Лабораториска вежба - 223260

ЗАДАЧА 1

Предложете решение како би ги споиле дадените датотеки во една датотека System.xml, така што би овозможиле брз и едноставен пристап до информациите за Клиентите и Артистите.

```
<?xml version="1.0" encoding="UTF-8"?>
<SYSTEM>
  <ARTISTS>
    <ARTIST ID="1" type="band">
      <NAME>Metallica</NAME>
      <COUNTRY>USA</COUNTRY>
      <GENRE>Metal</GENRE>
    </ARTIST>
    <ARTIST ID="2" type="band">
      <NAME>Disturbed</NAME>
      <COUNTRY>USA</COUNTRY>
      <GENRE>Metal</GENRE>
    </ARTIST>
    <ARTIST ID="3" type="pop_group">
      <NAME>The Backstreet Boys</NAME>
      <COUNTRY>USA</COUNTRY>
      <GENRE>Pop</GENRE>
    </ARTIST>
    <ARTIST ID="16" type="dj">
      <YEAR_STARTED_PERFORMING>2009</YEAR_STARTED_PERFORMING>
      <NET_WORTH>25,000,000</NET_WORTH>
      <YEAR_OF_BIRTH>1991</YEAR_OF_BIRTH>
    </ARTIST>
    <ARTIST ID="17" type="dj">
      <YEAR_STARTED_PERFORMING>1980</YEAR_STARTED_PERFORMING>
      <NET_WORTH>75,000,000</NET_WORTH>
      <YEAR_OF_BIRTH>1967</YEAR_OF_BIRTH>
    </ARTIST>
    <ARTIST ID="18" type="dj">
      <YEAR_STARTED_PERFORMING>2013</YEAR_STARTED_PERFORMING>
      <YEAR_OF_BIRTH>1987</YEAR_OF_BIRTH>
    </ARTIST>
    <ARTIST ID="13" type="singer">
      <YEAR_STARTED_PERFORMING>1988</YEAR_STARTED_PERFORMING>
      <YEAR_OF_BIRTH>1972</YEAR_OF_BIRTH>
    </ARTIST>
    <ARTIST ID="14" type="singer">
      <YEAR_STARTED_PERFORMING>1987</YEAR_STARTED_PERFORMING>
      <YEAR_OF_BIRTH>1971</YEAR_OF_BIRTH>
    </ARTIST>
    <ARTIST ID="15" type="singer">
      <YEAR_STARTED_PERFORMING>1988</YEAR_STARTED_PERFORMING>
    </ARTIST>
  </ARTISTS>
  <GROUPS>
```

```
    <GROUP ID="1">
      <YEAR_FORMED>1981</YEAR_FORMED>
      <NUMBER_OF_MEMBERS>9</NUMBER_OF_MEMBERS>
    </GROUP>
    <GROUP ID="2">
      <YEAR_FORMED>1994</YEAR_FORMED>
      <NUMBER_OF_MEMBERS>6</NUMBER_OF_MEMBERS>
    </GROUP>
    <GROUP ID="3">
      <YEAR_FORMED>1993</YEAR_FORMED>
      <NUMBER_OF_MEMBERS>10</NUMBER_OF_MEMBERS>
    </GROUP>
  </GROUPS>
  <ALBUMS>
    <ALBUM ID="1" ARTIST_ID="1">
      <NAME>Master Of Pupets</NAME>
      <RELEASE_YEAR>1986</RELEASE_YEAR>
      <PRICE>$10.00</PRICE>
    </ALBUM>
    <ALBUM ID="2" ARTIST_ID="1">
      <NAME>Metallica</NAME>
      <RELEASE_YEAR>1991</RELEASE_YEAR>
      <PRICE>$12.99</PRICE>
    </ALBUM>
    <ALBUM ID="3" ARTIST_ID="2">
      <NAME>Believe</NAME>
      <RELEASE_YEAR>2002</RELEASE_YEAR>
      <PRICE>$8.00</PRICE>
    </ALBUM>
    <ALBUM ID="4" ARTIST_ID="2">
      <NAME>Evolution</NAME>
      <RELEASE_YEAR>2018</RELEASE_YEAR>
      <PRICE>$15.00</PRICE>
    </ALBUM>
    <ALBUM ID="5" ARTIST_ID="2">
      <NAME>Immortalized</NAME>
      <RELEASE_YEAR>2015</RELEASE_YEAR>
      <PRICE>$16.00</PRICE>
    </ALBUM>
    <ALBUM ID="6" ARTIST_ID="2">
      <NAME>Indestructible</NAME>
      <RELEASE_YEAR>2008</RELEASE_YEAR>
      <PRICE>$14.00</PRICE>
    </ALBUM>
  </ALBUMS>
  <CATALOG>
    <CD ID="1" ALBUM_ID="1">
      <STATE>functional</STATE>
      <OCCUPIED>0</OCCUPIED>
    </CD>
    <CD ID="2" ALBUM_ID="1">
      <STATE>0</STATE>
      <OCCUPIED>1</OCCUPIED>
```

```
    </CD>
    <CD ID="3" ALBUM_ID="1">
      <STATE>2</STATE>
      <OCCUPIED>free</OCCUPIED>
    </CD>
    <CD ID="4" ALBUM_ID="2">
      <STATE>0</STATE>
      <OCCUPIED>0</OCCUPIED>
    </CD>
    <CD ID="5" ALBUM_ID="2">
      <STATE>0</STATE>
      <OCCUPIED>0</OCCUPIED>
    </CD>
  </CATALOG>
  <CLIENTS>
    <CLIENT ID="1">
      <NAME>Stephen</NAME>
      <SURNAME>Sims</SURNAME>
      <ADDRESS>Hollywood St.75</ADDRESS>
      <EMAIL>stephen.sims@example.com</EMAIL>
      <PHONE_NUMBER>754-1234</PHONE_NUMBER>
    </CLIENT>
    <CLIENT ID="2">
      <BASIC_INFO>
        <NAME>Barney</NAME>
        <SURNAME>Stinson</SURNAME>
        <ADDRESS>Some St.75</ADDRESS>
      </BASIC_INFO>
      <EMAIL>barney.stinson@example.com</EMAIL>
      <PHONE_NUMBER>755-1111</PHONE_NUMBER>
    </CLIENT>
    <CLIENT ID="3">
      <BASIC_INFO>
        <NAME>Walter</NAME>
        <SURNAME>White</SURNAME>
        <ADDRESS>Other St.55</ADDRESS>
      </BASIC_INFO>
      <EMAIL>walter.white@example.com</EMAIL>
    </CLIENT>
    <CLIENT ID="4">
      <NAME>Ned</NAME>
      <SURNAME>Stark</SURNAME>
      <ADDRESS>Second St.3</ADDRESS>
      <EMAIL>ned.stark@starks.com</EMAIL>
    </CLIENT>
    <CLIENT ID="5">
      <NAME>John</NAME>
      <SURNAME>Snow</SURNAME>
      <ADDRESS>Third St.2</ADDRESS>
      <EMAIL>john.snow@almoststark.com</EMAIL>
      <PHONE_NUMBER>754-1112</PHONE_NUMBER>
    </CLIENT>
  </CLIENTS>
```

```
  <RENTS>
    <RENT ID="1" CLIENT_ID="24" CD_ID="1">
      <FROM_DATE>2020-10-01</FROM_DATE>
      <RETURN_STATE>functional</RETURN_STATE>
      <RETURN_DATE>2020-10-20</RETURN_DATE>
    </RENT>
    <RENT ID="2" CLIENT_ID="1" CD_ID="2">
      <FROM_DATE>2020-01-10</FROM_DATE>
      <RETURN_STATE>0</RETURN_STATE>
      <RETURN_DATE>2020-01-21</RETURN_DATE>
    </RENT>
    <RENT ID="3" CLIENT_ID="16" CD_ID="3">
      <FROM_DATE>2020-01-10</FROM_DATE>
      <RETURN_STATE>0</RETURN_STATE>
      <RETURN_DATE>2020-01-25</RETURN_DATE>
    </RENT>
  </RENTS>
</SYSTEM>
```

