Deadline:
Due by 11:59 pm on Thursday, February 23

How to submit:
Submit your files for this homework using ~st10/318submit on nrs-projects, with a homework number of 4

Purpose:
To give you more PL/SQL, XHTML, and CSS practice, along with some JavaScript practice

Important notes:
• In your JavaScript code, you are expected to indent the contents of all { }’s by at least 3 spaces, and each { and } should be on its own line, even with the preceding line (as seen in posted class examples).
  – also, all JavaScript functions are expected to start with a comment that at least gives its name, and a purpose statement which explicitly describes what the function expects and what it does and/or returns
• Remember to follow the CIS 318 SQL and PL/SQL Style Standards as given in the CIS 318 Homework 1 handout for all SQL and PL/SQL code.
• Make sure that you have executed the scripts create-bks.sql and pop-bks.sql, and that the bookstore tables are successfully created and populated.
• Unless explicitly indicated otherwise, for the entire semester, all XHTML pages must include the image-link to the W3C XHTML 1.1 validator as shown in example pages xhtml-template.html and xhtml-css-template.html, and all must validate as valid XHTML 1.1. Each page that does not will cause a loss of points on the homework problem involved.
  – if a page also uses CSS, it must include the image-link to W3C CSS validator as shown in example page xhtml-css-template.html, and it must validate as valid CSS level 2.1. Each page that does not will cause a loss of points on the homework problem involved.
• I'm not requiring specific indentation for XHTML yet - I reserve the right to do so, however, if necessary. In the meantime, find a readable way of indenting it, and consistently do so...
• However, for CSS styles, you are expected to indent the contents of all { }’s by at least 3 spaces, and each { and } should be on its own line (as seen in posted class examples).
The Problems:

Problem 1

Create a SQL script 318hw4.sql, and start it off with comments including your name, CIS 318 - Homework 4, and the last-modified date.

Next, add the command to run the pop-bks.sql script each time this script is run, so that you have "fresh", original versions of these tables. (Their contents are mucked with below, so it is important that these are "reset" here.)

Include the command to set serveroutput on, followed by a SQL*Plus spool command to spool the results of running this SQL script to a file named 318hw4-out.txt. Then write a SQL*Plus prompt command that says problem 1. (You may add additional prompt commands around this to make it more visible, if you would like.)

Now, for a more interesting PL/SQL stored function that involves some exception handling: design and write a PL/SQL stored function sell_book that will represent the sales transaction of selling one or more copies of a particular single book. So, you will not be shocked to hear that sell_book expects two parameters (in this order): an ISBN representing the book being sold, and an integer representing the quantity being sold.

Then sell_book returns an integer representing a results code, letting the caller know if the sales transaction for this book was successfully completed. We'll describe its possible values further below.

sell_book's purpose is to manage the database fields relating to the inventory of this ISBN. Here are its tasks (assume they are based on this scenario's "business rules"):

• reduce the qty_on_hand field of the title table for this ISBN by the number of copies being sold

• determine if we need to note that an order is now needed for this ISBN (because of this sale):
  - when is it needed?
  - It is NOT needed yet if the qty_on_hand for this title is larger than that title's order_point; the stock is not low enough, yet.
  - It is NOT needed at this point if the qty_on_hand for this title is less than or equal to that title's order_point, but it is already on-order.
  - And it is NOT needed if the qty_on_hand for this title is less than or equal to that title's order_point, it is NOT on order yet, but it DOES have a pending order_needed row already.
  - ...so, it is ONLY needed if the qty_on_hand for this title is less than or equal to that title's order_point, it is NOT on order yet, and it does NOT have a pending order_needed row already...! (whew!)
  - be sure to make appropriate use of is_on_order (from Homework 2, Problem 2, whose example solution is available on the course Moodle site if you need it...) and pending_order_needed (from Homework 3, Problem 2) in determining this;
only if it IS needed, then, should sell_book call insert_order_needed (from Homework 2, Problem 1) appropriately to make an entry into the order_needed table.

- Use the ISBN from the ongoing transaction;
- use the order_qty from the title table for this ISBN as the value of the order_qty attribute of the order_needed table.

BUT, of course, there's always the chance that sell_book might receive inappropriate arguments. It should protect against these problems:

- an ISBN that doesn't exist in the title table. (Let the system raise this NO_DATA_FOUND exception; your procedure should merely be able to handle it.)
  - The results code should be set to -1 in this case.
  - Make sure any changes made up to this point by this procedure get un-done. (This is a transaction, after all...)

- a value for the number of copies being sold that is not greater than zero. (Raise this exception yourself: a user-defined exception.)
  - The results code should be set to -2 in this case;
  - again, you should make sure any changes made up to this point get un-done.

- a value for the number of copies being sold that is greater than the current qty_on_hand for this ISBN. (Raise this exception yourself, also: another user-defined exception.)
  - The results code should be set to -3 in this case;
  - any changes made by sell_book up to this point should be un-done.

- handle any other exceptions that occur, setting the results code to -4 in this case, and un-doing any changes made by sell_book up to this point. (This is purely defensive coding; such an exception will probably not actually be raised.)

