CS 328 - Homework 6

Deadline

11:59 pm on **Friday, March 14, 2025**

Purpose

To look over the class CSS style standards, to practice writing another PL/SQL trigger, to think about color and accessibility, and to practice more with CSS external style sheets.

How to submit

You complete **Problem 1** on the course Canvas site (short-answer questions on CS 328 CSS coding standards), so that you can see if you are on the right track.

Each time you wish to submit your work-so-far for **Problems 2 onward**, submit your files using `~st10/328submit` on nrs-projects, with a homework number of `6`.

Important note: It is quite likely that your PL/SQL files will be in a different directory than your HTML files. That's fine, and preferable!

- Just remember that you need to run `~st10/328submit` from **EACH** directory with files to be submitted for Homework 6.

Important notes

- Remember: You are required to use `normalize.css` for all of your web pages for CS 328, and to add `link` elements for additional CSS external stylesheets *after* this (but still within the `head` element).

  EXCEPT for `normalize.css`, **DO NOT USE ANY CSS FRAMEWORKS or PREDEFINED LIBRARIES for this course** (unless you get prior, explicit permission). One of this course's purposes is to provide you with some practice with the basics of "plain" CSS, so you can better make use of frameworks and predefined libraries later.
- Remember that there are **CS 328 CSS Coding Standards so far** posted on the public course web site, under **References** -- you are also expected to follow these for all course documents.
  - External style sheets are expected to validate as valid CSS using the validator link included in the **CS 328 CSS Coding Standards so far**.
- You are expected to follow all course coding standards posted on the public course web site; course documents are also expected to validate as strict-style HTML.

Problem 1 - 14 points

Problem 1 is correctly answering the "HW 6 - Problem 1 - Short-answer questions on CS 328 CSS coding standards" on the course Canvas site.

Problem 2

Imagine that **`set-up-ex-tbls.sql`**'s tables' scenario has the following two **business rules**:

- Once an employee is hired, they are assigned an **`empl_num`** that is **not** allowed to be changed.
- An employee's salary cannot be changed more than once in a day.

2 part a - get the table `salary-history`

This scenario now wants to keep track of how employees' salaries change over time.

The SQL script **`init-sal-history.sql`** creates a table **`salary_history`**:

```
create table salary_history
(empl_num     char(4),
 prev_salary number(6, 2),
 next_salary number(6, 2),
 change_date date,
 primary key (empl_num, change_date),
 foreign key (empl_num) references empl);
```

...and inserts rows for the initial salaries of all of the current employees from `set-up-ex-tbls.sql`.

(When someone is hired, they should have a row in **`salary_history`** with their **`empl_num`**, *no* **`prev_salary`** , their initial salary as the **`next_salary`**, and their initial hiredate as the **`change_date`**).

Assuming you are in your desired work directory on nrs-projects for this problem, you can get a copy of this script by using the Linux command:

```
cp ~st10/init-sal-history.sql .  # DON'T forget the space and dot!
```

Run this script, and confirm that you now have this table, with 14 initial rows for the 14 usual employees.

2 part b - start setting up your SQL script

Create a file **`328hw6.sql`**. Give this file permissions of **`600`** (`rw-------`) by typing this at the nrs-projects prompt:

```
chmod 600 328hw6.sql
```

Start this file with the following:

- comments containing at least your name, **`CS 328 - Homework 6`**, and the last-modified date.
- include the command to **`set serveroutput on`**
  - (although, for this homework's trigger, you may not need it unless you use **`dbms_output.put_line`** for debugging purposes)
- followed by a SQL*Plus **`spool`** command to spool the results of running this SQL script to a file named **`328hw6-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 creating and testing your trigger).