A) Напишете ја XML Schema датотеката за System.xml, и додадете ги следните ограничувања:

- атрибутот occupied во ентитетот CD и атрибутот return state во релацијата Rent може да бидат 0, 1 и 2 или functional, slightly damaged и damaged, соодветно.
- проверка на валидноста на форматот на email атрибутот кај Client ентитетот.
- телефонските броеви треба да се во формат 75x-yyyy каде x е цифра од 0 до 9 а y е цифра од 1 до 9
- адресата може и да не постои, но ако постои може да биде составена или од два поделементи (улица и број) или пак да биде во формат улица (составена само од мали и големи букви) по што следува St. па една или две цифри за бројот.
- сите други ограничувања кои би ги извеле од податоците дадени во XML документите.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="SYSTEM">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ARTISTS">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="ARTIST" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:choice minOccurs="0" maxOccurs="1">
                      <xs:sequence>
                        <xs:element name="NAME" type="xs:string"/>
                        <xs:element name="COUNTRY" type="xs:string"/>
                        <xs:element name="GENRE" type="xs:string"/>
                      </xs:sequence>
                    </xs:choice>
                    <xs:element name="YEAR_STARTED_PERFORMING" type="xs:gYear"
minOccurs="0"/>
                    <xs:element name="NET_WORTH" type="xs:string" minOccurs="0"/>
                    <xs:element name="YEAR_OF_BIRTH" type="xs:gYear" minOccurs="0"/>
                  </xs:sequence>
```

```
            <xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
            <xs:attribute name="type" use="required">
              <xs:simpleType>
                <xs:restriction base="xs:string">
                  <xs:enumeration value="band"/>
                  <xs:enumeration value="pop_group"/>
                  <xs:enumeration value="dj"/>
                  <xs:enumeration value="singer"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="GROUPS">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="GROUP" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="YEAR_FORMED" type="xs:gYear"/>
              <xs:element name="NUMBER_OF_MEMBERS" type="xs:positiveInteger"/>
            </xs:sequence>
            <xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="ALBUMS">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="ALBUM" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="NAME" type="xs:string"/>
              <xs:element name="RELEASE_YEAR" type="xs:gYear"/>
              <xs:element name="PRICE" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
            <xs:attribute name="ARTIST_ID" type="xs:positiveInteger" use="required"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:element name="CATALOG">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="CD" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
```

```
                    <xs:element name="STATE" type="CDStateType"/>
                    <xs:element name="OCCUPIED" type="CDStateType"/>
                  </xs:sequence>
                  <xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
                  <xs:attribute name="ALBUM_ID" type="xs:positiveInteger" use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>

        <xs:element name="CLIENTS">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="CLIENT" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:choice>
                    <xs:sequence>
                      <xs:element name="NAME" type="xs:string"/>
                      <xs:element name="SURNAME" type="xs:string"/>
                      <xs:element name="ADDRESS" type="AddressType" minOccurs="0"/>
                    </xs:sequence>
                    <xs:element name="BASIC_INFO">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="NAME" type="xs:string"/>
                          <xs:element name="SURNAME" type="xs:string"/>
                          <xs:element name="ADDRESS" type="AddressType" minOccurs="0"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:choice>
                  <xs:sequence>
                    <xs:element name="EMAIL" type="EmailType"/>
                    <xs:element name="PHONE_NUMBER" type="PhoneNumberType"
minOccurs="0"/>
                  </xs:sequence>
                  <xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>

