Due Date: Saturday, November 23, 2013 11:00 PM
Points: 40 points max
Turn In: The script and spool files turned in via the assignment drop box

General Directions

Create the functions in the a_testbed database and use the full name(a_testbed.fnc_name) when using the function.

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

After you have created and tested all of your functions, you set up a script to show me the source code for these functions. See the script at the end of this assignment for this file.

You also need to create a script file that tests your functions and demonstrates that they work properly. The minimal set of tasks for testing your code is provided below. You may add additional tests if you wish.

Warning: The files to be turned in for this assignment differ from the normal files you submit. There are **three** files to be turned in for this assignment: One should show the source code (spooled file); the second the test script and the third the spooled result of running the testing script.

1) A13*_yourLastName*_SourceCode. *LST* a spool file that shows the source code for your functions (See the directions at the end of the assignment). You create this file by running the script shown at the bottom of this assignment**.** Do not turn in the script file for this since I gave you that code; turn in the spooled source code listing.

2) A13*_yourLastName.SQL* a script file that shows the sql for your tasks testing the functions.

3) A13*_yourLastName.LST* a spool file that shows the results of your tests of the functions.

Tasks: 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 **NameFormat.** The function has three input parameters. The first is expected to be a 'F' or a 'L', the second and third parameters are strings which we expect to be the last and first names. Use a varchar(25) for the second and third parameter. The function returns a string which is a formatted name.
If the first argument is 'F', then the name is formatted as firstName space lastName.
If the first argument is not 'F', then the name is formatted as lastName comma space firstName
If the first name is a null, then return only the last name- with no extra space or comma.
You should assume the last name will never be null.
Samples

```
a_testbed.nameFormat('F', 'Smith', 'John') #      returns John Smith
a_testbed.nameFormat('L', 'McDermott', 'Annie') # returns McDermott, Annie
a_testbed.nameFormat('L', 'McDermott', null)#     returns McDermott
```

Use the file header

```
create function a_testbed.NameFormat (
    p_mode    char(1)
,   lastname  varchar(25)
,   firstname varchar(25)
)   returns varchar(52)
```

Create a function named **InternalBlankCount**. The function has one string parameter and returns an integer which is the number of blanks in the string, not counting any leading or trailing blanks.
For example: the string `'  asgn 14  '` would return a value of 1:
the string `' Sign up for the final exam'` would return a value of 5. If the input parameter is null or a zero-length-string, the return value is 0 since those parameters do not have any internal blanks.

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 long the book is.

If the book has 200 pages or fewer, it is classified as a "Mini" book.
If it has 201- 500, pages it is classified as a "Small" book.
If it has 501- 1000, pages it is classified as a "Medium" book.
If it has 1001- 1500, pages it is classified as a "Large" book.
If it has more than 1500, pages it is classified as a "TooLong" book.

If the input argument is null, then the function returns the string message "Invalid Input".
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_count`). The first is a date and the second is an integer which is expected to be the number of months. The function returns a string with the format 'YYYY-MM' which is the year and month for a the month that is `in_mn_count` previous to the first parameter. Use the same definition as for this as was used in assignment 10. If the first parameter is null use the current date.
For example, `a_testbed.PrevMonth('2012-05-19', 6)` returns '2011-11'

Tasks: Function Demonstrations

We want to demonstrate that the function works by supplying a variety of arguments. For some tasks you are to use the technique described in the notes for this unit to set up a virtual test table for each function as required by a task. For other tasks you use the function with the database tables.

Task 01: **NameFormat:** Demonstrate your function by running a query using a virtual test table to supply the arguments. Include enough rows to fully demonstrate that your function is correct.

Task 02: Using the **NameFormat** functionwith the mode 'L', display the customer id and formatted name of the first 10 customers ordered by the customer id.

Task 03: Use the **NameFormat** function to display a single column that shows the books id and title formatted as shown here. `ID 1106: SQL for Smarties .` Use a column alias of Book. Limit the display to 10 rows. Titles that are longer than 25 characters will be limited to 25 characters. Do *not* use a null parameter for this.

Task 04: **InternalBlankCount.:** Demonstrate your function by running a query using a virtual test table to supply the arguments. Include enough rows to fully demonstrate that your function is correct.

Task 05: **BookSize:** 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.

Task 06: Use the **BookSize** function to produce a display as shown here**.**

```
Sample rows only
BookSize    NumBooks
-------- -----------
Mini               7
Small             29
Medium            23
Large              0
Too Long           1
```

Task 07: Use the **PrevMonth** function to display customer id and name for all customers who have at least one purchase in every one of the three months as defined in assignment 10 - a three month period starting 6 months ago and extending for two months.

Task 08: 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 the previous two months. The term "previous month" means any date in the month before the current month. So if you run the query in July 2012, the query will return data for May-June 2012.

Displaying the Function Source Code:

Use the show create function fncX command for this. Use the \G command delimiter as shown here. Be certain to include your name as a comment in the script.

```
show create function  BookSize\G
show create function  InternalBlankCount\G
show create function  NameFormat\G
show create function  PrevMonth\G
```