2 part c - create trigger `log_salary`

Now create and compile a PL/SQL trigger **`log_salary`**:

- This trigger should fire **after** each **`insert`** or **`update`** to the **`empl`** table, for **each** row inserted or updated.
- **After** each **`insert`** into **`empl`**:
  - It inserts a row into **`salary_history`** with the new employee's employee number, a previous salary of null, a next salary of the initial salary, and a change date of their hiredate.
- **After** each **`update`** into **`empl`**:
  - **IF** that **`update`** changed the employee's salary, it inserts a row into **`salary_history`** with the updated employee's employee number, a previous salary of their pre-update salary, a next salary of their post-update salary, and a change date of the current date (use **`sysdate`** for this).
  - (**But** if the **`update`** did **not** change the employee's salary, this trigger should do **nothing** -- it should ***not*** add a row to **`salary_history`** in that case.)

Here are additional requirements for this problem:

- Create an opening comment block for your procedure that has a **`trigger:`** part and **`purpose:`** part in the same style as used in posted SQL scripts from Week 6 Lecture 1 **`trigger1.sql`** through **`trigger4.sql`** for the different versions of the trigger **`empl_trig`**.
  - Follow that with the PL/SQL code creating your trigger.
- Remember to follow your PL/SQL trigger with:

  `/`

  `show errors`

2 part d - test trigger `log_salary`

- Then put a **comment** saying you are about to **test** your trigger **`log_salary`**,

  followed by a **`prompt`** command saying you are about to test your trigger **`log_salary`**.
- Put a **`commit;`** to commit the pre-testing version of your database.
- Put a **`prompt`** command that says you are about to see the pre-test versions of **`empl`** and **`salary_history`**,

  followed by two **`select`** statements projecting the current contents of the **`empl`** and **`salary_history`** tables.
- Set up a test to see if your trigger works for a **newly-inserted** employee:
  - Put a **`prompt`** command saying you are about to insert a new employee (specifying at least their name),

    followed by an **`insert`** statement inserting that new employee, specifically giving them a

`hiredate` EARLIER than today.

- Put a **`prompt`** command saying you should now see **15** employees, including the new one with their last name specified,

  followed by a **`select`** statement projecting the contents of the **`empl`** table.

- Put a **`prompt`** command saying you should now see **15** rows in **`salary_history`**, with the last being for the new employee (with their employee number, initial salary, and hiredate specified, to make it easier to check for those in the new row),

  followed by a **`select`** statement projecting the contents of the **`salary_history`** table.

- Set up tests to see if your trigger works for an **updated** employee:
  - For an employee ***other*** than your newly-inserted employee,

    put a **`prompt`** command saying you are about to update their salary (specifying at least their name and their new salary),

    followed by an **`update`** statement doing that update.

  - For a *different*, second employee *other* than your newly-inserted employee,

    put a **`prompt`** command saying you are about to update something ELSE for them (**other** than their `empl_num`, `salary`, or `hiredate`) (specifying at least their name and updated attribute's new value),

    followed by an **`update`** statement doing that update.

  - Put a **`prompt`** command describing what changes in **`empl`** should now be checked for,

    followed by a **`select`** statement projecting the new contents of the **`empl`** table.

  - Put a **`prompt`** command saying you should now see **just 16** rows in **`salary_history`**, with the last being for the employee whose salary was updated (with their employee number, initial salary, updated salary, and a note that the change date should be today's date specified, to make it easier to check for those in the new row),

    followed by a **`select`** statement projecting the contents of the **`salary_history`** table.

- THEN put a **`rollback;`** to undo these changes that were made just for testing purposes.

Make sure you have a **`spool off`** command at the end of **`328hw6.sql`**, and submit your files **`328hw6.sql`** and **`328hw6-out.txt`**.

Problem 3

Problem 3 background:

We have discussed how you can use CSS to specify the color and background color of HTML elements. Also, we have said that accessibility is an important goal for HTML documents.