        <xs:element name="RENTS">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="RENT" maxOccurs="unbounded">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="FROM_DATE" type="xs:date"/>
                    <xs:element name="RETURN_STATE" type="CDStateType"/>
                    <xs:element name="RETURN_DATE" type="xs:date"/>
                  </xs:sequence>
                  <xs:attribute name="ID" type="xs:positiveInteger" use="required"/>
                  <xs:attribute name="CLIENT_ID" type="xs:positiveInteger" use="required"/>
                  <xs:attribute name="CD_ID" type="xs:positiveInteger" use="required"/>
```

```
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>

      <xs:simpleType name="CDStateType">
        <xs:restriction base="xs:string">
          <xs:enumeration value="0"/>
          <xs:enumeration value="1"/>
          <xs:enumeration value="2"/>
          <xs:enumeration value="functional"/>
          <xs:enumeration value="slightly damaged"/>
          <xs:enumeration value="damaged"/>
          <xs:enumeration value="free"/>
        </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="EmailType">
        <xs:restriction base="xs:string">
          <xs:pattern value="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"/>
        </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="PhoneNumberType">
        <xs:restriction base="xs:string">
          <xs:pattern value="75[0-9]-[1-9]{4}"/>
        </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="AddressType">
        <xs:union memberTypes="FullAddressType SplitAddressType"/>
      </xs:simpleType>

      <xs:simpleType name="FullAddressType">
        <xs:restriction base="xs:string">
          <xs:pattern value="[a-zA-Z ]+ St\.[0-9]{1,2}"/>
        </xs:restriction>
      </xs:simpleType>

      <xs:simpleType name="SplitAddressType">
        <xs:restriction base="xs:string">
          <xs:pattern value=".*"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:schema>
```

B) Напишете XQuery израз со кој користејќи ги оригиналните датотеки ќе ја изгенерирате датотеката System.xml. Забелешка: Резултатот од XQuery прашањето треба да ја претставува содржината на System.xml со точната структура што ја дефинирате во одговорот на барањето под а).

```
let $clients := doc("Clients.xml")//CLIENT
let $artists := doc("Artists.xml")//ARTIST
let $albums := doc("Albums.xml")//ALBUM
```

```
let $groups := doc("Groups.xml")//GROUP
let $rents := doc("Rent.xml")//RENT
let $catalog := doc("CatalogCD.xml")//CD

return
<SYSTEM>
  <CLIENTS>
    {
      for $client in $clients
      return
        <CLIENT ID="{$client/@ID}">
          <NAME>{$client/NAME/text()}</NAME>
          <SURNAME>{$client/SURNAME/text()}</SURNAME>
          <ADDRESS>
            {
              if ($client/ADDRESS/STREET and $client/ADDRESS/NUMBER) then (
              <STREET>{$client/ADDRESS/STREET/text()}</STREET>,
              <NUMBER>{$client/ADDRESS/NUMBER/text()}</NUMBER>
             )
              else
              <FULL_ADDRESS>{$client/ADDRESS/FULL_ADDRESS/text()}</FULL_ADDRESS>
            }
          </ADDRESS>
          <EMAIL>{$client/EMAIL/text()}</EMAIL>
          <PHONE_NUMBER>{$client/PHONE_NUMBER/text()}</PHONE_NUMBER>
        </CLIENT>
    }
  </CLIENTS>

  <ARTISTS>
    {
      for $artist in $artists
      return
        <ARTIST ID="{$artist/@ID}" type="{$artist/@type}">
          <NAME>{$artist/NAME/text()}</NAME>
          <COUNTRY>{$artist/COUNTRY/text()}</COUNTRY>
          <GENRE>{$artist/GENRE/text()}</GENRE>
          {
            if ($artist/@type = 'dj') then (

<YEAR_STARTED_PERFORMING>{$artist/YEAR_STARTED_PERFORMING/text()}</YEAR_STARTED_
PERFORMING>,
              <NET_WORTH>{$artist/NET_WORTH/text()}</NET_WORTH>,
              <YEAR_OF_BIRTH>{$artist/YEAR_OF_BIRTH/text()}</YEAR_OF_BIRTH>
             )
             else ()
          }
        </ARTIST>
    }
  </ARTISTS>

  <ALBUMS>
    {
```

```
        for $album in $albums
        return
          <ALBUM ID="{$album/@ID}" ARTIST_ID="{$album/@ARTIST_ID}">
            <NAME>{$album/NAME/text()}</NAME>
            <RELEASE_YEAR>{$album/RELEASE_YEAR/text()}</RELEASE_YEAR>
            <PRICE>{$album/PRICE/text()}</PRICE>
          </ALBUM>
    }
  </ALBUMS>

  <GROUPS>
    {
      for $group in $groups
      return
        <GROUP ID="{$group/@ID}">
          <YEAR_FORMED>{$group/YEAR_FORMED/text()}</YEAR_FORMED>

