CS 328 - Homework 10

Deadline

11:59 pm on **Friday, April 24, 2026**

Purpose

To get more practice writing PHP postback documents that send requests to Oracle using OCI, now also including:

- using **OCI bind variables** to more-safely create dynamic **SQL `select`** statements
- using OCI to request calls of **PL/SQL stored subroutines**

How to submit

Each time you wish to submit your work-so-far, submit your files using **`~st10/328submit`** on nrs-projects, with a homework number of **`10`**.

Important notes

- **NOTE**: you are **welcome** to use **`require_once`**/**`include_once`**/**`require`**/**`include`** in your PHP documents!
  - BUT when you use them in homework problems' documents, be sure to also **SUBMIT** copies of all files that you are requiring/including!
- **DO NOT USE ANY PHP FRAMEWORKS FOR THESE PROBLEMS** -- one of this course's purposes is to provide you with practice with "plain" PHP.
- 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: there are now **CS 328 PHP Coding Standards so far** posted on the public course web site, under References -- you are also expected to follow these for all course PHP documents.
- You are expected to follow all course coding standards posted on the public course web site; course documents (including documents generated by your PHP files) are also expected to validate as "strict"-style HTML, and valid CSS.
- 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.

  And when you re-run these, you should also re-run:

- **Homework 3**'s **`new-titles.sql`**, to re-add your additional titles
- **Homework 5**'s **`328hw5.sql`**, to re-create its stored subroutines
- **Homework 6**'s **`328hw6.sql`**, to re-create its stored function.

Problem 1 - practice using OCI bind variables to more-safely create dynamic SQL `select` statements

Purpose: to get more practice using OCI bind variables to help thwart SQL injection.

Task: Create a modified version of a previous PHP postback document that now responds to its form using a dynamic **`select`** statement that uses **bind variables** rather than concatenation (to help thwart SQL injection).

In **Homework 9, Problem 4**, in the file **`328hw9-4.php`**, you created a form with a ***dynamically-generated*** **`select`**/drop-down box form element that allows a user to elect a title's ISBN from the current titles in your bookstore database.

For this problem, decide on **at least two** pieces of information a user might like to ask about a title available at your bookstore -- as just a *few* of the possible examples:

- What is the price and currently quantity of a particular title?
- Who is the author and publisher of a particular title?
- Who is the author, and what is the price, of a particular title?

Make a copy of the connection-helper-function **`hum_conn_no_login`** in your directory for this problem:

```
cp ~st10/hum_conn_no_login.php .  # DON'T FORGET the SPACE & PERIOD!!
```

Also, make a **COPY** of your file **`bks.css`** from **Homework 9 - Problem 4** in this problem's directory on nrs-projects (so you will not interfere with your earlier homework's files!).

Then, create a PHP postback document **`328hw10-1.php`** that meets the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** its **`head`** element, edit the **`title`** element, giving it appropriate content.
- Within its **`head`** element: include a regular PHP tag or tags:
  - to enable PHP error reporting
  - to include the definition of function **`hum_conn_no_login`**, from your local copy of file **`hum_conn_no_login.php`**
- **Within** its **`head`** element, add a second **`link`** element so that this will be further styled using **Homework 10's** version of **`bks.css`**.
  - **Do not include any inline or internal CSS rules in your `328hw10-1.php`.**
- Within its **`body`** element, include an **`h1`** element with appropriate content that somehow includes

"your" bookstore's name (from previous homeworks' **`about-bks.html`**).

- Within its **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- Your **`328hw10-1.php`**'s logic should be designed so that its response includes **either** a form, **or** a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw10-1.php`** can be the **SAME** form as that generated by your **`328hw9-4.php`** -- that is, it meets the following requirements:
  - its **`action`** attribute should have as its action:

    `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`
  - instead of being hard-coded, the **`option`** elements of ISBN-title pairs in its **`select`**/drop-down **must** be **dynamically built** based on **querying** for the current ISBNs and titles of books in the bookstore databases' **`title`** table.
  - AS in Homework 4 - Problem 4:
    - Set up this **`select`**'s **`option`** elements so that the user **sees**, in the **`select`**/drop-down box, ISBN-title name pairs, **but**...
    - ...when a particular option is selected, the **value** in the resulting name=value pair for this option is **JUST** the ISBN for the user's choice.
    - (for example, if the user selects an option that is displayed as:

      `9780131103627 - The C Programming Language`

      ... the form will submit a name=value pair whose value is just `9780131103627`)
    - **ASK ME** if you are not sure what I am asking for here.
- When this form is submitted, the response generated by your **`328hw10-1.php`** should include the following:
  - Appropriately **sanitize** the information submitted by this form.
  - Build a dynamic **SQL `select`** statement that projects **at least two attributes**, and uses at least one **bind variable** (instead of concatenation!) in its **`where`** clause based on the user's sanitized selection from the form's **`select`**/drop-down widget.
    - Note that you will lose **substantial** credit if you use concatenation to include any submitted information within your **`select`** statement string -- you are **required** to use a **bind variable** instead!
  - Add the **SQL `select`**'s results to the response in a pleasing, strict-HTML-style way.
  - The response should **ALSO** include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw10-1.php`**.