If a person has color vision deficiency, or is trying to read something in bright conditions, or is using a screen happening to be lacking in terms of size, resolution, and/or contrast, an HTML document being displayed with too-little color contrast between text and background may be difficult, or even impossible, to read. (reference: https://accessibility.blog.gov.uk/2016/06/17/colour-contrast-why-does-

it-matter/)

In addition to the Color Picker in Section 3.4 of the zyBooks course text, there are many color tools on the Web -- for example:

- https://www.sessions.edu/color-calculator/
- https://www.w3schools.com/colors/colors_hsl.asp

...and there are numerous color contrast tools as well. Here is one example of a tool that lets you specify a foreground and background color, and then helps you determine if they provide enough contrast:

- https://snook.ca/technical/colour_contrast/colour.html

Such tools can be useful for testing a foreground and background color you are interested in using, to see if they provide sufficient contrast. But it can be tough to use this to determine pairs of such colors to begin with.

Happily, this document from the Web Content Accessibility Guidelines (WCAG):

- http://web-accessibility.carnegiemuseums.org/design/color/

...includes a nicely-varied selection of "color palette combinations that fall within the range of Section 508 compliant foreground/background color contrast ratios".

(Section 508 is, as noted at:

https://www.epa.gov/accessibility/learn-about-section-508-and-digital-accessibility

...part of the U.S. Rehabilitation Act and "requires federal agencies to make their ICT such as technology, online training, and websites accessible for everyone.")

your task for Problem 3:

- Make a file **`328hw6-color.txt`**, and start this file with your **name** and the **last modified date**.
- Determine a foreground color and background color pair -- **at least one** of which is **not** black (`#ffffff`) or white (`#000000`) -- you are interested in using later in this homework that is **at least WCAG 2 AA Compliant** according to the tester at https://snook.ca/technical/colour_contrast/colour.html
  - (That is, make sure it shows **YES** in the **fifth** textfield in the **Results** section for your chosen pair of colors.)
- Demonstrate that you did so by turning in the following **for at least one WCAG 2 AA Compliant** pair of colors:
  - include your **foreground** color choice, given in hexadecimal format (e.g., `#046b99`)
  - include your **background** color choice, given in hexadecimal format
  - include the **brightness difference** for these (from the **first** textfield in the **Results** section)
  - the **color difference** for these (from the **second** textfield in the **Results** section), and
  - the **contrast ratio** for these (from the **fourth** textfield in the **Results** section).

  ... in https://snook.ca/technical/colour_contrast/colour.html.

Submit your file **`328hw6-color.txt`**.

Problem 4

Consider your HTML document **`about-bks.html`** from Homework 2 - Problem 3, describing your chosen theme for your instance of the bookstore database.

Make a **COPY** of your **`about-bks.html`** in a **DIFFERENT** directory on nrs-projects (so you will not interfere with your earlier homework's files!).

Design an initial version of an external style sheet **`bks.css`** for this homework's copy of **`about-bks.html`** (and hopefully/possibly for other future bookstore-related documents). Modify this homework's copy of **`about-bks.html`** to use **both** **`normalize.css`** and this new style sheet **`bks.css`**.

Requirements for this initial version of **`bks.css`**:

- Include a comment including at least your **name** and the last-modified date.
- Include at least **five** distinct rules in **`bks.css`** that have a visible effect (but you can certainly add more if you are inspired!).
- Have at least **one** rule visibly specifying the **`color`** or **`background-color`** property of some element(s).
  - Make sure the text/background contrast for your styled document is **at least WCAG 2 AA Compliant** based on the tester at https://snook.ca/technical/colour_contrast/colour.html.
- As discussed in the zyBooks course text in Chapter 3, Section 3.5, "Font and text properties", there are **five font-related properties** in addition to the shortcut property **`font`**.
  - For **this** problem, for experimenting purposes, **do not use** the shortcut property **`font`** -- instead, use **at least three** of the **non-shortcut** font-related properties amongst your rules, such that the results are clearly visible in your styled document.
  - Note: remember the follow the course style standards with regard to specifying sizes!
- Feel free to also add rules for additional formatting as desired.

Make sure your resulting documents successfully validate as strict-style HTML and valid CSS.

Submit this homework's modified copies of **`about-bks.html`** and **`about-bks.xhtml`** along with your **`bks.css`**.