<NUMBER_OF_MEMBERS>{$group/NUMBER_OF_MEMBERS/text()}</NUMBER_OF_MEMBERS>
        </GROUP>
    }
  </GROUPS>

  <RENTS>
    {
      for $rent in $rents
      return
        <RENT ID="{$rent/@ID}" CLIENT_ID="{$rent/@CLIENT_ID}" CD_ID="{$rent/@CD_ID}">
          <FROM_DATE>{$rent/FROM_DATE/text()}</FROM_DATE>
          <RETURN_STATE>{$rent/RETURN_STATE/text()}</RETURN_STATE>
          <RETURN_DATE>{$rent/RETURN_DATE/text()}</RETURN_DATE>
        </RENT>
    }
  </RENTS>
   <CATALOG>
  {
   for $cd in $catalog
   return
    <CD ID="{$cd/@ID}" ALBUM_ID="{$cd/@ALBUM_ID}">
     <STATE>{$cd/STATE/text()}</STATE>
     <OCCUPIED>{$cd/OCCUPIED/text()}</OCCUPIED>
    </CD>
  }
 </CATALOG>
</SYSTEM>
```

C) Кои се проблемите кои ги има претходното решение, доколку ги има? Што би смениле за да овозможите брз пристап до сите албуми кои ги има во системот, но и сите албуми кои ги има даден артист? Образложете зошто сте го избрале ова решение!

Претходното XQuery решение **технички е точно** – ги зема податоците од оригиналните XML-документи и ја генерира посакуваната структура System.xml. Сепак, од аспект на **организација на податоците, ефикасност и брз пристап**, има неколку **важни слабости**.

1. Раздвоени податоци (структури кои не се вгнездени)

- Албумите (<ALBUM>) се ставени во посебен <ALBUMS> елемент, додека артистите (<ARTIST>) се во друг.
- За да ги најдеш **сите албуми на еден артист**, мора да ги филтрираш сите албуми според @ARTIST_ID.
- Ова води до **неоптимални пребарувања**, особено ако се користи оваа XML структура за преглед, анализа или веб-апликација.

2. Нема би-дирекционална навигација
   - Не можеш од <ALBUM> лесно да „скокнеш“ до податоци за артистот (освен преку ID), ниту пак обратно.
   - Секој ID мора експлицитно да се обработува преку where или let, што значи **потребни се дополнителни XQuery изрази** секогаш кога се прави поврзување.

Решение:

Секој <ARTIST> да ги вклучува своите албуми како вложен <ALBUMS> елемент.

