Tutorial: XQuery and XSLT

Your company, Apasaja Private Limited, is commissioned by an online music database company to design and prototype XML version of a database application for the company's service. You are provided with a sample XML document.
Please note that minor changes have been made to the file from last week's tutorial. Download the following file from "`Luminus Files > Cases > Music Library`".

`library.xml`.

The following is an excerpt of the XML document "`library.xml`".

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <album>
        <title>Bua Hati</title>
        <artists>
            <artist>
                <name>Anang Ashanty</name>
                <country>Indonesia</country>
            </artist>
            <artist>
                <name>Kris Dayanti</name>
                <country>Indonesia</country>
            </artist>
        </artists>
        <songs>
            <song>
                <title>Timang-Timang</title>
                <duration>5:13</duration>
            </song>
            <song>
                <title>Miliki Diriku</title>
                <duration>5:35</duration>
            </song>
            <song>
                <title>Bua Hati</title>
                <duration>5:07</duration>
            </song>
        </songs>
        <genres>
            <genre>Pop</genre>
            <genre>World</genre>
        </genres>
        <year>1998</year>
    </album>
    ...
</library>
```

Your answers to the questions should be general enough to work for other similar documents (namely other documents following the same design logic).

XQuery

1. Find the albums where the number of songs is greater than or equal to 3. Return the album title and count sorted by the count with the highest count appearing first. Your query should be written in the form of a FLWOR expression.

   For reference, your XML output should look somewhat similar to this:

```
<albums>
    <album>
        <title>No War</title>
        <count>4</count>
    </album>
    <album>
        <title>Bua Hati</title>
        <count>3</count>
    </album>
    <album>
        <title>Separuh Jiwaku Pergi</title>
        <count>3</count>
    </album>
</albums>
```

Solution:

```
<albums>{
    for $album in doc("library.xml")/child::library/child::album
        let $title := $album/child::title/text()
        let $count := count($album/child::songs/child::song)
        where $count ge 3
        order by $count descending
        return
            <album>
                <title>{$title}</title>
                <count>{$count}</count>
            </album>
}</albums>
```

2. Show the title and duration of the songs made by Indonesian artists sorted by duration in descending order.

   You may find declaring a local helper function as well as the built-in `tokenize($input, $pattern)` function useful. For reference, your output should look something like this:

```
<songs>
    <song>
        <title>Miliki Diriku</title>
        <duration>10:35</duration>
    </song>
    <song>
        <title>Belajarlah Untuk Cinta</title>
        <duration>10:23</duration>
    </song>
    <song>
        <title>Bua Hati</title>
        <duration>5:35</duration>
    </song>
    <song>
        <title>Timang-Timang</title>
        <duration>5:13</duration>
    </song>
    <song>
        <title>Separuh Jiwaku Pergi</title>
        <duration>5:00</duration>
    </song>
    <song>
        <title>Hujanpun Menangis</title>
        <duration>4:17</duration>
```

```
    </song>
</songs>
```

Solution: Start by declaring a function to get the duration as the number of seconds with type integer:

```
declare function local:duration-in-sec($duration-str as xs:string) as xs:int {
    let $min-secs := tokenize($duration-str, ':')
    return 60 * xs:int($min-secs[1]) + xs:int($min-secs[2])
};
```

Solution 1 (without let-binding):

```
<songs>{
    for $song in doc("library.xml")/child::library/child::album[child::artists/child::artist/
                                    child::country = 'Indonesia']/child::songs/child::song
        order by local:duration-in-sec($song/child::duration) descending
        return $song
}</songs>
```

Solution 2 (with let-binding):

```
<songs>{
    for $song in doc("library.xml")/child::library/child::album[child::artists/child::artist/
                                    child::country = 'Indonesia']/child::songs/child::song
        let $duration := local:duration-in-sec($song/child::duration)
        order by $duration descending
        return $song
}</songs>
```

3. Show the list of titles of the albums made by Indonesian artists sorted by title. For each album show the songs sorted by duration in descending order.

   For reference, your HTML output when rendered should look somewhat similar to this:

   - **Bua Hati**
     - 10:35 - Miliki Diriku
     - 5:35 - Bua Hati
     - 5:13 - Timang-Timang
   - **Separuh Jiwaku Pergi**
     - 10:23 - Belajarlah Untuk Cinta
     - 5:00 - Separuh Jiwaku Pergi
     - 4:17 - Hujanpun Menangis

Solution:

```
declare function local:duration-in-sec($duration-str as xs:string) as xs:int {
    let $min-secs := tokenize($duration-str, ':')
    return 60 * xs:int($min-secs[1]) + xs:int($min-secs[2])
};