Your Homework 10 version of **`bks.css`** should meet the following requirements:

- Actually, it is fine if you decide that you do not need to make any changes to this, as long as it meets Homework 7 - Problem 2's requirements.

- But, whether it is changed or not, make sure the resulting **bks.css** still validates as valid CSS.

Strict-validate **BOTH** parts generated by your **328hw10-1.php** as you did for the Week 9 Lab Exercise's 328lab09.php:

- Put your **328hw10-1.php**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **328hw10-1-1.xhtml**, and put the URL of your **328hw10-1-1.xhtml** into the validator.
- Put your **328hw10-1.php**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **328hw10-1-2.xhtml**, and put the URL of your **328hw10-1-2.xhtml** into the validator.

Submit your resulting files:

- **328hw10-1.php** (and all additional files it uses, if any)
- **328hw10-1-1.xhtml** and **328hw10-1-2.xhtml**
- **bks.css**

Problem 2 - calling a PL/SQL stored subroutine using OCI

Purpose: to practice calling a PL/SQL stored subroutine using OCI from PHP

Consider: On **Homework 5 - Problem 6**, you wrote a PL/SQL stored function **get_pub**, that expects a title's ISBN and returns the **name** of its publisher (**not** its publisher ID).

On **Homework 6 - Problem 2**, you wrote a PL/SQL stored function **get_order_date**, that expects an **order_stock**'s order number and returns a string version of the date that order was placed, in the format '<3-letter-day of the week>, <3-letter-month> <two-digit day of the month>, <4-digit year>'.

Choose which of these you would like to call for this problem's PHP postback document **328hw10-2.php**.

Task: Create a PHP postback document **328hw10-2.php** that meets the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** its **head** element, edit the **title** element, giving it appropriate content.
- Within its **head** element: include a regular PHP tag or tags:
  - to enable PHP error reporting
  - to include the definition of function **hum_conn_no_login**, from your local copy of file **hum_conn_no_login.php**
- **Within** its **head** element, add a second **link** element so that this will be further styled using **Homework 10's** version of **bks.css**.
  - **Do not include any inline or internal CSS rules in your 328hw10-2.php**.
- Within its **body** element, include an **h1** element with appropriate content that somehow includes

"your" bookstore's name (from previous homeworks' **`about-bks.html`**).

- Within its **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- Your **`328hw10-2.php`**'s logic should be designed so that its response includes **either** a form, **or** a response to its form -- its response should never include both.
- **IF** you selected function **`get_pub`** for this problem, the initial form element generated by your **`328hw10-2.php`** can be the SAME as that generated by **`328hw10-1.php`**.

  IF you selected function **`get_order_date`** for this problem, the initial form generated by **`328hw10-2.php`** should include a **`select`**/drop-down **dynamically built** based on **querying** for the current order numbers in the bookstore databases' **`order_stock`**'s table.

  In **EITHER** case, your initial **`form`** should meet the following requirements:

  - its **`action`** attribute should have as its action:

    `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`
  - instead of being hard-coded, the **`option`** elements in its **`select`**/drop-down **must** be **dynamically built** based on **querying** the appropriate current information from the appropriate table in the bookstore database.
  - include an **`input`** element with **`type="submit"`** (which does *not* need a logically-related **`label`** element)
- When this form is submitted, the response generated by your **`328hw10-2.php`** should include the following:
  - Appropriately **sanitize** the information submitted by this form.
  - Use the sanitized information as the argument to your chosen PL/SQL stored function (**`get_pub`** or **`get_order_date`**)
    - Note that you will lose **substantial** credit if you use concatenation to include any submitted information within your PL/SQL function call string -- you are **required** to use a bind variable instead!
  - Output a **`p`** element with content including **both** the sanitized information from the submitted form AND the value returned by your chosen PL/SQL stored function (**`get_pub`** or **`get_order_date`**)
  - For more-convenient re-use, include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw10-2.php`**.

Your Homework 10 version of **`bks.css`** should meet the following requirements:

- Actually, it is fine if you decide that you do not need to make any changes to this, as long as it meets Homework 7 - Problem 2's requirements.
- But, whether it is changed or not, make sure the resulting **`bks.css`** still validates as valid CSS.

