CS 328 - Homework 5

Deadline

Due by 11:59 pm on **Sunday, March 5, 2017**

Purpose

To set up a PL/SQL stored function for future use, and to practice with CSS3, including formatting & laying out forms using CSS3.

How to submit

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

Important notes

- **DO NOT USE ANY CSS FRAMEWORKS FOR THIS HOMEWORK** -- one of this homework's purposes is to provide you with some practice with "plain" CSS3.
- 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 remember that these include the requirement that, to receive full credit, all of your HTML5 documents must validate as strict-style HTML5.)
- 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

As a small warm-up -- let's give you an immediate reason to refer to those newly-posted CS 328 - CSS3 coding standards! (Note that they can be found in the References section on the public course web site, AND in the CSS references page linked to from the public course web site, AND on the "Homeworks and handouts" page along with this homework handout.)

Make a file `328hw5-css.txt`, and start this file with your name and the last modified date. Then answer the following questions based on the posted CS 328 - CSS3 coding standards, preceding each answer with the part that it is for (1 part a, 1 part b, etc.).

1 part a

Assume you have an external CSS3 style sheet in a file `my-style.css`, and assume that it happens to be in the same directory as an HTML5 document that you would like to be styled using this style sheet.

Give the entire element that you are expected to use for this, according to class style standards.

1 part b

Within what element inside your HTML5 document is your answer to **1 part a** expected to be placed? (Just give the name of this element.)

1 part c

Should your answer to **1 part a** be placed **before** or **after** the element that causes this HTML5 document to be styled using `normalize.css`?

1 part d

Consider the following units: `em` `px` `pts` `%`

Which **two** of these are considered to be preferable, based on the course style standards?

1 part e

Based on the course style standards, is it considered acceptable to use a `table` element to layout an HTML5 form?

Problem 2

Create a file `328hw5.sql`.

(Make sure that you have a copy of `pop-bks.sql` in the same directory as `328hw5.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 `328hw5.sql` file with the following:

- comments containing at least your name, `CS 328 - Homework 5 - Problem 2`, 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 `328hw5-out.txt`
- followed by a `prompt` command including your name
- followed by a `prompt` command that says `problem 2`. (You may add additional `prompt` commands around this to make it more visible, if you would like.)

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

function `sell_book`

Now, for a PL/SQL stored function that makes use of several earlier stored subroutines and 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 attributes relating to the inventory of this ISBN. Here are its tasks (assume they are based on this scenario's business rules):

- since this is a transaction, start this function by by committing the current database state.
- determine if its arguments are appropriate!
  - what if the ISBN doesn't exist in the `title` table? Write a query such that a `NO_DATA_FOUND` built-in exception will be thrown if this is the case,
    - and write exception-handling code to have `sell_book` return a results code of -1 if this exception occurs,
    - and because you'd like to void the transaction at this point, be sure to `rollback` any changes thr function may have made up to this point.
  - what if someone tries to call this with a number of copies that is not greater than zero? For some ADDITIONAL exception-handling practice, raise a **user-defined exception** in this case,
    - and write exception-handling code to have `sell_book` return a results code of -2 if this exception occurs,
    - and because you'd like to void the transaction at this point, be sure to `rollback` any changes thr function may have made up to this point.
    - (hint: there's an example of a user-defined exception in the Week 2 Labs posted example `insert-tool-ex.sql`)
  - what if someone tries to sell a number of copies greater than the current `qty_on_hand` for this ISBN? AGAIN, for some ADDITIONAL exception-handling practice, raise another **user-defined exception** in this case,
    - and write exception-handling code to have `sell_book` return a results code of -3 if this exception occurs,
    - and because you'd like to void the transaction at this point, be sure to `rollback` any changes thr function may have made up to this point.
- BUT -- if **no** exceptions have been raised at this point -- 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 add a row to `order_needed`, if 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.
    - the qty_on_hand for this title is less than or equal to that title's order_point, but it is ALREADY on-order.
    - 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, AND
    - 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 3, Problem 2, whose example solution is available on the course Moodle site if you need it...) and pending_order_needed (from Homework 4, Problem 2, whose example solution is likewise available) in determining this;
- only if it IS needed, then, should sell_book call insert_order_needed (from Homework 4, Problem 1, whose example solution is likewise available) appropriately to make an entry into the order_needed table.
  - Use the ISBN from the current sale-in-progress;
  - use the auto_order_qty from the title table for this ISBN as the value of the order_qty attribute of the order_needed table.
- and, finally, (if no exceptions have been raised), return a results code of 0.
- If **no** exceptions are raised, commit these changes, and return a results code of 0, which lets the caller know that the book sale transaction succeeded.

Separately (outside of 328hw5.sql) run the posted testing script sell_book_test.sql posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, sell_book_test_out.txt, along with your files for this homework.

You may also add additional testing calls along with your sell_book code in your 328hw5.sql script if you would like.

Make sure that you submit your 328hw5.sql and 328hw5-out.txt files as well as the sell_book_test_out.txt file.