- If no exceptions are raised, set the results code to 0. The caller can look at the results code in the out parameter to see if his/her book sale transaction succeeded or not.

This is a transaction -- don't forget to commit your database updates. Remember that we have an atomic transaction here, so the collection of database updates should be committed all together or not at all. It would be wise then, I think, to start this function with a commit. And then this function should have another commit statement after (if) all the database interactions have successfully completed. (And what statement does this suggest should be included in each exception handler within the exception section??)

Then, write a small tester stored procedure named test_sell_book -- it takes an ISBN and a quantity to be sold as its two parameters, and it prints to the screen a descriptive message that shows what results code sell_book returned.

To vigorously test these takes quite a bit of testing code; you'll find the code for testing these in a posted file prob1-test-code accompanying this homework handout. Paste this code after your code for the above procedures, and be sure to inspect your 318hw4-out.txt file results carefully to see if the tests passed.
And, as always, you may add additional testing calls if you would like.

Follow all of this with spool off and rollback commands; submit your final versions of 318hw4.sql and 318hw4-out.txt

**Problem 2**

Here are several additional tidbits about XHTML and JavaScript:

- NOTE: You PROBABLY won't need this for THIS problem, but NOTE that, for several reasons, including being able to use comparisons like < or operators like && in your JavaScript without the XHTML validator complaining, you may want to use a special style of comment **inside** your JavaScript script tags -- these start normally (<-- ) but end unusually ( //--> )

  That is, you want:

```html
<script type="text/javascript">
<!--
    // put your JavaScript here
-->//-->
</script>
```

  – You don't need this when using these operators in external JavaScripts, but only when using them in JavaScripts actually within the XHTML page.

- We mentioned that many HTML element objects have methods -- for example, have you perhaps wondered how you can indicate that the cursor should be moved or put in a particular textfield?

  – You can do this by calling the **focus()** method for that textfield object.

- You can attempt to convert a value to type Number by using the JavaScript built-in function **Number** (it takes one argument, and attempts to return that argument in numeric form). If it can't, because it is not a number, it returns that special value **NaN** (for Not a Number).

- How can you tell if something has the special value **NaN**? You can use the function **isNaN** to do so -- **isNaN** expects one argument, and returns **true** if its value is the special value **NaN**, and returns **false** otherwise.

  – (note that you CANNOT use == or === to see if something is this special **NaN** value!!)

I think, between the in-class examples and the above, perhaps with reference to the Mozilla Developer Network JavaScript Guide, you now have what you need for the following:

- You will find an XHTML page, **hw4-warmup.html**, which uses the little external css file **my-align-css-plus.css** (now augmented with some spacing styles to help add space around buttons) and external JavaScript file **hw4-functions.js**, on the public course web page along with this handout. **hw4-warmup.html** is valid XHTML 1.1.

- When you read **hw4-warmup.html**'s source code, you will see that there are some areas marked **FILL IN** that you are expected to replace with appropriate information. And when you read **hw4-functions.js**, you will see that it likewise contains some areas marked **FILL IN** that you
are expected to replace with appropriate contents, including completing some incomplete JavaScript functions.

− Make local copies of both of these files, and replace all of the FILL IN instances with appropriate contents.

− Each of the functions in hw4-functions.js is an onclick event handler for one of the buttons in hw4-warmup.html. For each of these, fill in its body, making sure that each meets the specifications given in its opening comment block.

− Maintain the unobtrusive JavaScript style being attempted here.

Submit your resulting files hw4-warmup.html and hw4-functions.js.

Problem 3

If a user needs a username and password to access the data in a database, then you can imagine that an opening page that starts by requesting this information from a user is quite common.

Design an XHTML page hw4-login.html containing a nicely-designed form asking the user for a username and password, that meets the following specifications:

• include a comment that includes the URL I can use to view your page, if I choose; for full credit, this URL must successfully display your page when used;

• use an external CSS stylesheet to control how it looks - it is fine if this is from a previous homework;

• include your name visibly somewhere in the displayed page;

• be sure to use a password field (rather than a plain textfield) for the password;

• use a \texttt{<table>} to line up the username and password text and fields nicely;

• follow this table with well-placed submit and reset buttons;

• this form can have any URL you'd like as its action's value, but use a method of "post";

• include an external JavaScript \texttt{ck-login.js} that contains a JavaScript function that expects nothing and returns true if there is something contained within both username and the password fields whose \texttt{id} attributes have the value that those fields just happen to have in hw4-login.html, and returns false otherwise;

• this form should be set up so that, on submit, its data is only submitted to the web server if that JavaScript function returns true at that point.

  − how should your JavaScript let the user know that something is awry if that JavaScript function returns false? You can either use an alert popup to let him/her know, or if you want to find some other means of doing so (such as inserting a paragraph or header element into your page, for example), you may.

• For full credit, use unobtrusive-style JavaScript for this.

Submit your resulting files hw4-login.html, ck-login.js, and the external .css file you used.