```
<ARTIST ID="a1" type="solo">
  <NAME>John Smith</NAME>
  <COUNTRY>USA</COUNTRY>
  <GENRE>Rock</GENRE>
  <ALBUMS>
    <ALBUM ID="alb1">
      <NAME>First Album</NAME>
      <RELEASE_YEAR>2018</RELEASE_YEAR>
      <PRICE>12.99</PRICE>
    </ALBUM>
    <ALBUM ID="alb2">
      ...
    </ALBUM>
  </ALBUMS>
</ARTIST>
```

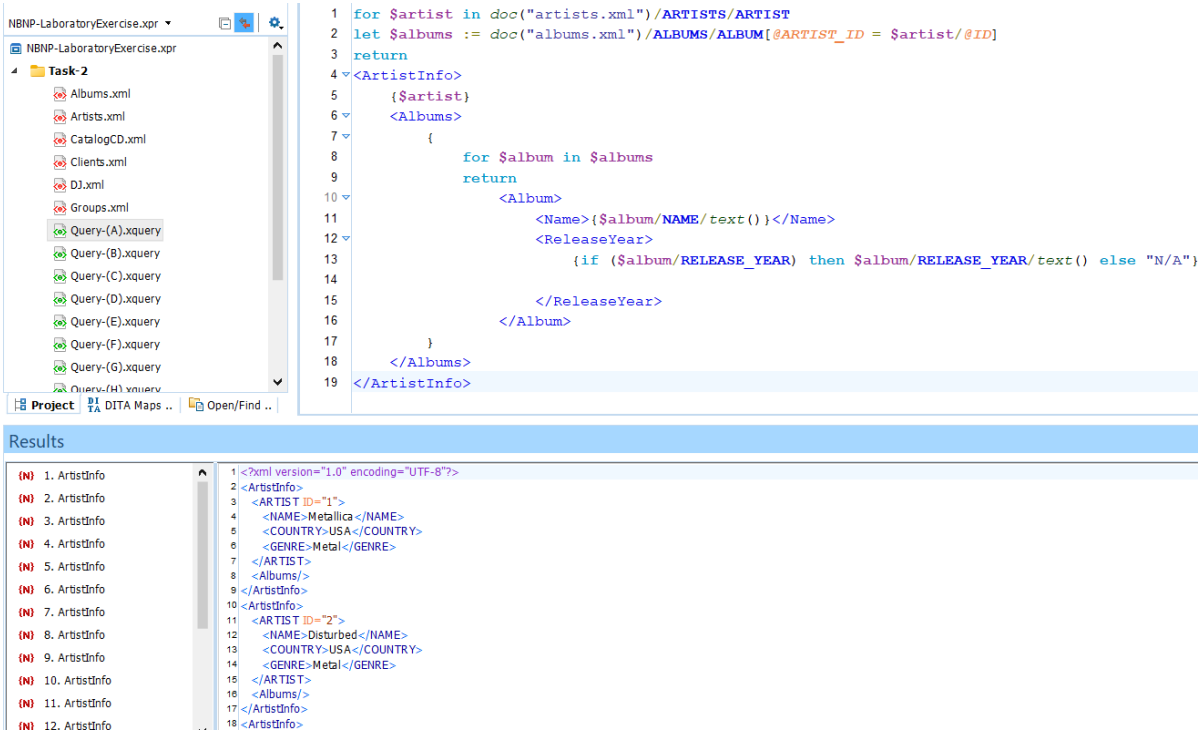
ЗАДАЧА 2

Да се напишат XQuery изразите со кои ќе се одговори на следните прашања:

а. Прикажи ги сите артисти (сите информации) заедно со насловите и датумите на издавање на сите албуми кои ги имаат снимено.

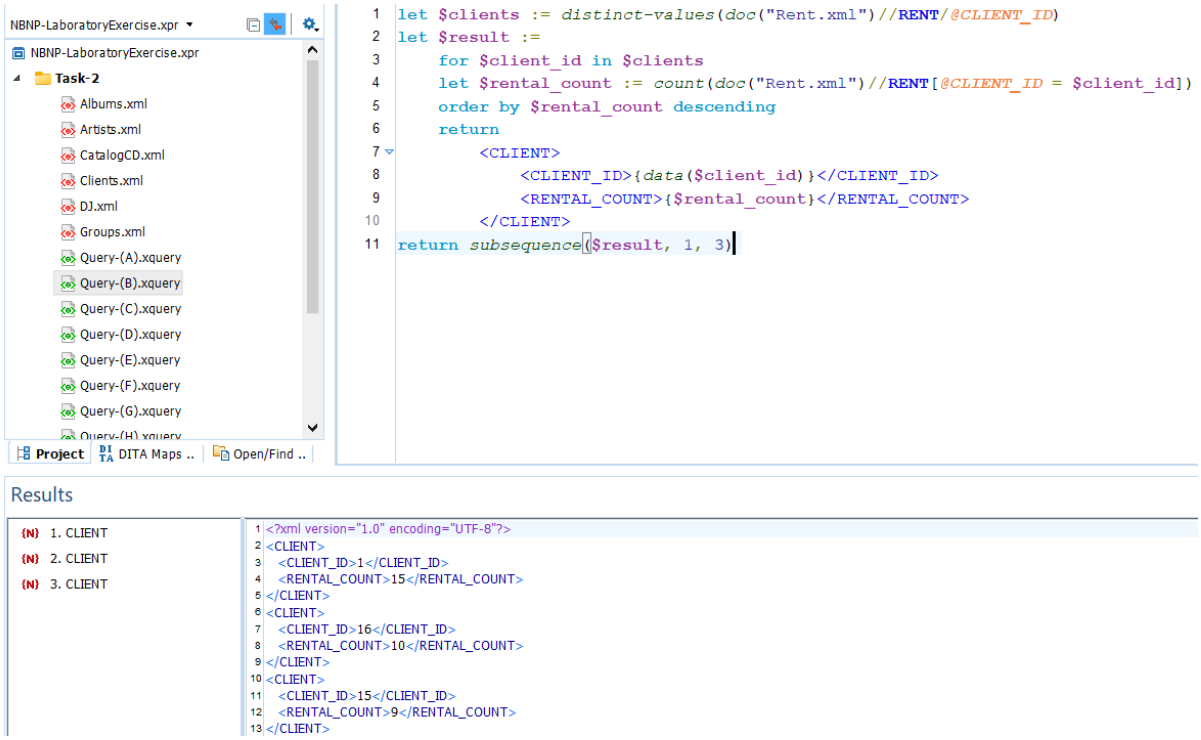
```
for $artist in doc("Artist.xml")/ARTISTS/ARTIST
let $albums := doc("Albums.xml")/ALBUMS/ALBUM[@ARTIST_ID = $artist/@ID]
return
<ArtistInfo>
  {$artist}
  <Albums>
```

```
    {
      for $album in $albums
      return
        <Album>
          <Name>{$album/NAME/text()}</Name>
          <ReleaseYear>
            {if ($album/RELEASE_YEAR) then $album/RELEASE_YEAR/text() else "N/A"}

          </ReleaseYear>
        </Album>
    }
  </Albums>
</ArtistInfo>
```



b. Најди ги првите три члена со најголем број изнајмувања.

```
let $clients := distinct-values(doc("Rent.xml")//RENT/@CLIENT_ID)
let $result :=
  for $client_id in $clients
  let $rental_count := count(doc("Rent.xml")//RENT[@CLIENT_ID = $client_id])
  order by $rental_count descending
  return
    <CLIENT>
      <CLIENT_ID>{data($client_id)}</CLIENT_ID>
      <RENTAL_COUNT>{$rental_count}</RENTAL_COUNT>
    </CLIENT>
return subsequence($result, 1, 3)
```



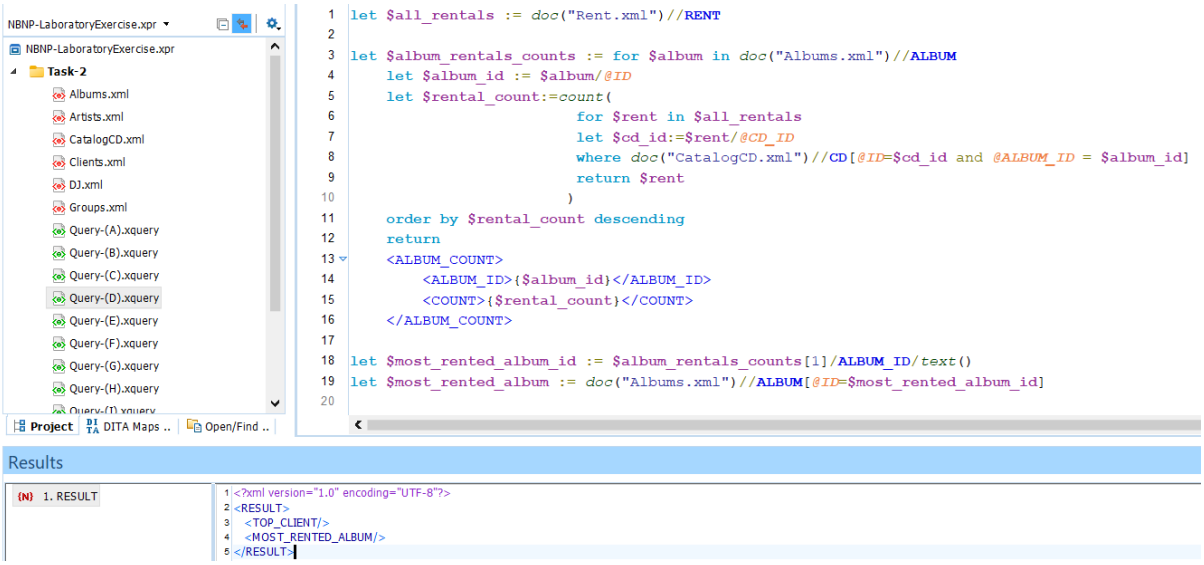
c. Врати ги сите албуми кои биле изнајмени најмалку три пати во периодот од јануари до март 2020.