<html>
    <body>
        <ul>{
            for $album in doc("library.xml")/child::library/child::album[child::artists/child::artist/
                                                child::country = 'Indonesia']
            let $album-title := $album/child::title/text()
            order by $album-title
            return
                <li>
                    <h2>{$album-title}</h2>
                    <ul>{
                        for $song in $album/child::songs/child::song
                        order by local:duration-in-sec($song/child::duration) descending
                        return
                            <li>{$song/child::duration/text()} - {$song/child::title/text()}</li>
                    }</ul>
                </li>
        }</ul>
    </body>
</html>
```

4. Color the songs with a duration greater than or equal to 600 seconds green and otherwise red.

   Note that "||" is used in XQuery for string concatenation. For reference, your HTML output when rendered should look somewhat similar to this:

   - **Bua Hati**
     - 10:35 - Miliki Diriku
     - 5:35 - Bua Hati
     - 5:13 - Timang-Timang
   - **Separuh Jiwaku Pergi**
     - 10:23 - Belajarlah Untuk Cinta
     - 5:00 - Separuh Jiwaku Pergi
     - 4:17 - Hujanpun Menangis

Solution:

```
declare function local:duration-in-sec($duration-str as xs:string) as xs:int {
    let $min-secs := tokenize($duration-str, ':')
    return 60 * xs:int($min-secs[1]) + xs:int($min-secs[2])
};

<html>
    <body>
        <ul>{
            for $album in doc("library.xml")/child::library/child::album[child::artists/child::artist/
                                                child::country = 'Indonesia']
```

```
            let $album-title := $album/child::title/text()
            order by $album-title
            return
                <li>
                    <h2>{$album-title}</h2>
                    <ul>{
                        for $song in $album/child::songs/child::song
                        let $duration := local:duration-in-sec($song/child::duration)
                        order by $duration descending
                        return
                            element li {
                                attribute style {if ($duration >= 600) then 'color:green;' else 'color:red;'},
                                $song/child::duration/text() || ' - ' || $song/child::title/text()
                            }
                    }</ul>
                </li>
        }</ul>
    </body>
</html>
```

5. Find the albums where the total listening duration (the sum of all the album's songs' durations) is greater than 1100 seconds. The albums should be sorted by title. In the XML output, you should return the album elements that satisfy the constraint with the title and an extra attribute `duration`. Your main code should be formed as a FLWOR expression. You may use functions you have declared in previous questions.

   For reference, your XML output should look somewhat similar to this:

```
<albums>
    <album>
        <title>Bua Hati</title>
        <duration>1283</duration>
    </album>
    <album>
        <title>Separuh Jiwaku Pergi</title>
        <duration>1180</duration>
    </album>
</albums>
```

Solution:

```
declare function local:duration-in-sec($duration-str as xs:string) as xs:int {
    let $min-secs := tokenize($duration-str, ':')
    return 60 * xs:int($min-secs[1]) + xs:int($min-secs[2])
};

<albums>{
    for $album in doc("library.xml")/child::library/child::album
        let $title := $album/child::title/text()
        let $durations := $album/child::songs/child::song/child::duration
        let $total-duration := sum(for $duration in $durations return local:duration-in-sec($duration))
        where $total-duration gt 1100
        order by $title
        return
            <album>
                <title>{$title}</title>
                <duration>{$total-duration}</duration>
```

```
            </album>
}</albums>
```

6. Explain a key difference between the for loop used in XQuery compared to an imperative language such as Python.

Solution: XQuery is a functional language. Each iteration is executed in parallel and no communication between the threads are allowed. This is in contrast to imperative languages where the iterations are executed sequentially and may i.e. increment a counter or likewise. See XQuery/FLWOR Expression for more info.

XSLT

1. Write an XSLT stylesheet that displays in a table or a list of an HTML document the name and released year of the albums with genre "`Pop`" of the XML document.

   Prefer XSLT template matching with "`value-of`" and "`apply-template`" to imperative control structures (e.g. "`for-each`").

Solution:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">
    <html>
        <body>
            <ul>
                <xsl:apply-templates select="child::library/child::album[child::genres/child::genre='Pop']"/>
            </ul>
        </body>
    </html>
</xsl:template>

<xsl:template match="album">
    <li>
        Album <xsl:value-of select="title"/> (<xsl:value-of select="year"/>)
    </li>
</xsl:template>

</xsl:stylesheet>
```

2. Perform XLST transformation through eXist-db using the XLST file you just created.

Solution:

```
transform:transform(doc("library.xml"), doc("library.xsl"), ())
```