CS 151A Spring 2015 | Assignment 14

Due Date: Dec 14, 2015 11:00 PM (Late date Dec 15, 11:00 pm) **Points:** 35 point max

General Directions

There are a series of steps to do with an xmltype table.

These tasks focus on the use of XML. Consequently, you **must use XML techniques indicated in the task description** to solve the problems. If the task description says to use a particular function or technique, you must use that function or technique.

Schema rules for the xml (These rules are not enforced by the system.)

- Assume that a book will have exactly one ID, exactly one title, one subject area and these elements will not be empty.
- The bk_id value will be unique for each row
- Each book will have a single price element(bk_price), the value for the price could be 0 or empty; it will not be negative
- Each book will have a single bk_authors element. That element could be empty or could contain 1 to 15 author elements. The value of the author elements will be a last name only with no embedded spaces.

This is an example of a book element. This book has three authors.

```
<book>
  <bk_id>106</bk_id>
  <bk_title>Alice in Oracle Land</bk_title>
  <bk_subject>Fiction</bk_subject>
  <bk_price>35.00</bk_price>
  <bk_authors>
    <author>Dodgson</author>
    <author>Martin</author>
    <author>Carroll</author>
  </bk_authors>
</book>
```

Do not worry about the column widths of your result sets.

I have provided some column commands to help with that; include them in your script. Remember that many of the SQL*Plus commands are not handled in SQL Developer client but will be handled when you run the script in the SQL*Plus client. If you are developing in SQL*Plus, paste in these set commands at the start of each session.

Several of these queries will be easier to do if you use CTEs or inline views to handle the xml methods and then a top level query to handle the formatting. The model for that is

```
With tbl as (
   select . .  this has the XML logic
   )
Select col1, col2, cast(col3 as …) , str( col4, 12,2)
From tbl;
```

The main query is using regular sql functions just to improve the display- not to do filtering. Filtering and extracting the data is done in the xml part. I have had people simply extract the xml to a string and then use sql string functions to do the logic- that is not allowed in this assignment.

Preliminary Steps

Create and populate a table named xml_A14_books using the sql provided.

Tasks

Start your script with the following commands. These will be marked as skipped in SQL Developer but will run in SQL*Plus. The script you turn in for grading needs to be run in SQL*Plus.

```
set Long 999999
```

The next set commands control the column width and display of some of the data.

```
column BookID     format A10
column Title      format A35
column Price      format 9,990.00
column AuthorList format A50
column Subject    format A15
column AvgPrice   format 9,990.00
```

Note: extractvalue will return strings; you may need to cast some values to numbers for formatting or testing.

Task 01: Use the **extractvalue** function to show the book id, title, the price, and the subject of each book. Follow the format shown below.

```
sample display
         BookID     Title                                       Price Subject
         ---------- ----------------------------------- ------------- ---------------
         101        The Art Of SQL                              44.99 Tech
         402        Birdland                                    12.34 Fiction
         415        Adventures in XML                             .00 XML
         485        Schema Refactoring                          76.50 XML
```

Task 02: Use the **extractvalue** function to show the book id, title, and the price; use the **extract** function to show the list of authors as an xml fragment. Display a row for each Fiction book that costs more than $30. Use extractvalue in the Where clause for all tests.

Sample display (I narrowed the title and price columns here, the author list might split over two or more display rows)

```
BookID     Title                          Price AuthorList
---------- ------------------------ --------- ---------------------------------------------
123        Whenever and other data       44.99 <bk_authors><author>Faroult</author></bk_authors>
456        Land of the Giants            35.00
<bk_authors><author>Dodgson</author><author>Martin</author><author>Carroll</author></bk_authors>

789        Birds of a Feather            69.99 <bk_authors><author>Dodgson</author></bk_authors>
```

Task 03: Use the **extractvalue** function to show the book id, title, and the price ; use the **extract** function to show the list of authors as an xml fragment of each Fiction book that costs more than $30. Use **existsNode** in the Where clause for all tests. Format the display as with the previous task.

Task 04: Display the book id, title and list of authors of all books where at least one of the authors is named Martin.

Task 05: Display the book id, title and list of authors of all books where the first author is named Martin.

Task 06: Display the book id, title and list of authors of all books where none of the authors is named Martin.

Task 07: For all books which have at least 4 authors but no more than 10 authors, display the book id and title.

Task 08: Display the book id, title, price, and the price rank for each book. Use the **dense_rank** function; the most expensive book should be ranked 1. Any books with missing prices should be ranked last.

```
sample display
BOOKID     TITLE                                            Price  PRICERANK
---------- ----------------------------------- ------------ ----------
234        Thinking about the final                         69.99          1
345        Thinking about the break                         65.00          2
456        Thinking about next semester                     65.00          2
567        Thinking about nothing much                      50.00          3
678        Thinking about Sets                              45.99          4
```

Task 09: Display the subject and the average price and count for each subject area of books that we have; include the average price and count for all books as the last row. Your subject areas may differ.

```
Sample display
Subject         Avg Price  Number Books
--------------- ---------- ------------
Fiction             104.99           23
Tech                160.00           22
History                               3
SUMMARY             132.45           48
```

Task 10: Display one row with the average price of the books which are classified as SQL books as the first column of the output display and the average price of books which are not SQL books as the second column of the output display.
Suggestion: If you use two inline views that each return a single value; then you can cross join these to get a single row two column result set.

```
Sample display
AvgPrice_SQL AvgPrice_Other
------------ --------------
       45.50          89.56
```