Problem 3

Consider a rather typical plain HTML5 form, in 328hw05-before.html, adapted from:
http://srv13.fountainheadcollege.com/mustafa.eminoglu/ws201/registration.html

Copy this example into a file 328hw05-after.html and modify it as you like with an external style sheet named 328hw05-after.css (whose link element should **follow** that for normalize.css). Meet the following specifications:

- Include comment(s) including at least your name and the last-modified date in **BOTH** your `328hw05-after.html` and `328hw05-after.css` .
- Add your name within the `body` element of `328hw05-after.html` in some **visible** fashion of your choice.
- Use CSS3 to tastefully, attractively, and readably format and layout this document, including its form.
  - Remember that your course textbook describes many interesting properties in Chapters 3 and 4.
  - You may add appropriate attributes to elements in `328hw05-after.html`.
  - You **may** remove some of the `<br />` elements currently within this form -- try to avoid adding any additional `<br />` elements.
  - Do **not** remove any of the existing form controls; do **not** change any of existing visible text.
  - 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.
  - At least one rule should use the `float` property in a visible and attractive way.
  - At least one rule should set at least one of an element's `padding` values in a visible and attractive way.
  - At least one rule should set at least one of an element's `margin` values in a visible and attractive way.
  - At least one rule should add an attractive **border** to an element.
- When you are satisfied with them (**AFTER** debugging and validating!!),

  submit your resulting `328hw05-after.html` and `328hw05-after.css` files,

  and **then "HIDE" them** by changing their permissions to make them **NOT** world-readable:

```
chmod 600 328hw05-after.html
chmod 600 328hw05-after.css
```

Problem 4

Consider Homework 4 - Problem 5's `hw4-play.html`. Surely it needs some style!

Make a **new** copy of your `hw4-play.html` named `hw5-play.html`.

Then, create an external style sheet `hw5-play.css` and modify `hw5-play.html` so that it uses both `normalize.css` and this external style sheet.

- In `hw5-play.html`, ALSO modify the URL from which it can be run included within a comment near its beginning. **It will be an automatic -10 penalty on this homework if this URL does not display this page when I/the grader attempt to use it in a browser.**

Your `hw5-play.css` should meet the following specifications in:

- Include a comment including at least your name and the last-modified date in `hw5-play.css` .
- Include at least the following rules in your external style sheet (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.
- At least one rule should use the `float` property in a visible and attractive way.
- At least one rule should set at least one of an element's `padding` values in a visible and attractive way.
- At least one rule should set at least one of an element's `margin` values in a visible and attractive way.
- At least one rule should add an attractive **border** to an element.
- Include rules to "nicely" (tastefully, attractively, and readably) lay out and format `hw5-play.html`'s table AND form, of course also making any needed changes to `hw5-play.html` along the way.
- Remember that your course textbook describes many interesting properties in Chapters 3 and 4.

Submit your files `hw5-play.html` and `hw5-play.css`.

Problem 5

Consider your HTML5 documents `bks-start.html` and `insert-ord-needed.html` from Homework 4.

Make a **new** copy of these in a **different** directory, since you will be modifying them and you don't want to change Homework 3's or Homework 4's version of them (since that could affect your Homework 3 or Homework 4 grade!)

We'll not be including a link to either of these in your `index.html` -- instead, make sure both new versions include the URL from which they can be run within a comment near their beginning. **It will be an automatic -10 penalty on this homework if this URL does not display this page when I/the grader attempt to use it in a browser.**

- 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` .
  - Modify these new versions of `bks-start.html` and `insert-ord-needed.html` to use both `normalize.css` and this new style sheet `bks.css`.
  - Include a comment including at least your name and the last-modified date in `bks.css` .
  - Include at least **five** distinct rules in `bks.css` that have a visible effect on these pages (but you can certainly add more if you are inspired!).
  - Include rules that "nicely" (tastefully, attractively, and readably) lay out and format `bks-start.html`'s login form, of course also making any needed changes to `bks-start.html` along the way.
  - Include rules that "nicely" (tastefully, attractively, and readably) lay out and format `insert-ord-needed.html`'s form, of course also making any needed changes to `insert-ord-needed.html` along the way.
  - Feel free to also add rules for additional formatting to these documents as desired (especially formatting making use of the CSS box model, but you are not limited to that!).

Submit your files `bks-start.html`, `insert-ord-needed.html`, and `bks.css`.

Problem 6

Consider your HTML5 documents `custom-start.html` (from Homework 3) and `custom-plsql.html` (from Homework 4) that you created for use with "your" database.

Make a **new** copy of these in a **different** directory, since you will be modifying them and you don't want to change Homework 3's or Homework 4's version of them (since that could affect your Homework 3 or Homework 4 grade!)

Make sure both new versions include the URL from which they can be run within a comment near their beginning. **It will be an automatic -10 penalty on this homework if this URL does not display this page when I/the grader attempt to use it in a browser.**

- Design a first version of an external style sheet `custom.css` that you would like to use for these and other pages you create making use of "your" database.
  - Modify these new versions of `custom-start.html` and `custom-plsql.html` to use both `normalize.css` and this new style sheet `custom.css`.
  - Include a comment including at least your name and the last-modified date in `custom.css` .
  - Include at least **five** distinct rules in `custom.css` that have a visible effect on these pages (but you can certainly add more if you are inspired!).
  - Include rules that "nicely" (tastefully, attractively, and readably) lay out and format `custom-start.html`'s login form, of course also making any needed changes to `custom-start.html` along the way.
  - Include rules that "nicely" (tastefully, attractively, and readably) lay out and format `custom-plsql.html`'s form, of course also making any needed changes to `custom-plsql.html` along the way.
  - Feel free to also add rules for additional formatting to these documents as desired (especially formatting making use of the CSS box model, but you are not limited to that!).

Submit your files `custom-start.html`, `custom-plsql.html`, and `custom.css`.