**CSIT 2520 Homework – PL/SQL Code Blocks (25 points)**

Perform each of the following tasks. Type answers to the questions directly into this document and copy/paste your output under each related question or create a script containing your question responses and code + results. Please cleanup the file before submitting to the PL/SQL HW#1 dropbox. Use the results document as a study aid for the final exam.

1. From the [Course Content] page of the online course, download the file **PL\_SQL\_Examples.txt**

to your class disk (located under Helpful Hints 🡪 Example Code). Review the contents of this file and use it to complete this lab.

2. **Read** Chapter 3, pp. 74-79 and the first part of Chapter 12 of your textbook and review the slide show named *PL\_SQL\_Introduction*. This should help you get started on this assignment.

**PL/SQL Code Blocks – Using the PL\_SQL\_Examples file**

3. In order to see the output from PL/SQL *put\_line* calls, you must type the command

**set serveroutput on;**

When using SQL Developer, you must choose DBMS Output option from the *View* menu to make the DBMS\_OUTPUT window/tab visible. Click the first icon (+) to turn on server output display.

4. Run Example 1 of the PL\_SQL\_Examples.txt file (using copy/paste). Repeat for several values of Height and area. Change the code to accept a value for the height and area from the user at the command line. (Hint: Use ampersands &).

declare

width number;

height integer := &var\_height;

area integer;

begin

area := &var\_area;

width := area/height;

dbms\_output.put\_line ('width = '|| width);

exception

when zero\_divide then

dbms\_output.put\_line('Division by zero');

end;

/

width = .5 (when height = 4 and area = 2)

5. Run Example 2a. Change it to display a different message that contains your name.

begin

dbms\_output.put\_line ('I am Candace and I like to play as a Hunter in World of Warcraft.');

end;

I am Candace and I like to play as a Hunter in World of Warcraft.

6. Run Example 2b. Explain here what the purpose of this code block is.

It’s like a switch statement but uses else-if’s instead. The block of code is assessing the value of month, which is substringed from a to\_chared sysdate, and testing each possible value and printing output based on that value.

7. Modify Example 2b by completing the code so that it displays a different (PG-rated) message for each month of the year. Personalize the messages so that they relate to a special meaning that each month holds for you. Test your code. Explain how you can test this code block for months other than the current month to be sure it will work for all months.

> declare

month char(3);

begin

month := substr(to\_char(sysdate, 'DD-MON-YY'), 4,3);

if (month = 'JAN') then

dbms\_output.put\_line ('January has Martin Luther Kings Birthday');

elsif (month = 'FEB') then

dbms\_output.put\_line ('February has Valentines Day');

elsif (month = 'MAR') then

dbms\_output.put\_line ('March has Spring Training');

elsif (month = 'APR') then

dbms\_output.put\_line ('April is for Fools');

elsif (month = 'MAY') then

dbms\_output.put\_line ('May has Mothers Day');

elsif (month = 'JUN') then

dbms\_output.put\_line ('June is for vacations');

elsif (month = 'JUL') then

dbms\_output.put\_line ('July is time for for fireworks');

elsif (month = 'AUG') then

dbms\_output.put\_line ('Sigh, August is time for school');

elsif (month = 'SEP') then

dbms\_output.put\_line ('September has Labor Day I think');

elsif (month = 'OCT') then

dbms\_output.put\_line ('Yay, my birthday is in October');

elsif (month = 'NOV') then

dbms\_output.put\_line ('November begins the time of year where everybody starts to get fat');

elsif (month = 'DEC') then

dbms\_output.put\_line ('December means snow and presents');

else

dbms\_output.put\_line ('You need to finish this example!');

end if;

end;

anonymous block completed

November begins the time of year where everybody starts to get fat

To test all months you could go in and manually set the month variable for each month that you wanted to test.

8. Run Example 2c. Explain here what the purpose of this code block is.

This example contains a loop that, when it starts, prints a statement. Then the loop starts by printing the value of counter and then incrementing it until counter is greater than 15. When the loop ends another statement is printed.

