# CIS 276 Lab  5 (50 points)

## Introduction

## The focus of this lab assignment is to create simple PL/SQL programs. You need to show that you understand the programming issues. Points will be DEDUCTED if you alter the question in any way. Use meaningful variable names and indent for readability. As before, you have write permission to the database tables in Oracle. Please remember the following key points:

## Follow the directions for each question.

## Work in your own private database. Use the userid and password assigned to you and not a shared userid.

## Include a comment box at the beginning of each file with your name, the lab number, and date.

## Format your output according to the SQL coding standards used in this class.

## Don't wait until the last minute to start this lab!

## For YOUR testing:

## reset your tables to the original data after a test so you know table contents

## you can use simple DELETEs or UPDATEs at the end of your program

## you can use the @ command to run the drop, create, and load script you created after the last assignment to refresh your tables.

## If anything is not clear, please ask questions.

## This lab will use the same tables as the previous labs:

## SALESPERSONS (Empid, Ename, Rank, Salary) CUSTOMERS (Custid, Cname, Credit) INVENTORY (Partid, Description, Stockqty, Reorderpnt, Price) ORDERS (Orderid, Empid, Custid, Salesdate) ORDERITEMS (Orderid, Detail, Partid, Qty)

## Submission Notes

* Create a separate .sql file for each task as indicated on each question naming each file according to the question number (i.e. q1.sql) just like you did in the previous lab.
* Zip the .sql files into a file called **lab5.zip**.
* Submit your **lab5.zip** file by the due date.

## Grading Notes

Each question is worth five points. Make sure you test your files. Any file that does not execute properly or produces an incorrect result will get zero (0) points. A grading script will be used to test your command files. There is no sample output for this lab since you have seen the SELECT queries before, and the data modification  queries don't produce any output. You should be able to write your own SELECT queries to verify the data modifications occurred as desired.

## Lab Assignment

1. Write a PL/SQL program that will declare two variables, vPartid and vDescription, and assign each values of 12345 and 'Fake Part' respectively.  Format your output line as "Part Number <vPartid> is a <vDescription>". No exception processing needed. Note: this does not require a SQL statement. Produce an output line that looks like the following line:

**Part Number 12345 is a Fake Part**

1. Write a PL/SQL program that will include a SQL statement that finds the description and price of the partid 1007.You must declare an additional variable to hold the value of the price. Embed an SQL statement that will select the description and price values from the inventory table and put them into the variables vDescription and vPrice. Assign the vPartid variable a value of 1007. The WHERE clause should not check for the constant 1007, but rather check for the value in the variable vPartid. Use the OTHERS EXCEPTION to display the SQLERRM. Format your output line as "Part Number <vPartid> is a <vDescription> that costs <vPrice>". Your output line should look like:

**Part Number 1007 is a THINGABOB that costs $12.50**

1. Write a PL/SQL program that includes a SQL statement that finds the partid, description, and price of the highest priced item in our inventory. Format your output line as "Part Number <vPartid> described as <vDescription> is the highest priced item at $<vPrice>". Assume there is only one part having the highest price! Use the OTHERS EXCEPTION to display the SQLERRM (there will be an exception when tested on multiple parts having the highest price). Your output line should look like the following:

**Part Number 1010 described as thingie is the highest priced item at $80.00**

1. What would happen if there were two or more items that have that same highest price???  (Let's say that the highest price is $80 and there are two or more items that have that price.)  Write the PL/SQL program that would handle this situation.  This requires using a CURSOR and a LOOP with a FETCH.  Use the OTHERS EXCEPTION to display the SQLERRM. Test your code for more than one item with the highest price.  Update your table in order to test this condition. You can change a price in one of the already existing partid's to match the maximum price or INSERT a new row.
2. Write a PL/SQL program that displays the employee number, name, and salary for all salespersons in ascending order of salary. This requires using a CURSOR and a LOOP with a FETCH. Use the OTHERS EXCEPTION to display the SQLERRM. Format your output line as follows:

