <country>Belgium</country>
    <country>France</country>
    <country>French Guiana</country>
    <country>Guadeloupe</country>
    <country>Guinea</country>
    <country>Haiti</country>
    <country>Saint Pierre and Miquelon</country>
    <country>Switzerland</country>
  </French>
  <German>
    <country>Austria</country>
    <country>Belgium</country>
    <country>Germany</country>
    <country>Namibia</country>
    <country>Switzerland</country>
  </German>
</result>
```

Submit

Q6

1.0/1.0 point (graded)

Return the countries with the highest and lowest population densities. Note that because the "/" operator has its own meaning in XPath and XQuery, the division operator is infix "div". To compute population density use "(@population div @area)". You can assume density values are unique. The result should take the form:

```
<result>
  <highest density="value">country-name</highest>
  <lowest density="value">country-name</lowest>
</result>
```

Note: Your solution will need to reference *doc("countries.xml")* to access the data.
(This problem is quite challenging; congratulations if you get it right.)

```
<result>
 {
 let $dlist := doc("countries.xml")/countries/country/(@population div @area)
 for $c in doc("countries.xml")/countries/country
 where $c/(@population div @area) = max($dlist)
 return <highest density ='{$c/(@population div @area)}'> {$c/data(@name)} </highest>
 }
 {
 let $dlist := doc("countries.xml")/countries/country/(@population div @area)
 for $c in doc("countries.xml")/countries/country
 where $c/(@population div @area) = min($dlist)
 return <lowest density ='{$c/(@population div @area)}'> {$c/data(@name)} </lowest>
 }
</result>
```

Press ESC then TAB or click outside of the code editor to exit

Correct

Correct

Your Query Result:

```
<result>
  <highest density='31052.3125'>Macau</highest>
  <lowest density='0.026752619966905682'>Greenland</lowest>
</result>
```

Expected Query Result:

```
<result>
  <highest density='31052.3125'>Macau</highest>
  <lowest density='0.026752619966905682'>Greenland</lowest>
</result>
```

Submit