

CS 328 - Homework 4

Deadline

Due by 11:59 pm on **SUNDAY, February 21, 2016** [still "nonstandard" deadline...]

How to submit

Submit your files for this homework using `~st10/328submit` on nrs-projects, with a homework number of 4.

Purpose

To practice some more with PL/SQL stored subroutines and "strict"-style HTML5 (now also including forms), and to get some introductory practice with CSS3.

Important notes

- Note: you *may* be presenting versions of some of these HTML5 pages to the class at some point.
- None of the problems below happens to involve styling your `index.html` on nrs-projects with an external CSS - but you may certainly do so, if you'd like! It would be good additional practice.
- Remember to follow the **CS 328 SQL and PL/SQL Coding Standards** as given in the CS 328 Homework 2 handout and the public course web site 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.
- Remember to follow the **CS 328 HTML5 Coding Standards** as discussed in class and as given on the public course web site for all HTML5 documents.
- And, now, also remember to follow the **CS 328 CSS3 Coding Standards** as discussed in class and as given on the public course web site for all CSS3 that you write, also!

Problem 1

CAVEAT: Yes, I realize that a sequence **would** be preferred here. This problem's purpose is give you more practice writing PL/SQL subroutines that use other PL/SQL subroutines.

Create a file `328hw4.sql`.

(Make sure that you have a copy of `pop-bks.sql` in the same directory as `328hw4.sql` -- it is called in this problem's testing script, to make sure the tests are run on "fresh", original versions of these tables, before the tests muck with them. Note that this script happens to already end with a `commit;` command.)

Start your `328hw4.sql` file with the following:

- comments containing at least your name, CS 328 - Homework 4, and the last-modified date
- 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 `328hw4-out.txt`
- followed by a `prompt` command including your name

Be sure to `spool off` at the end of this script (after your statements for the remaining problems).

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.)

Remember `next_ord_needed_id` from Homework 3, Problem 1? You are now going to write a PL/SQL stored procedure that makes use of that stored function. (Remember, `next_ord_needed_id` is already stored in your database -- you can simply call it from this script, you need not recreate it.)

When we want to add rows to table `order_needed`, we will call a PL/SQL stored procedure `insert_order_needed` whose parameters are the desired ISBN and order quantity for the desired order. This will do the following:

- insert a new row into `order_needed` using a key returned by calling `next_ord_needed_id`, the parameter ISBN and parameter order quantity, and the current date. (It should NOT fill the `date_placed` attribute for `order_needed`; that attribute should have the value null.)
- (since this order is needed, but not yet placed, you should NOT do anything to the title's `on_order` attribute yet -- when an order is actually placed, THAT's when the title's `on_order` attribute should be updated. We'll address that in another homework.)

Separately (outside of `328hw4.sql`) run the posted testing script

`insert_order_needed_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `insert_order_needed_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `insert_order_needed` code in your `328hw4.sql` script if you would like.

Problem 2

In `328hw4.sql`, write a SQL*Plus `prompt` command that says `problem 2`. (You may add additional `prompt` commands around this to make it more visible, if you would like.)

Consider again the `order_needed` table in the bookstore database. The idea/hope here is that, when a title's quantity on hand becomes less than the `order_point`, then a row should be added to the `order_needed` table indicating that, well, an order is needed for that title. Recall -- as noted above -- that, initially, the `date_placed` attribute for that new `order_needed` row is null -- the order is needed, but it has not yet been placed.

When, later, the order IS placed, then the `date_placed` attribute for the corresponding `order_needed` row is filled accordingly. (I hope to have you write a trigger on an upcoming homework that will take care of this.)

At any given time, then, the rows in `order_needed` that have a null value for `date_placed` indicate orders that, well, need to be made. One could think of these as "pending" `order_needed` rows.

There could be times (indeed, as part of an application you will be working on later this semester) where you would like to know if, for a given ISBN, there is a "pending" `order_needed` row for that ISBN (if there is a row with that ISBN whose `date_placed` attribute is null).

Write this little PL/SQL stored function `pending_order_needed` that expects an ISBN, and returns a boolean value indicating whether or not that ISBN has such a "pending" `order_needed` row.

Separately (outside of `328hw4.sql`) run the posted testing script

`pending_order_needed_test.sql` posted along with this homework handout (note that it uses the stored function `bool_to_string` which was created by one of Homework 3's testing SQL scripts, `is_on_order_test.sql`, so it should still be in your database). Make sure the tests all pass, and submit the file it spools with the test results, `pending_order_needed_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `pending_order_needed` code in your `328hw4.sql` script if you would like.

Make sure that you submit your `328hw4.sql` and `328hw4-out.txt` files as well as the `*_test_out.txt` files for each of the subroutines in Problems 1 and 2.

Problem 3

Consider your `bks-splash.html` from Homework 3, Problem 5.

Make a **new** copy of this page in a **different** directory, since you will be modifying it and you don't want to change Homework 3's version of this (since that could affect your Homework 3 grade!)

Add a link from your `index.html` on `nrs-projects` to this **new** version of `bks-splash.html` (making clear in some OBVIOUS fashion this is **Homework 4's** version, and **NOT** removing the earlier link to Homework 3's version!!)

Eventually, it is going to be handy for a qualified user to log in from here. So, add a form that allows the user to enter and then submit their Oracle username and password:

- your form's action can be the URL of any functioning web page -- later, we'll replace this with a URL that will actually attempt to process this form.
- give your form a `method` attribute with a value of `"post"` (although while you are debugging you can use `"get"`, as long as you replace it with `"post"` for the version that you submit)
- for the password entry, use a **password field** instead of a textfield (this is an `input` element with `type="password"` -- it is VERY much like a textfield, except when a user types into it, little circles or asterisks are displayed instead of what is actually typed.

