**HOMEWORK 4**

Due Date: April 25, 2019

Points: 50 (not 40).

*It might* be a good idea if you do question 4)a) first.

Then do question 1).

Then do question 2).

Then do question 4)b).

Then do question 3).

Then do question 5).

That seems to be the easiest order.

---

1) We are going back once again to the table BOROUGH_NEIGH from Question 1)e) in Homework 2.

In my solution that table started as follows:

<table>
<thead>
<tr>
<th>Borough</th>
<th>Neighborhood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bronx</td>
<td>Melrose</td>
</tr>
<tr>
<td>Bronx</td>
<td>Mott Haven</td>
</tr>
<tr>
<td>Bronx</td>
<td>Port Morris</td>
</tr>
<tr>
<td>Bronx</td>
<td>Hunts Point</td>
</tr>
<tr>
<td>Bronx</td>
<td>Longwood</td>
</tr>
<tr>
<td>Bronx</td>
<td>Claremont</td>
</tr>
<tr>
<td>Bronx</td>
<td>Concourse Village</td>
</tr>
</tbody>
</table>

...  

...  

Please make sure that you have this table and that

- Every row is unique

- It is sorted by the borough column in ascending alphabetical order

So, all rows with Bronx should come first. Then all rows with Brooklyn, etc. The neighborhood column does not have to be sorted and does not have to be unique either.
If your table does not have unique rows and is not sorted, then please create a new table BOROUGH_NEIGHBOR1 that is sorted with unique rows.

Now the purpose of this homework is to go “in the opposite direction.”

Now we are going from data in first normal form back to NFNF data.

First with XML. Then with JSON.

So the goal of this question is create and fill a new table BN_XML with one column called BNDATA of type XMLtype.

a) Create a new table BN_XML with one column called BNDATA of type XMLtype.

Show the create statement.

b) Write a PL/SQL program that translates data from BOROUGH_NEIGHBOR into XML and inserts it into the new table BN_XML.

Use the tag pairs <BN>, <BORO>, <NEIGH>

For the partial data EXAMPLE above it should look as follows:

<table>
<thead>
<tr>
<th>BN_XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>BNDATA</td>
</tr>
<tr>
<td>&lt;BN&gt;</td>
</tr>
<tr>
<td>&lt;BORO&gt;Bronx&lt;/BORO&gt;</td>
</tr>
<tr>
<td>&lt;NEIGH&gt;Melrose&lt;/NEIGH&gt;</td>
</tr>
<tr>
<td>&lt;/BN&gt;</td>
</tr>
<tr>
<td>&lt;BN&gt;</td>
</tr>
<tr>
<td>&lt;BORO&gt;Bronx&lt;/BORO&gt;</td>
</tr>
<tr>
<td>&lt;NEIGH&gt;Mott Haven&lt;/NEIGH&gt;</td>
</tr>
<tr>
<td>&lt;/BN&gt;</td>
</tr>
<tr>
<td>&lt;BN&gt;</td>
</tr>
<tr>
<td>&lt;BORO&gt;Bronx&lt;/BORO&gt;</td>
</tr>
<tr>
<td>&lt;NEIGH&gt;Port Morris&lt;/NEIGH&gt;</td>
</tr>
<tr>
<td>&lt;/BN&gt;</td>
</tr>
<tr>
<td>&lt;BN&gt;</td>
</tr>
<tr>
<td>&lt;BORO&gt;Bronx&lt;/BORO&gt;</td>
</tr>
<tr>
<td>&lt;NEIGH&gt;Hunts Point&lt;/NEIGH&gt;</td>
</tr>
<tr>
<td>&lt;/BN&gt;</td>
</tr>
<tr>
<td>&lt;BN&gt;</td>
</tr>
<tr>
<td>&lt;BORO&gt;Bronx&lt;/BORO&gt;</td>
</tr>
<tr>
<td>&lt;NEIGH&gt;Longwood&lt;/NEIGH&gt;</td>
</tr>
<tr>
<td>&lt;/BN&gt;</td>
</tr>
<tr>
<td>&lt;BN&gt;</td>
</tr>
<tr>
<td>&lt;BORO&gt;Bronx&lt;/BORO&gt;</td>
</tr>
<tr>
<td>&lt;NEIGH&gt;Claremont&lt;/NEIGH&gt;</td>
</tr>
<tr>
<td>&lt;/BN&gt;</td>
</tr>
</tbody>
</table>
HINTS: So of course you need a cursor. That cursor will give you the values (Bronx and Melrose, for example).

