

CS 328 - Homework 2

Deadline

Due by 11:59 pm on **Friday, February 5, 2016**.

How to submit

Submit your files for this homework using `~st10/328submit` on nrs-projects, with a homework number of 2.

Reminder: Instructions for using the tool ~st10/328submit

- If they are not already on nrs-projects, transfer your files to be submitted to a directory on nrs-projects.
 - You can do so by using `sftp` (secure file transfer) and connecting to `nrs-projects.humboldt.edu`, and then transferring them
- Once all of your files to be submitted are in a directory on nrs-projects, then use `ssh` to connect to `nrs-projects.humboldt.edu`.
- use `cd` to change to the directory containing the files to be submitted -- for example,

```
cd 328hw02
```
- type the command:

```
~st10/328submit
```

...and give the number of the homework being submitted (or whatever number you have been asked to do for lab-related files) when asked, and answer that, `y`, you do want to submit all of the files with of-interest-to-328 suffixes in the current directory. (Note that I don't mind if a few extraneous files get submitted as well -- I'd rather receive too many files than too few, and typing in all of the file names for each assignment is just too error-prone...)
- you are expected to carefully check the list of files that the tool believes have been submitted, and make sure all of the files you hoped to submit were indeed submitted! (The most common error is to try to run `~st10/328submit` while in a different directory than where your files are.)

Purpose

To practice with PL/SQL stored procedures and functions.

Initial set of CS 328 PL/SQL and SQL coding standards

Here is an opening/beginning set of CS 328 SQL and PL/SQL Style Standards -- your SQL and PL/SQL code is expected to conform to these standards:

- THOU SHALT precede each PL/SQL subroutine with an opening comment block that includes at least:
 - the subroutine's name

- a purpose statement which explicitly describes what the subroutine expects (for a trigger, describe when it is fired), and what it does and/or returns
- THOU SHALT put a blank line before and after each `SELECT` statement, before and after each PL/SQL subroutine, and usually before each multi-line SQL statement, for better readability. And logically group SQL*Plus statements within a script as well.
- THOU SHALT write a `SELECT` statement's `FROM` clause on its OWN line.
- THOU SHALT write the beginning of a `SELECT` statement's `WHERE` clause on its OWN line.
- THOU SHALT indent nested selects by at least 4 spaces.
- When a `SELECT` statement has N tables/relations in its `FROM` clause (OR for joins written using the `INNER JOIN` syntax), THOU SHALT have (N-1) join conditions (unless one REALLY, TRULY wants a true Cartesian product, a rare occurrence...).
- THOU SHALT use mnemonic table aliases (d and e for tables dept and empl, for example, not x and y...).
- THOU SHALT only use `GROUP BY` clauses when one has a good reason (usually a computation that you wish done to those groups).
- THOU SHALT only use `ORDER BY` clauses for an outermost `SELECT` (not within any sub-selects), and it shall be indented to make clear that it "belongs" to the outermost `SELECT`.

...and I reserve the option to add to this list over the course of the semester.

Problem 1

Recall that we discussed many lovely and useful string-related Oracle SQL functions in CS 325, including `upper`, that expected a column with a string domain and returned an all-uppercase version of the string in that column for each row projected. Happily, these can be used in PL/SQL outside of `select` statements, also, with string or string-variable arguments.

Create a file `328hw2.sql`. Start this file with the following:

- comments containing at least your name, CS 328 - Homework 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 `328hw2-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.)

Then, design and write a PL/SQL stored procedure `silly_shout`, to try your hand at PL/SQL's loop and if statements. `silly_shout` expects two parameters, a `VARCHAR2` parameter and an integer parameter. If the integer is less than or equal to 0, the procedure should print a message to the screen saying that it cannot show the parameter string that many times; otherwise, it should print an all-uppercase version of that parameter string to the screen that many times, once per line, each time concatenating two exclamation point character ('!!') to the end (get it? so it is "shouting" that parameter to the screen? 8-))

Separately (outside of `328hw2.sql`) run the posted testing script `silly_shout_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `silly_shout_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `silly_shout` code in your `328hw2.sql` script if you would like.