```
let $janToMarchRentals := doc("Rent.xml")//RENT[
  let $date_parts := tokenize(FROM_DATE, '/')
  let $month := number($date_parts[1])
  let $year := number($date_parts[3])
  return $year=2020 and $month>=1 and $month<=3
]

let $rentedAlbums := for $album in doc("Albums.xml")//ALBUM
let $album_id := $album/@ID
let $rental_count := count(
                for $rent in $janToMarchRentals
                let $cd_id := $rent/@CD_ID
                where doc("CatalogCD.xml")//CD[@ID=$cd_id and @ALBUM_ID = $album_id]
                return $rent
            )
  where $rental_count>=3
  return
    <ALBUM>
    <ID>{$album_id}</ID>
    <NAME>{$album/NAME/text()}</NAME>
    <RENTAL_COUNT>{$rental_count}</RENTAL_COUNT>
    </ALBUM>
return $rentedAlbums
```

d. Прикажи го клиентот кој најмногу изнајмувал од најизнајмениот албум заедно со информациите за албумот

```
let $all_rentals := doc("Rent.xml")//RENT
```

```
let $album_rentals_counts := for $album in doc("Albums.xml")//ALBUM
  let $album_id := $album/@ID
  let $rental_count:=count(
              for $rent in $all_rentals
              let $cd_id:=$rent/@CD_ID
              where doc("CatalogCD.xml")//CD[@ID=$cd_id and @ALBUM_ID = $album_id]
              return $rent
             )
  order by $rental_count descending
  return
  <ALBUM_COUNT>
    <ALBUM_ID>{$album_id}</ALBUM_ID>
    <COUNT>{$rental_count}</COUNT>
  </ALBUM_COUNT>

let $most_rented_album_id := $album_rentals_counts[1]/ALBUM_ID/text()
let $most_rented_album := doc("Albums.xml")//ALBUM[@ID=$most_rented_album_id]

let $album_cd_id := for $cd in doc("CatalogCD.xml")//CD[@ALBUM_ID = $most_rented_album_id]
  return $cd/@ID

let $album_rentals := for $rent in $all_rentals
  where $rent/@CD_ID = $album_cd_id
  return $rent

let $client_rental_counts :=
  for $client_id in distinct-values($album_rentals/@CLIENT_ID)
  let $count := count($album_rentals[@CLIENT_ID = $client_id])
  order by $count descending
  return
    <CLIENT_COUNT>
      <CLIENT_ID>{$client_id}</CLIENT_ID>
      <COUNT>{$count}</COUNT>
    </CLIENT_COUNT>

let $top_client_id := $client_rental_counts[1]/CLIENT_ID/text()
let $top_client := doc("Clients.xml")//CLIENT[@ID=top_client_id]

return
  <RESULT>
    <TOP_CLIENT>
      {$top_client}
    </TOP_CLIENT>
    <MOST_RENTED_ALBUM>
      {$most_rented_album}
    </MOST_RENTED_ALBUM>
  </RESULT>
```



e. Прикажи го просечниот број на изнајмувања на секој албум посебно.

```
let $albums := doc("Albums.xml")//ALBUM
let $catalog := doc("CatalogCD.xml")//CD
let $rents := doc("Rent.xml")//RENT