But then you have to “wrap” these values into the tag pairs using lots of || concatenation operators.
The result of this will be one long string from <BN> … to </BN> for each row of the new table, but of course for every row this string looks different.

And then you call the XML constructor with this long string and insert into the table for every row.

Students are usually struggling with this idea. Start simple. Create a test table. And then write a program that just inserts <Boro>Bronx</Boro> into the first couple of rows and <Boro>Brooklyn</Boro> into the next group of rows, but USING A CURSOR. Do not “hardcode” the words Bronx and Brooklyn into this program.

After that works, drop the table and start over with the more complicated row data.

Show the complete program.

[6]

c) Write a select statement that shows the complete content of the table BN_XML. All values have to be readable. No object identifiers should appear.

Show the select statement.
Show the result.

(Show either the complete result or the first 20 table rows and last 20 table rows of the result. Whatever you prefer. Note that a table row really consists of FOUR text rows. )

[2]

d) Write an SQL select statement against the table BN_XML that will display all the neighborhoods in Brooklyn WITHOUT XML TAGS in one column. Do NOT use any other table.

[3]

2) Draw the following JSON structure using the method shown in class. You can use any tool you like, but no hand-drawing. [10]
3) We are going back once again to the table BOROUGH_NEIGH from Question 1) above. Our goal is now to generate a JSON table.

a) Create a new table BN_JSON with one column called JSONDATA of type JSON.

Show the create statement.
b) Write a PL/SQL program that translates data from BOROUGH_NEIGHBOR into JSON and inserts it into the new table BN_JSON. Of course this requires a cursor.

The first rows of BN_JSON should look like this:

```
<table>
<thead>
<tr>
<th>Boro</th>
<th>Neigh</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bronx</td>
<td>Melrose</td>
</tr>
<tr>
<td>Bronx</td>
<td>Mott Haven</td>
</tr>
<tr>
<td>Bronx</td>
<td>Port Morris</td>
</tr>
</tbody>
</table>
```

Show the complete program.

[8]

c) Write a select statement that shows the complete content of the table BN_JSON. All values have to be readable. No object identifiers.

Show the select statement.
Show the result.

[3]

(Show either the complete result or the first 20 table rows and last 20 table rows of the result. Whatever you prefer. Note that a table row really consists of TWO text rows. )

4) NOTE: XML data could be stored in just FIVE rows.
So the data in the example for Question 1) could be stored as follows:

```
<BNDATA>
  <BN>
    <BORO>Bronx</BORO>
    <NEIGH>Melrose</NEIGH>
    <NEIGH>Mott Haven</NEIGH>
    <NEIGH>Port Morris</NEIGH>
    <NEIGH>Hunts Point</NEIGH>
  </BN>
</BNDATA>
```
But I am NOT asking you to program this.

a) Rather, I am asking you to create a table like the example above that represents all and ONLY the Staten Island CB 3 neighborhoods in XML by hand. In other words, this would be a table with one single table row. But that table row would contain 17 text lines.

This is plain SQL. This is NOT PL/SQL.
Show the insert statement and the complete resulting table.
Data in the table should be readable.

The data could also be represented in JSON in a table with 5 rows.

b) Create a table like the example above that represents all Staten Island CB 3 neighborhoods in JSON by hand. In other words, this would be a table with one single table row. But that table
row would contain more than 2 text lines. (It is MORE than 2 text lines, because the long […] will probably overflow and make 3 rows).

This is plain SQL. This is NOT PL/SQL.
Show the insert statement and the complete resulting table.
Data in the table should be readable.
[3]

5) Create a MongoDB collection NEIGHMONGO.

a) Insert from Question 3)b) or 3)c) the first four JSON data sets of the Bronx into
NEIGHMONGO, using simple insert.
[2]

b) Insert from Question 3)b) or 3)c) the first four JSON data sets of Brooklyn, Manhattan,
Queens and Staten Island into NEIGHMONGO, using multi-insert.

In other words you will have four insert statements, each inserting four JSON expressions, for a total of 16 JSON expressions.
[2]

c) Display the complete content of NEIGHMONGO in descending alphabetical order by borough, using the correct “find” statement in MongoDB.

In other words, the Staten Island neighborhoods should appear first.
[2]

d) Display the complete content of NEIGHMONGO in ascending alphabetical order by borough, using the correct “find” statement in MongoDB.

In other words, the Staten Island neighborhoods should appear LAST.
[2]