Problem 2

In `328hw2.sql`, write a SQL*Plus prompt command that says `problem 2`. (Again, you may add additional prompt commands around this to make it more visible, if you would like -- assume that is the case for the remaining homework problems as well.)

Then, design and write a PL/SQL stored procedure `title_total_cost` that expects one parameter, a title's ISBN, and prints to the screen a message that gives the total **COST** (not price!) of all of the current quantity-on-hand for that title. This message should also include the title's ISBN and its current quantity on hand. (If there is no title with that ISBN, it should print a message saying so, also including that non-existent ISBN in the error message.)

Separately (outside of `328hw2.sql`) run the posted testing script `title_total_cost_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `title_total_cost_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `title_total_cost` code in your `328hw2.sql` script if you would like.

Problem 3

In `328hw2.sql`, write a SQL*Plus prompt command that says `problem 3`.

Then, design and write a PL/SQL stored function `total_on_hand`. `total_on_hand` expects one parameter, the publisher name, and returns the sum of all of the current quantity on hand amongst all of the titles from the publisher with that name. (If there is no publisher with that name, it should just return 0.) For example, the call `total_on_hand('Prentice Hall')` returns 22, the call `total_on_hand('Merrill')` returns 0, and the call `total_on_hand('nonexistent')` returns 0.

Separately (outside of `328hw2.sql`) run the posted testing script `total_on_hand_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `total_on_hand_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `total_on_hand` code in your `328hw2.sql` script if you would like.

Problem 4

In `328hw2.sql`, write a SQL*Plus prompt command that says `problem 4`.

Then, design and write a PL/SQL stored procedure `which_titles` which expects two parameters, which I'll call `quant_limit` and `price_limit`. `which_titles` should print the title name, quantity on hand, and price (separated by hyphens, with a '\$' before the price) of each title whose `qty_on_hand` \geq `quant_limit` and whose price \geq `price_limit`.

For example,

```
SQL> exec which_titles(15, 20)
```

...should result in:

```
Computers and Data Processing-15-$34.95
Data Base Management-20-$37.95
Intro to Biology: A Human-35-$41.95
SPSS-75-$28.95
Management Information Sy-30-$28.95
```

...being printed to the screen.

Separately (outside of 328hw2.sql) run the posted testing script `which_titles_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `which_titles_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `which_titles` code in your 328hw2.sql script if you would like.

Problem 5

IF you do not have a current copy of the `empl` and `dept` tables, you can get a copy of the `set-up-ex-tbls.sql` script either from the Fall 2015 CS 325 public course web site, or by copying it over from my account on nrs-projects:

```
cp ~st10/set-up-ex-tbls.sql . # remember the space and dot at the end!
```

Then, in 328hw2.sql, write a SQL*Plus prompt command that says problem 5.

Write a PL/SQL stored function `get_manager` that expects the name of an employee and returns the name of that employee's manager, BUT with the following special cases:

- IF the name given is not that of an employee, it should return the string 'Not an employee'.
- And, if TWO or more employees have that name, it should return the string 'Name not unique'.
 - THIS has an interesting wrinkle to beware of -- what if two or more employees with the same name happen to have the same manager? After some consideration/waffling, the client decided they still want this case to return the string 'Name not unique'. And a test for this case is included below.
- Finally, if there is only one employee with that name, but that employee has no manager, it should return the string 'No manager' instead of NULL.

You should thoroughly test your resulting function, of course.

Separately (outside of 328hw2.sql) run the posted testing script `get_manager_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `get_manager_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `get_manager` code in your 328hw2.sql script if you would like.

Make sure that you submit your 328hw2.sql and 328hw2-out.txt files as well as the *_test_out.txt files for each of the subroutines in Problems 1-5.