return
 for $album in $albums
 let $album_id := $album/@ID
 let $cds := $catalog[@ALBUM_ID = $album_id]
 let $cd_ids := $cds/@ID
 let $rent_count := count(
  for $r in $rents
  where some $id in $cd_ids satisfies $r/@CD_ID = $id
  return $r
 )
 let $cd_count := count($cd_ids)
 let $average :=
  if ($cd_count > 0) then
   round($rent_count div $cd_count * 100) div 100
  else
   0
 return
  <AlbumAverage>
   <AlbumID>{ $album_id }</AlbumID>
   <Name>{ $album/NAME/text() }</Name>
   <CD_Count>{ $cd_count }</CD_Count>
   <TotalRentals>{ $rent_count }</TotalRentals>
   <AverageRentalsPerCD>{ $average }</AverageRentalsPerCD>
  </AlbumAverage>
```

f. Врати го вкупниот профит кој е направен за секој албум посебно.

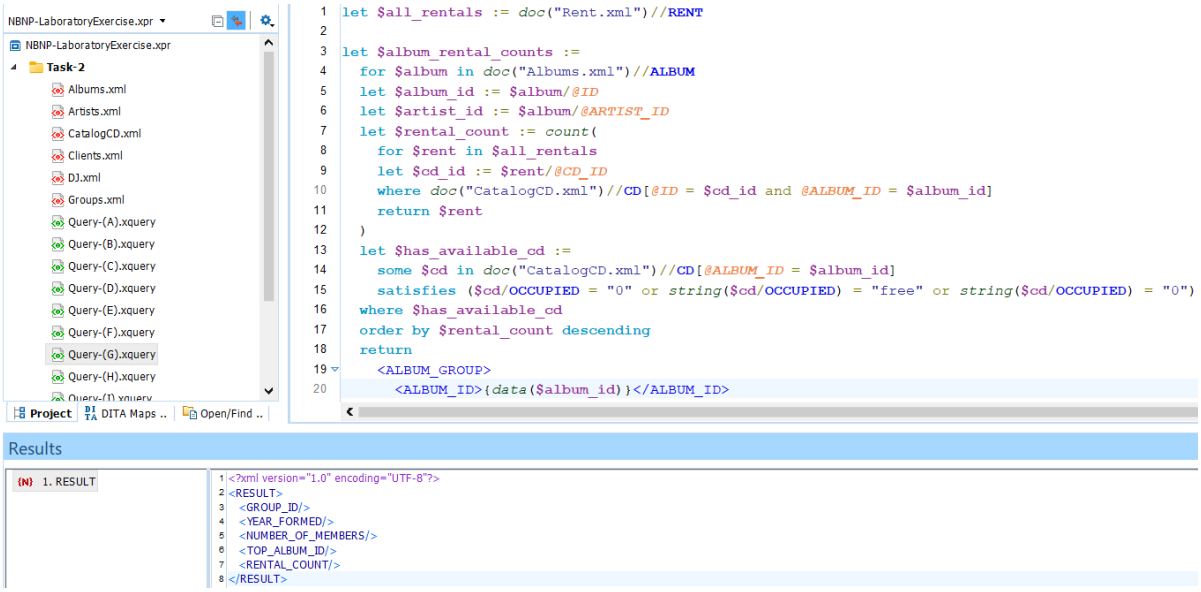
```
let $albums := doc("Albums.xml")//ALBUM
let $catalog := doc("CatalogCD.xml")//CD
```

```
let $rents := doc("Rent.xml")//RENT

return
  for $album in $albums
  let $album_id := $album/@ID
  let $album_price := xs:decimal(replace($album/PRICE, '[$]', ''))
  let $cds := $catalog[@ALBUM_ID = $album_id]
  let $cd_ids := $cds/@ID
  let $rent_count := count(
                for $r in $rents
                where some $id in $cd_ids satisfies $r/@CD_ID=$id
                return $r
                )
  let $total_profit := $rent_count * $album_price
  return
    <AlbumProfit>
      <AlbumID>{ $album_id }</AlbumID>
      <Name>{ $album/NAME/text() }</Name>
      <Price>{ $album_price }</Price>
      <TotalRentals>{ $rent_count }</TotalRentals>
      <TotalProfit>{ format-number($total_profit, '#.00') }</TotalProfit>
    </AlbumProfit>
```

g. Најди ја групата чиј албум е најмногу пати изнајмен, но постои барем едно CD кое во моментот не е изнајмено.

```
let $all_rentals := doc("Rent.xml")//RENT

let $album_rental_counts :=
 for $album in doc("Albums.xml")//ALBUM
 let $album_id := $album/@ID
 let $artist_id := $album/@ARTIST_ID
 let $rental_count := count(
  for $rent in $all_rentals
  let $cd_id := $rent/@CD_ID
  where doc("CatalogCD.xml")//CD[@ID = $cd_id and @ALBUM_ID = $album_id]
  return $rent
 )
 let $has_available_cd :=
  some $cd in doc("CatalogCD.xml")//CD[@ALBUM_ID = $album_id]
  satisfies ($cd/OCCUPIED = "0" or string($cd/OCCUPIED) = "free" or string($cd/OCCUPIED) = "0")
 where $has_available_cd
 order by $rental_count descending
 return
  <ALBUM_GROUP>
   <ALBUM_ID>{data($album_id)}</ALBUM_ID>
   <ARTIST_ID>{data($artist_id)}</ARTIST_ID>
   <RENTAL_COUNT>{$rental_count}</RENTAL_COUNT>
  </ALBUM_GROUP>

let $top_album := $album_rental_counts[1]
let $group_id := $top_album/ARTIST_ID/text()
let $group := doc("Groups.xml")//GROUP[@ID = $group_id]
```

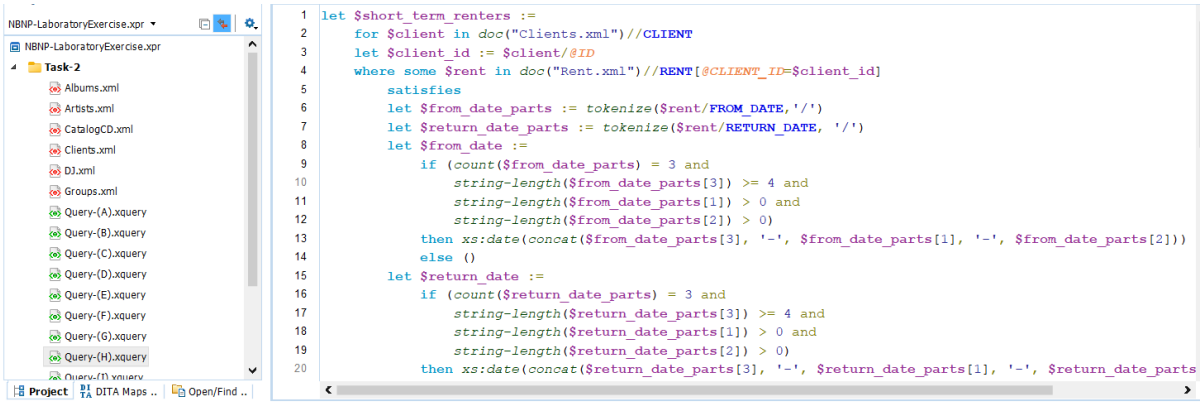
```
return
 <RESULT>
  <GROUP_ID>{data($group/@ID)}</GROUP_ID>
  <YEAR_FORMED>{$group/YEAR_FORMED/text()}</YEAR_FORMED>
  <NUMBER_OF_MEMBERS>{$group/NUMBER_OF_MEMBERS/text()}</NUMBER_OF_MEMBERS>
  <TOP_ALBUM_ID>{$top_album/ALBUM_ID/text()}</TOP_ALBUM_ID>
  <RENTAL_COUNT>{$top_album/RENTAL_COUNT/text()}</RENTAL_COUNT>
 </RESULT>
```



h. Прикажи ги сите клиенти кои барем еднаш изнајмиле CD во времетраење пократко од 10 дена.