BUT NOTE!!!! that it does NOT obscure the actual password submitted in any way -- you still need `https` to encrypt it en route, and if your form uses `method="get"`, the password typed in WILL be displayed at the end of the `action` attribute's URL in plain text...!)

- don't forget a submit button! (but you likely remember that including one of those in a form is one of the course coding standards, of course)

On a later problem below, you'll add some CSS3 styling to this -- and on a later homework, you'll format/layout this form using CSS3. So, just concentrate on getting the appropriate form controls in here for this problem. (And remember that submitting problem versions frequently is encouraged, so it is fine for you to submit this problem's version of `bks-splash.html` early in the week and its modified version later in the week.)

Problem 4

Consider Problem 1's PL/SQL stored procedure `insert_order_needed`. What might an HTML5 form look like that could submit the information needed by this procedure to the application tier, so that the application tier could call this stored procedure (in conjunction the data tier)?

Create an HTML5 document `insert-ord-needed.html` containing such a form that meets the following specifications:

- Include the URL from which it can be run within a comment near its beginning
- Try to select HTML5 form control elements that would be appropriate and reasonable for allowing the user to indicate each "piece" of necessary information expected by this stored procedure.
 - (If your choice of control happens to require that you hard-code parts now that you would likely want to dynamically generate later in the semester, that is fine at this point.)
- This form also needs to allow the user to enter their Oracle username and password. For the password entry, use a password field rather than a textfield (as described in Problem 3).
- This HTML5 document should visibly include, in its `body` section, your name, CS 328, and the name of your bookstore from `bks-splash.html`.
- Your form's action can be the URL of any functioning web page -- **later**, we'll replace this with a URL that will actually attempt to process this form, building a call to the PL/SQL stored procedure `insert_order_needed`.
- Your form's method should be "post" (although while you are debugging you can use "get", as long as you replace it with "post" for the version that you submit).
- Do **NOT** add a link from your `index.html` on `nrs-projects` to your resulting `insert-ord-needed.html`

On a later problem below, you'll add some CSS3 styling to this -- and on a later homework, you'll format/layout this form using CSS3. So, just concentrate on getting the appropriate form controls in here for this problem. (And remember that submitting problem versions frequently is encouraged, so it is fine for you to submit this problem's version of `insert-ord-needed.html` early in the week and its modified version later in the week.)

Problem 5

To give you a chance to practice with some additional HTML5 elements (especially form control elements)... using the posted `html5-template.html` as the initial basis, create an opening "strict" HTML5 document `hw4-play.html` that simply permits you to practice with some of these. It should include at least:

- the URL from which it can be run included within a comment near its beginning
- your name visibly included somewhere within its `body` element.
- CS 328 visibly included somewhere within its `body` element
- a level-1 heading of your choice
- a `table` element with at least 3 rows and 3 columns, with content of your choice

- a form that includes at least the following controls:
 - a submit button (of course)
 - at least 3 grouped radio buttons
 - at least 3 checkboxes
 - at least one drop-down menu/box
 - at least one textfield
 - at least one `textarea` element (you can read about this element in the course textbook)
 - your form's action can be the URL of any functioning web page
 - give your form a `method` attribute with a value of "post" (although while you are debugging you can use "get", as long as you replace it with "post" for the version that you submit)
 - (you may also add additional element(s) from the course textbook that you are interested in trying out)
- Do **NOT** add a link from your `index.html` on nrs-projects to your resulting `hw4-play.html`

Again, you'll add some CSS3 styling to this on a problem below -- and on a later homework, you'll format/layout these forms using CSS3. So, just concentrate on getting the desired form controls in here for this problem. Your resulting `hw4-play.html` file is not quite ready to submit yet (unless you want to submit a "partial" version early on).

Problem 6

Consider Problem 5's `hw4-play.html`. Surely it needs some style!

Create an external style sheet `hw4-play.css` and modify `hw4-play.html` so that it uses this external style sheet as well as `normalize.css`. Meet the following additional specifications:

- Include a comment including at least your name and the last-modified date in `hw4-play.css`.
- Include at least **five** distinct rules in your external style sheet that have a visible effect on this page (but you can certainly add more if you are inspired!).
 - At least one rule should style one or more aspects of an element's font in some visible way.
 - At least one rule should style one or more aspects of some element's color in some visible way.
 - Remember that your course textbook describes many interesting properties in Chapters 3 and 4.
 - You don't have to use layout-related CSS3 yet, unless you want to. You may be getting an opportunity to do that on a later homework.

Submit your files `hw4-play.html` and `hw4-play.css`.

Problem 7

Consider your HTML5 documents `bks-splash.html` from Problem 3 and `insert-ord-needed.html` from Problem 4.

- Design a first version of an external style sheet `bks.css` that you would like to use for these and other pages we create making use of the database created by `create-bks.sql`.

- Include a comment including at least your name and the last-modified date in `bks.css`.
- Include at least **five** distinct rules in your external style sheet that have a visible effect on these pages (but you can certainly add more if you are inspired!).
- You don't have to use layout-related CSS3 yet, unless you want to. You will be getting an opportunity to do that later.
- Modify these two HTML5 documents (`bks-splash.html` from Problem 3 and `insert-o-needed.html` from Problem 4) to use this style sheet as well as `normalize.css`.

Submit your resulting `bks.css` and your `bks-splash.html` and `insert-o-needed.html`.