CS 328 - Homework 5

Deadline
11:59 pm on Friday, February 28, 2014

How to submit
Submit your files for this homework using ~st10/328submit on nrs-projects, with a homework number of 5

Purpose
To practice a little with JavaScript, to practice a little more with CSS3, and to set up a PL/SQL stored function for future use.

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 EXCEPT for anonymous functions used in assignment statements, in which case the { and } are indented. See the 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.
  – We will avoid the use of JavaScript within the body element of a page, in an attempt at unobtrusive-style JavaScript.
  – And, (for now, at least), any HTML5 page that uses JavaScript is expected to include a noscript element within its body element that warns a user displaying it within a browser that does not support JavaScript.
• Remember to follow the CS 328 SQL and PL/SQL Style Standards as given in the CS 328 Homework 1 and Homework 2 handouts 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 web pages submitted are expected to use "strict" style HTML5, as discussed in class and in the course textbook.
• Likewise, unless explicitly indicated otherwise, all web pages submitted are expected to include the links to the W3C experimental HTML5 validator and the CSS3 validator as well as the link to http://validator.nu/ as shown in example page html5-template.html, and all must validate/pass the tests from all 3. Each page that does not could cause a loss of points on the problem involved.
the EXCEPTION is that, if your only http://validator.nu/ error is due to a noscript element within your body element, that is OK and will NOT be counted off for!!

• I'm not requiring specific indentation for HTML5 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 rules, you are expected to indent the contents of all '{ }'s by at least 3 spaces, and each '{ and } should be on its own line, lined up with the beginning of the selector (as seen in posted class examples).

• Unless explicitly indicated otherwise, you are expected to use external CSS style sheets, and not internal or inline CSS style sheets.

• HTML5 forms should not use the table element for layout purposes -- CSS should be used for such layout, instead. The table element should only be used for truly-tabular data.

Problem 1

Create a SQL script 328hw5.sql, and start it off with comments including your name, CS 328 - Homework 5, 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 328hw5-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 6, 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 3, 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.)
  - sell_book should return a results code of -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.)
  - sell_book should return a results code of -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.)
  - sell_book should return a results code of -3 in this case;
  - any changes made by sell_book up to this point should be un-done.

- handle any other exceptions that occur, returning a results code of -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, return a results code of 0. The caller can look at the returned results code value 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??)

To vigorously test this takes quite a bit of testing code; you'll find the code for testing this 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 328hw5-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 a spool off command; submit your files 328hw5.sql and 328hw5-out.txt

**Problem 2**

Consider your HTML5 page bks-splash.html and the external CSS3 file bks.css from Homework 4.

Make new copies of these in a different directory, since you will be modifying them and you don't want to change Homework 4's version of them (since that could affect your Homework 4 grade!)

Add a link from your index.html on nrs-projects to this new version of bks-splash.html (making clear somehow this is Homework 5's version, and NOT removing the earlier link to Homework 4's version!!)

- Note: In index.html's unordered list, if you would like to include an unordered sub-list for Homework 5's links, that would be fine indeed.

Now consider this new version of bks-splash.html.

- Write an external JavaScript ck-login.js that contains a JavaScript function that expects nothing and returns true if there is something contained within both username and password fields whose id attributes have the value that those fields just happen to have in bks-splash.html, and returns false otherwise.

- Modify bks-splash.html so that, on submit, its form's data is only submitted to the application tier 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 may choose -- you may use an alert popup, or you may change some page element's value or contents, or you may find some other means of doing so.

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

Submit your resulting versions of bks-splash.html, bks.css, and ck-login.js.

**Problem 3**

I forgot to have you create an external CSS for your additional database's title page, from Homework 3, Problem 6!
Make a new copy of this title page in a different directory, since you will be modifying it and you don't want to change Homework 3's version of it (since that could affect your Homework 3 grade!)

Add a link from your index.html on nrs-projects to this new version of your additional database's title page, (making clear somehow this is Homework 5's version, and NOT removing the earlier link to Homework 3's version!!)

3 part a
Create an external CSS file to style your title page, and make the changes needed to your title page so that it is indeed styled by that external CSS file. Make sure it is attractive and nicely laid-out.

3 part b
We'd like to also make sure that this page's form is not submitted if there is not something contained within both its username and password fields.

IF you would like, you can use Problem 2's ck-login.js for this -- or, you can create a new external JavaScript to accomplish this. Either way, make sure that, on submit, your title page's form data is ONLY submitted to the application tier if both of its fields contain something.

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

Submit your resulting versions of your title page, the external CSS, and the external JavaScript file.

Problem 4

4 part a
Determine at least one numeric computation you would like to perform.

Then, design an HTML5 page number-fun.html that includes:

• your name
• CS 328
• at least two textfields (so the user can enter the needed numbers for the computation you chose)
• at least one button element (so the user can indicate that he/she would like for a computation to now be done)
• (you may also include additional elements as you would like)

4 part b
Create an external CSS file number-fun.css to style your page number-fun.html.

• Include rule(s) to make sure your page is attractive and nicely laid-out.

• Include rule(s) that will make sure that the textfields' contents will be right-justified (since that works well for numbers).
4 part c

Using unobtrusive-style JavaScript, write an external JavaScript `number-fun.js` to now perform the numeric computation(s) you decided upon, using `number-fun.html`'s textfield's contents, when its button element is clicked, making sure to somehow show the computation's results to the user. Do what is needed for `number-fun.css` to use this external JavaScript `number-fun.js`.

Add a link from your `index.html` on nrs-projects to your resulting `number-fun.html`.

Submit your resulting versions of `number-fun.html`, `number-fun.css`, and `number-fun.js`. 