Strict-validate **BOTH** parts generated by your **`328hw10-2.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw10-2.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw10-2-1.xhtml`**, and put the URL of your **`328hw10-2-1.xhtml`** into the validator.
- Put your **`328hw10-2.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw10-2-2.xhtml`**, and put the URL of your **`328hw10-2-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw10-2.php`** (and all additional files it uses, if any)
- **`328hw10-2-1.xhtml`** and **`328hw10-2-2.xhtml`**
- **`bks.css`**

Problem 3 - building dynamic form widget(s) involving your "second" database

Purpose: to get more practice dynamically generating form widget and using OCI bind variables to help thwart SQL injection, in this case making use of your "second" database.

Consider: In **Homework 4 - Problem 5**, in the file **`custom-choice.html`**, you included a **`form`** element that included form widget(s) for selecting something that could be used to answer a question from an expected user of your "second" database, that could be answered by a SQL `select` statement that has a `where` clause specifying something entered by the user.

And, in **Homework 7, Problem 3**, you created an external CSS stylesheet **`custom.css`** to style **`custom-choice.html`**.

Task: Create a PHP postback document **`328hw10-3.php`** that either:

- creates a form that REPLACES the hand-written/hard-coded form widget element/set of elements in **`custom-choice.html`** with ***dynamically-generated*** form widgets (**dynamically** built based on a query's results), or
- crafts a response to that form when it is submitted with the help of a dynamic **`select`** statement that uses **bind variables** rather than concatenation (to help thwart SQL injection).

Make a **COPY** of your file **`custom.css`** from **Homework 7 - Problem 3** in this problem's directory on nrs-projects (so you will not interfere with your earlier homework's files!).

Your PHP postback document **`328hw10-3.php`** should meet the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** its **`head`** element, edit the **`title`** element, giving it appropriate content.
- Within its **`head`** element: include a regular PHP tag or tags:
  - to enable PHP error reporting
  - to include the definition of function **`hum_conn_no_login`**, from your local copy of file

`hum_conn_no_login.php`

- **Within** its **`head`** element, add a second **`link`** element so that this will be further styled using Homework 10's version of **`custom.css`**.
  - **Do not include any inline or internal CSS rules in your `328hw10-3.php`.**
- Within its **`body`** element, include an **`h1`** element with appropriate content..
- Within its **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- Your **`328hw10-3.php`**'s logic should be designed so that its response includes **either** a form, **or** a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw10-3.php`** should be a variation of your **`form`** from Homework 7 - Problem 3's **`custom-choice.html`**, meeting the following requirements:
  - its **`action`** attribute should have as its action:

    `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`
  - instead of being hard-coded, your form widget element/set of elements for the user's choice **must** be **dynamically built** based on **querying** for the current choices from your "second" database.
  - include an **`input`** element with **`type="submit"`** (which does *not* need a logically-related **`label`** element)
- When this form is submitted, the response generated by your **`328hw10-3.php`** should include the following:
  - Appropriately **sanitize** each piece of information submitted by this form.
  - Build a dynamic **`select`** statement that projects something(s) that can be used to answer a question such as that you imagined back in Homework 4 - Problem 5, and uses at least one **bind variable** (instead of concatenation!) in its **`where`** clause based on the user's selection from the form's dynamically-built widget/set of widgets.
    - Note that you will lose **substantial** credit if you use concatenation to include any submitted information within your **`select`** statement string -- you are **required** to use bind variables instead!
  - Add the **`select`**'s results to the response in a pleasing, strict-HTML-style way.
  - The response should **ALSO** include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw10-3.php`**.

Your Homework 10 version of **`custom.css`** should meet the following requirements:

- Actually, it is fine if you decide that you do not need to make any changes to this, as long as it meets Homework 7 - Problem 3's requirements.
- But, whether it is changed or not, make sure the resulting **`custom.css`** still validates as valid CSS.

Strict-validate the two parts generated by your **`328hw10-3.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw10-3.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw10-3-1.xhtml`**, and put the URL of your **`328hw10-3-1.xhtml`** into the validator.
- Put your **`328hw10-3.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw10-3-2.xhtml`**, and put the URL of your **`328hw10-3-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw10-3.php`** (and all additional files it uses, if any)
- **`328hw10-3-1.xhtml`** and **`328hw10-3-2.xhtml`**
- **`custom.css`**
- (**IF** you make any changes to your "second" database, make sure you also submit your current/latest version of its files.)

Problem 4 - calling a PL/SQL stored subroutine for your "second" database

Purpose: to practice calling another PL/SQL stored subroutine using OCI from PHP, in this case making use of your "second" database.