```
let $short_term_renters :=
  for $client in doc("Clients.xml")//CLIENT
  let $client_id := $client/@ID
  where some $rent in doc("Rent.xml")//RENT[@CLIENT_ID=$client_id]
    satisfies
    let $from_date_parts := tokenize($rent/FROM_DATE,'/')
    let $return_date_parts := tokenize($rent/RETURN_DATE, '/')
    let $from_date :=
      if (count($from_date_parts) = 3 and
        string-length($from_date_parts[3]) >= 4 and
        string-length($from_date_parts[1]) > 0 and
        string-length($from_date_parts[2]) > 0)
      then xs:date(concat($from_date_parts[3], '-', $from_date_parts[1], '-', $from_date_parts[2]))
      else ()
    let $return_date :=
      if (count($return_date_parts) = 3 and
        string-length($return_date_parts[3]) >= 4 and
        string-length($return_date_parts[1]) > 0 and
        string-length($return_date_parts[2]) > 0)
      then xs:date(concat($return_date_parts[3], '-', $return_date_parts[1], '-', $return_date_parts[2]))
      else ()
    return
      if (exists($from_date) and exists($return_date)) then
```

```
        days-from-duration($return_date - $from_date) < 10
      else
        false()
  return $client

return
  <SHORT_TERM_RENTERS>
    {
    for $client in $short_term_renters
    return
      <CLIENT>
        <ID>{$client/@ID}</ID>
        {
          if ($client/NAME) then
            <NAME>{$client/NAME/text()}</NAME>
          else
            <NAME>{$client/BASIC_INFO/NAME/text()}</NAME>
        }
        {
          if($client/SURNAME) then
            <SURNAME>{$client/SURNAME/text()}</SURNAME>
          else
            <SURNAME>{$client/BASIC_INFO/SURNAME/text()}</SURNAME>
        }
        {
          if ($client/EMAIL) then
            <EMAIL>{$client/EMAIL/text()}</EMAIL>
          else ()
        }
      </CLIENT>
    }
  </SHORT_TERM_RENTERS>
```



i. Најди го омилениот артист на секој клиент посебно (сите информации за артистот од кој клиентот има изнајмено најмногу албуми, ако има повеќе вратете го првиот).

```
for $client in doc("Clients.xml")//CLIENT
let $rentals := doc("Rent.xml")//RENT[CLIENT_ID=$client/@ID]
let $artistRentals :=
  for $artist in doc("Artists.xml")//ARTIST
  let $artistAlbums := doc("Albums.xml")//ALBUM[@ARTIST_ID=$artist/ID]
  let $rentCount := count($rentals[CD_ID=$artistAlbums/@ID])
  where $rentCount > 0
  order by $rentCount descending
  return
    <ARTIST_RENTAL>
      <ARTIST_ID>{$artist/@ID}</ARTIST_ID>
      <ARTIST_NAME>{$artist/NAME/text()}</ARTIST_NAME>
      <RENT_COUNT>{$rentCount}</RENT_COUNT>
      <COUNTRY>{$artist/COUNTRY/text()}</COUNTRY>
      <GENRES>
        {for $genre in $artist/GENRE
         return <GENRE>{$genre/text()}</GENRE>}
      </GENRES>
    </ARTIST_RENTAL>

let $favoriteArtist := $artistRentals[1]
where count($artistRentals) > 0
return
  <CLIENT_FAVORITE>
    <CLIENT_ID>{$client/@ID}</CLIENT_ID>
    <CLIENT_NAME>
      {if ($client/NAME) then $client/NAME/text()
      else $client/BASIC_INFO/NAME/text()}
    </CLIENT_NAME>
    <CLIENT_SURNAME>
      {if ($client/SURNAME) then $client/SURNAME/text()
      else $client/BASIC_INFO/SURNAME/text()}
    </CLIENT_SURNAME>
    <FAVORITE_ARTIST>
      {$favoriteArtist}
    </FAVORITE_ARTIST>

  </CLIENT_FAVORITE>
```

j. Најди го најомилениот артист (артистот кој е омилен на најголемиот број клиенти).

😵

k. Напиши кориснички дефинирана функција за генерирање месечен извештај. Месецот и годината се задаваат како влезни параметри, а во извештајот за тековниот месец ќе се вратат профитот направен од сите албуми и бројот на продажби по албум за секој артист посебно.

😵