Due Date: Dec 07, 2015 11:00 PM (Late date Dec 08, 11:00 pm) Points: 35 point max

General Directions

These tasks focus on the use of programming techniques to create user defined functions. For this assignment you will create two PL/SQL functions and use them in queries. You need to create and test the functions first, using a variety of values for testing your functions and correcting your functions as needed.

After you have created and tested your functions, you set up a script to show me the source code for these functions and the test queries. Follow the script template for this shown at the end of this assignment.

Functions to be created

You should test for and handle null parameters as explained in the function directions. You do not need to try to handle situations where the user of the function tries to pass in an argument of the wrong data type. A function will not execute if the arguments passed in do not match the data types of the parameters.

Create a function named **BookSize**. The function has one input parameter which is expected to be the number of pages in a book. The function returns a **string** that indicates how many pages the book has. There are 7 possible string values that can be returned.

Use the following function header.

```
create  function  booksize  (
   size_in   integer
   )
return varchar
```

If the book has 200 pages or fewer, it is classified as a "Short" book.
If it has 201- 500, pages it is classified as a "Medium" book.
If it has 501- 1200, pages it is classified as a "Long" book.
If it has 1201 - 4000 pages, it is classified as a "ExtraLong" book.
If it has more than 4000 pages, it is classified as a "SuperLong" book.
If the input argument is null, then the function returns the string message " Input value is missing".
If the input argument is zero or negative, then the function returns the string message " Input value must be a strictly positive value".

Use the **If selection structure** in this function (not a Case expression).

Create a function named **PrevMonth**. The function has two input parameters (`in_date` and `in_mn_adjust`). The first parameter is a date and the second parameter is an integer which is expected to be the number of months to adjust. The function returns a **string** with the format 'YYYY-MM' which is the year and month for the month that is `in_mn_adjust` previous to the first parameter. See the sample return values below.
If the first parameter is null, use the current date as the first argument value.
If the second parameter is null or a negative number, return a null string.

For example:

```
PrevMonth(date '2014-04-10',  2) returns '2014-02'
PrevMonth(date '2014-04-10',  6) returns '2013-10'
PrevMonth(date '2012-04-10', 18) returns '2010-10'
PrevMonth(date '2014-04-10', null) returns null
PrevMonth(null, 0) returns '2015-11'-- if run in November 2015.
```

Function Demonstrations

We want to demonstrate that the function works by supplying a variety of arguments. For task 1 and 2, you are to use the technique described in the notes for this unit to set up a virtual test table for the function.
For task 01, use the demo 6 in the document on testing functions as a model; you can use a CTE if you wish.
For task 02, use the demo 7 or 8 in the document on testing functions as a model; include test run numbers.
For other tasks you use the function with the database tables.

Task 01: **BookSize function test using a virtual table:**
Demonstrate your function by running a query using a virtual test table to supply arguments. Include enough rows to fully demonstrate that your function is correct. The result of running this query should follow this sample run- I have shown only some of the rows.

```
Sample rows only
    testrun   PageCount actual                   expected
---------- ---------- ------------------------- ------------
          4        199 Short                     Short
          5        200 Short                     Short
          7        325 Medium                    Medium
         11        999 Long                      Long
         13       1250 ExtraLong                 ExtraLong
```

Task 02: **PrevMonth: function test using a virtual table:**
Demonstrate your function by running a query using a virtual test table to supply arguments. Include enough rows to fully demonstrate that your function is correct, include tests for nulls and negative values in the virtual table approach. Use the report model shown below. For the Status column, use a case statement to deal with the expected value being null. The sample run shows some rows getting a ***Fail **** message because my function is not yet complete. **Your output should not show fail if your function is correct.**

```
    testrun date_in       month_in actual  expected status
---------- ---------- ---------- ------- -------- ----------
          1 NULL                0 NULL    2015-11  ***FAIL***
          5 2012-04-01          0 2015-04 2012-04  ***FAIL***
          6 2012-04-01          1 2012-03 2012-03  pass
          7 2012-04-30          6 2011-10 2011-10  pass
         10 2012-04-19       NULL NULL    NULL     pass
         11 2012-04-19        -25 NULL    NULL     pass
```

Task 03: Use the **BookSize** function to produce a display as shown here**.** Use the books table as the data source for this query. The order of the rows must match the order shown here. Do not include any books with a null or negative page count. It is possible that the books table might not have any books of one of these size categories; in that case your result set would not have a row for that size category.

```
Sample rows only

BOOKSIZE    NUMBOOKS
--------- ----------
Short            125
Medium           289
ExtraLong          3
SuperLong         10

4 rows selected.
```

Task 04: Use the **BookSize** function to produce a display as shown here**.** Use the books table as the data source for this query. The order of the rows must match the order shown here. Do not include any books with a null or negative page count. Display a row for each of the size categories, even if we have no books

with a page count for that category; display a count of 0 for that row. (Hint: consider a virtual table of size categories similar to the calendar task in a previous query .)

```
BOOKSIZE    NUMBOOKS
--------- ----------
Short             14
Medium            43
Long              32
ExtraLong          3
SuperLong          0

5 rows selected.
```

Task 05: Use the **PrevMonth** function to display the number of orders we had in the previous two months and the number of customers we have who have at least one order in the previous two months. The term "previous two months" means any date in the two month before the current month. So if you run the query in April 2015, the query will return data for Feb 2015 and March 2015. You need to derive the months based on the system date. Use the books tables as the data source for this query.
Do not redo the calculations for a previous month in the task- use the function for this calculation. Do not use any variables in the task.

```
NumberOrders NumberCustWithOrders
------------ --------------------
         234                   98
```

Template for the Script file

This is similar to the regular template but includes two statements to show me the source code for your functions and there are several extra column commands to use at the top of the file.
It is essential that you read this spool file. If you tend to write long lines in code, you might find that your code has been truncated and cannot be graded. In that case, rewrite your code with shorter lines. If I cannot read the entire source code you have turned in, that function will receive a zero and you cannot resubmit it.

```
set echo on
set feedback 1
set pagesize 999
set trimspool on
set linesize 200
set tab off
clear columns

column line format 999
column text format a70
column text wrapped

replace this line with comment with your name


/* function source code */
      select   line, text
      from     user_source
      where    type in ('FUNCTION')
      and      name in ('BOOKSIZE')
      order by name, line;

/* function source code */
      select   line, text
```

```
      from     user_source
      where    type in ('FUNCTION')
      and      name in ('PREVMONTH')
      order by name, line;

/*  TASK 00 */
select user, sysdate
from dual;

/*  TASK 01 */
replace this line with your SQL for task 01

/*  TASK 02 */
replace this line with your SQL for task 02

/*  TASK 03 */
replace this line with your SQL for task 03

/*  TASK 04 */
replace this line with your SQL for task 04

/*  TASK 05 */
replace this line with your SQL for task 5
```