

CS 328 - Week 5 Lab Exercise - 2024-02-16

Deadline

Due by the end of lab.

Purpose

To practice more with PL/SQL, including writing a stored function and a stored procedure, and including practice with parameters, exception-handling, and a cursor-controlled `for` loop.

How to submit

Submit your files for this lab using `~st10/328submit` on nrs-projects, each time entering a lab number of **85**.

Requirements

- You are required to work in **pairs** for this lab exercise.
 - This means **two** people working at **ONE** computer, one typing ("driving"), one saying what to type ("navigating"),
while **BOTH** are looking at the **shared** computer screen and **discussing** issues along the way.
- Make sure **BOTH** of your names appear in each file submitted.
- When you are done, before you leave lab, somehow e-mail or copy the lab exercise files so that **BOTH** of you have copies, and **BOTH** of you should submit these files using `~st10/328submit` on nrs-projects, with a lab number of **85**.
- You are expected to follow the style standards from the posted "CS 328 SQL and PL/SQL Coding Standards so far" (at <https://nrs-projects.humboldt.edu/~st10/s24cs328/328-sql-plsql-coding-standards.pdf>).

Lab set-up

- On nrs-projects, if the driver has not previously executed `set-up-ex-tb1s.sql` in their Oracle account, they should do so, so that they have the tables `empl`, `dept`, and `customer` in their database.
 - If needed, they can get a copy of this script using:

```
cp ~st10/set-up-ex-tb1s.sql .    # don't forget the blank and period!
```
- In a SQL script `lab5.sql`:
 - In opening comment(s), **FIRST** put the script name, both of your names, and today's date/last modified date.
 - Put in the SQL*Plus command:

```
set serveroutput on
```


...so that you will see output from `dbms_output.put_line` statements you put into today's PL/SQL subroutine.

- Start spooling to a file **lab5-out.txt**:
`spool lab5-out.txt`
 ...(and make sure you **spool off** at the script's end!)
- Put both of your names in a **prompt** command.

Problem 1 - stored function **num_pd_more**

To get some practice writing a PL/SQL stored function, in your SQL script `lab5.sql`, write a stored function **num_pd_more** that meets the following requirements:

- It expects a lower-limit salary value.
 - Note: use the type **number** for this parameter.
- It returns the number of employees in the **emp1** table whose salary is strictly greater than that given lower-limit salary value.
- Look in the posted SQL script `328lect05-1.sql` at the version of the stored function **job_count** that we created during class on Monday.
 - Create an opening comment block for your function that has a **function:** part and **purpose:** part in the same style that you see here. (You don't have to give an **examples:** part, but you can if you wish.)
 - Follow that with the PL/SQL code creating your function.
- Remember to follow your PL/SQL function with:
`/`
`show errors`
- Then put a comment saying you are about to **test** your function **num_pd_more**.
- Follow that with at least two tests of your function, written in the same style as the tests for function **job_count**, making sure that, for each test, you put a **prompt** command describing what results should be seen followed by the statements for that test.

Remember:

- You will need to declare a SQL*Plus local variable to hold the result returned by your function.
- The **exec** command is a little **different** when calling a function than when calling a procedure.
- You can use the **print** command to display the value of a SQL*Plus local variable.

If successful, your resulting `lab5-out.txt` should show that your function successfully compiled, and that its tests passed.

Problem 2 - some light PL/SQL exception-handling practice

Consider: in the **emp1** table created in `set-up-ex-tb1s.sql`, the **mgr** attribute is a foreign key referencing **emp1**'s **emp1_num** attribute -- **mgr** is the employee number of that employee's manager.

One attempt at a PL/SQL stored function **get_manager**, that expects an employee's last name and returns the last name of that employee's manager, can be found in `get-mgr-v1.sql`, which you can copy

directly to the driver's nrs-projects current workin directory using:

```
cp ~st10/328lab05/get-mgr-v1.sql . # don't forget the space and period!
```

But, when you run this, it has some definite issues, that you should be able to see when you run this in sqlplus. (The function is followed by some testing calls.)

So: **COPY** this first version (along with its opening comment and its tests) into your lab5.sql file, and then modify it:

- Insert *your names* in the comment that specifies that you do so
- Add **EXCEPTION HANDLING** to the function `get_manager` to handle the tests that currently do not pass, so that they now DO pass.

If successful, your resulting lab5-out.txt should show that your modified version of `get_manager` successfully compiled, and that its tests passed.

Problem 3 - stored procedure `list_managers`

Now that you have `get_manager`, that allows for a good opportunity to practice calling a PL/SQL function from another PL/SQL subroutine, and also allow for a good excuse to practice with a cursor-controlled `for` loop.

In your SQL script lab5.sql, write a PL/SQL procedure `list_managers` that meets the following requirements:

- It expects a job title.
- It prints to the screen, for each employee with that job title:
 - their last name,
 - then a blank and a dash and a blank,
 - then `managed by:` followed by the last name of their manager.
- (And it returns nothing, since it is a procedure!)
- If there are no employees with that job title, it should simply print a message to the screen that includes the nonexistent job title, and notes that there are no employees with that job title.
- It should make appropriate use of a cursor-controlled `for` loop.
- It should appropriately call Problem 3's function `get_manager` to get the last names of the managers for the employees with that job title.
- Look in the posted SQL script `328lect05-2.sql` at the **refactored** version of the stored procedure `job_overview` that we created during class on *Wednesday*.
 - Create an opening comment block for your function that has a **procedure:** part, a **purpose:** part, **AND** a **uses:** part, in the same style that you see here. (You don't have to give an `examples:` part, but you can if you wish.)
 - Follow that with the PL/SQL code creating your procedure.
- Remember to follow your PL/SQL procedure with:

```
/
```

```
show errors
```

- Then put a comment saying you are about to **test** your procedure `list_managers`.
- Follow that with at least two tests of your procedure, written in the same style as the tests for procedure `job_overview`, making sure that, for each test, you put a **prompt** command describing what results should be seen followed by the statements for that test.
 - At least one of these tests should be for a job title held by more than one employee.
 - At least one of these tests should be for a non-existent job title.
- Make sure that your `lab5.sql` ends with:
spool off

If successful, your resulting `lab5-out.txt` should show that your procedure successfully compiled, and that its tests passed.

BEFORE you leave lab:

Make sure that you **both** have copies of the files:

- `lab5.sql` and `lab5-out.txt`

...and you BOTH submit these two files using `~st10/328submit` on `nrs-projects`, with a lab number of **85**.

How the navigator can get files `lab5.sql` and `lab5-out.txt`:

These may be in a directory that is harder for the navigator to make a copy from than `public_html`.

For example -- they might be in a directory `328lab5` that is **not** a sub-directory of `public_html`.

Here is an approach for this:

- The **driver** should *temporarily* make the directory with these files world-readable and -executable, and these files world-readable -- assuming the driver is currently in their directory `328lab5`:

```
chmod 755 . # notice the space and the period!
```

```
chmod 644 lab5.sql lab5-out.txt
```

- Now the **navigator** can copy these into a directory of their choice -- assuming they are within the directory they want to copy into:

```
cp ~ab12/328lab5/* . # notice the space and the period!
```

- The **driver** and **navigator** should **BOTH** then **protect** these files:

```
chmod 600 lab5.sql lab5-out.txt
```

...and both can protect the directory containing them:

```
chmod 700 . # notice the space and the period!
```

...and **both** can now submit these using `~st10/328submit` from the directory containing these files.