**"<empid> - <ename> $<salary>"** i.e. your output should look something like:

**101 - Andrew Allen $1,000.00**

**102 - Burbank Burkett $1,000.00**

**110 - Larry Little $1,500.00**

**103 - Charles Cox $2,000.00**

**104 - Dale Dahlman $2,000.00**

**105 - Edward Everling $2,200.00**

**107 - Gloria Garcia $2,500.00**

**106 - Faulkner Forest $2,500.00**

**108 - Harvey Harrison $3,000.00**

**109 - Kevin Kody $3,100.00**

1. Write a PL/SQL program that accepts an empid (**&empid**) from the keyboard using variable substitution, and displays the employee number, name, and salary of the salesperson with the empid. Use the OTHERS EXCEPTION to display the SQLERRM. Format your output lines as

**"<empid> - <ename> $<salary>"**  i.e. your output should look something like:

**110 - Larry Little $1,500.00**

1. Write a PL/SQL program with no CURSOR to input a custid (**&custid**) and then display the customer's name and the total amount of all the customer's orders. The output for each test should be two values:  the customer's name and the total amount the customer has ordered. Test your program with a good customer id,  a customer id that does not exist in the database, and a customer id that exists in the database but has no orders.  Be sure to distinguish between the customer that does not exist and a customer that exists but has no orders. Use the NO\_DATA\_FOUND and OTHERS EXCEPTION handlers.
2. Write a PL/SQL program using a CURSOR for the same problem as #7 above. Do all the same tests as #7 and hopefully get the same results! Use the NO\_DATA\_FOUND and OTHERS exception handlers. The customer that has no orders can be checked using %ROWCOUNT in a condition.
3. Write a PL/SQL program to input a custid (**&custid**). Display the customer name of that customer and each orderid with salesdate and the total value of the order. Produce the output in descending order by value of each order. This output will produce one or more lines for the customer, depending on how many orders that customer has made. Test your program with a good  custid and one that is not in the CUSTOMERS table, and one that is in the CUSTOMERS table but has no orders. Be sure to distinguish between the customer that does not exist and a customer that exists but has no orders. Use the NO\_DATA\_FOUND and OTHERS exception handlers.

1. Write a PL/SQL program to input a partid (**&partid**). Show the customers who have ordered this part by displaying the customer name, quantity ordered, and salesdate of order. Display the information in descending order of salesdate. Test your program with a good partid, one that does not exist in the INVENTORY table and one that no customer has ordered. Be sure to distinguish between a partid that does not exist and a partid that exists but has not been ordered. Use the NO\_DATA\_FOUND and OTHERS exception handlers. You can modify your INVENTORY table for the third test (I will be doing so when grading).

**ATTENTION**: When you are done with all ten questions and prior to submittal, please change the substitution variables to numbers (i.e. for question 6 change **&empid** to **&1**). Then your assignment can be graded using a script that includes values instead of having to manually enter values -- this makes grading go much faster! (example: @q6 1002 will execute file q6.sql and provide input value 110)

Keep your output lines to 100 characters or less please.

Do you have a test plan?

What kinds of errors can happen with each program you write above?

Is there output that tells the user what went wrong?

Keywords of interest (some may require research): **SQLCODE, SQLERRM, OPEN, CLOSE, CURSOR, LOOP, FETCH, %ROWTYPE, %ROWCOUNT, %ISOPEN, %FOUND, WHILE, DBMS\_OUTPUT.PUT\_LINE, SET SERVEROUTPUT ON, NO\_DATA\_FOUND, TOO\_MANY\_ROWS, IF THEN END IF**

Here is a template for your programs (Indentation of four is good ) . . .

**/\***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**CIS276 @ PCC on Oracle10g with SQL Developer**

**Lab5, question #**

**Question verbage (copy/paste the question here)**

**Date Yourname Comments**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\*/**

**SET SERVEROUTPUT ON**

**DECLARE**

**-- Variables defined here;**

**BEGIN**

**-- Program code here;**

**EXCEPTION**

**WHEN OTHERS THEN**

**DBMS\_OUTPUT.PUT\_LINE(SQLERRM);**

**END;**