Consider: On **Homework 8 - Problem 4**, you wrote a PL/SQL stored function or PL/SQL stored procedure for your "second" database, in **`*second-db*.sql`**.

Task: Create a PHP postback document **`328hw10-4.php`** that either:

- creates an appropriate **`form`** allowing the user to enter appropriate argument(s) for your "second" database's stored subroutine, or
- crafts an appropriate response after calling your "second" database's stored subroutine with the (appropriately sanitized) user data from that submitted form.

Your PHP postback document **`328hw10-4.php`** should meet the following requirements:

- Include your name and last modified date in its opening comment, **AND** the **URL** this can be run from.
  - (You *will* lose some credit if this URL does not work when I or the grader paste it into a browser!)
- **Within** its **`head`** element, edit the **`title`** element, giving it appropriate content.
- **Within** its **`head`** element: include a regular PHP tag or tags:
  - to enable PHP error reporting
  - to include the definition of function **`hum_conn_no_login`**, from a local copy of file **`hum_conn_no_login.php`**
- **Within** its **`head`** element, add a second **`link`** element so that this will be further styled using Homework 10's version of **`custom.css`**.
  - **Do not include any inline or internal CSS rules in your `328hw10-4.php`.**

- Within its **`body`** element, include an **`h1`** element with appropriate content.
- Within its **`footer`** element near the end of the **`body`** element, add a **`p`** element whose content includes your name.
- Your **`328hw10-4.php`**'s logic should be designed so that its response includes **either** a form, **or** a response to its form -- its response should never include both.
- The initial **`form`** element generated by your **`328hw10-4.php`** should meet following requirements:
  - its **`action`** attribute should have as its action:

    `"<?= htmlentities($_SERVER["PHP_SELF"], ENT_QUOTES) ?>"`
  - it should include appropriate form widget element(s) (with logically-associated **`label`** element(s)) so that the user can enter/select the needed argument(s) for your "second" database's stored subroutine.
    - ***If*** any of these are based on choosing from an existing table attribute's values, be sure to dynamically build the choices based on querying the current values from your database rather than hard-coding them.
  - include an **`input`** element with **`type="submit"`** (which does *not* need a logically-related **`label`** element)
- When this form is submitted, the response generated by your **`328hw10-4.php`** should include the following:
  - Appropriately **sanitize** the information submitted by this form.
  - Appropriately use the sanitized submission(s) to determine the argument(s) to your "second" database's PL/SQL stored subroutine.
    - Note that you will lose **substantial** credit if you use concatenation to include any submitted information within your PL/SQL subroutine call string -- you are **required** to use bind variable(s) instead!
  - What should you output in this response? It depends on the nature of your "second" database's PL/SQL stored subroutine:
    - If it is a stored function that returns a desired value, include its results in the response in a pleasing, strict-HTML-style way.
    - If it is a stored procedure (or a stored function that returns a status code) that performs some desired task, note what it did in a pleasing, strict-HTML-style way.
  - ***If*** your "second" databases' PL/SQL stored subroutine changed the database, be sure to call **`oci_commit`** when your logical transaction is complete to commit those changes!
  - For more-convenient re-use, include an **`a`**/anchor element (hypertext link) with appropriate text that links back to your **`328hw10-4.php`**.

Your Homework 10 version of **`custom.css`** should meet the following requirements:

- Actually, it is fine if you decide that you do not need to make any changes to this, as long as it meets

Homework 7 - Problem 3's requirements.

- But, whether it is changed or not, make sure the resulting **`custom.css`** still validates as valid CSS.

Strict-validate the two parts generated by your **`328hw10-4.php`** as you did for the Week 9 Lab Exercise's `328lab09.php`:

- Put your **`328hw10-4.php`**'s URL in a browser and **view its source**, **copy and paste** that **source** into a file named **`328hw10-4-1.xhtml`**, and put the URL of your **`328hw10-4-1.xhtml`** into the validator.
- Put your **`328hw10-4.php`**'s URL in a browser **and fill out and submit its form**, *then* **view that *response*'s source**, and **copy and paste** that ***response*'s source** into a file named **`328hw10-4-2.xhtml`**, and put the URL of your **`328hw10-4-2.xhtml`** into the validator.

Submit your resulting files:

- **`328hw10-4.php`** (and all additional files it uses, if any)
- **`328hw10-4-1.xhtml`** and **`328hw10-4-2.xhtml`**
- **`custom.css`**
- your current/latest version of **`*second-db*.sql`**
- (**IF** you make any changes to your "second" database, make sure you also submit your current/latest version of its files.)