9. Modify Example 2c. to display only odd numbers between 11 and 33 and to accept a variable increment from the command line. Be sure to change the related message to match.

declare

counter integer := 11;

iterator integer := &var\_iterator;

begin

dbms\_output.put\_line ('This is the beginning of the series of numbers');

loop

if (MOD(counter, 2) != 0) then

dbms\_output.put\_line (counter);

end if;

counter := counter + iterator;

if (counter > 33) then

exit;

end if;

end loop;

dbms\_output.put\_line ('This is the end of the numbers that counts from 11 to 33 by ' || iterator);

end;

/

This is the beginning of the series of numbers

11

17

23

29

This is the end of the numbers that counts from 11 to 33 by 3

10. Modify the original Example 2c. to use a For…Loop instead of the loop….exit….end loop statement.

declare

counter integer := 11;

begin

dbms\_output.put\_line ('This is the beginning of the series of numbers');

for counter in 11..33 loop

if (MOD(counter, 2) != 0) then

dbms\_output.put\_line (counter);

end if;

end loop;

dbms\_output.put\_line ('This is the end of the numbers that counts from 11 to 33 by 1');

end;

/

This is the beginning of the series of numbers

11

13

15

17

19

21

23

25

27

29

31

33

This is the end of the numbers that counts from 11 to 33 by 1

11. Review example 2d. Add comments to the code block that describe the purpose of the block. Notice that the tense of the verb “to be” may not match output that follows it. Fix the message so that it matches the numeric values being displayed (e.g. 1 birth instead of 1 births)

declare

bday date; day\_of\_week varchar2(10); sunday number := 0;

monday number := 0; tuesday number := 0; wednesday number := 0;

thursday number := 0; friday number := 0; saturday number := 0;

cursor birth\_days is select dob from video\_customer; -- retrieve all the birth dates from video\_customer

begin

open birth\_days; --open the cursor

loop --start the loop to read through all the birth dates in birth\_days

fetch birth\_days into bday; --put a row retrieved from the cursor into the bday variable

if (birth\_days%notfound) then

exit;--if the end of the cursor is reached then exit the loop

end if;

day\_of\_week := rtrim(to\_char(bday, 'DAY'), ' '); --get the day of the week from the bday variable

--test day\_of\_week for each day of the week the birthday might be on or increment

--the week day variables above accordingly.

if (day\_of\_week = 'SUNDAY') then

sunday := sunday + 1;

elsif (day\_of\_week = 'MONDAY') then

monday := monday + 1;

elsif (day\_of\_week = 'TUESDAY') then

tuesday := tuesday + 1;

elsif (day\_of\_week = 'WEDNESDAY') then

wednesday := wednesday + 1;

elsif (day\_of\_week = 'THURSDAY') then

thursday := thursday + 1;

elsif (day\_of\_week = 'FRIDAY') then

friday := friday + 1;

elsif (day\_of\_week = 'SATURDAY') then

saturday := saturday + 1;

end if;

end loop;

close birth\_days;--close the cursor

--display how many birthdays were on each day of the week

if (sunday > 1 or sunday = 0) then

dbms\_output.put\_line ('There were '||sunday||' births on Sunday');

else

dbms\_output.put\_line ('There was '||sunday||' birth on Sunday');

end if;

if (monday > 1 or monday = 0) then

dbms\_output.put\_line ('There were '||monday||' births on Monday');

else

dbms\_output.put\_line ('There was '||monday||' birth on Monday');

end if;

if (tuesday > 1 or tuesday = 0) then

dbms\_output.put\_line ('There were '||tuesday||' births on Tuesday');

else

dbms\_output.put\_line ('There was '||tuesday||' birth on Tuesday');

end if;

if (wednesday > 1 or wednesday = 0) then

dbms\_output.put\_line ('There were '||wednesday||' births on Wednesday');

else

dbms\_output.put\_line ('There was '||wednesday||' birth on Wednesday');

end if;

if (thursday > 1 or thursday = 0) then

dbms\_output.put\_line ('There were '||thursday||' births on Thursday');

else

dbms\_output.put\_line ('There was '||thursday||' birth on Thursday');

end if;

if (friday > 1 or friday = 0) then

dbms\_output.put\_line ('There were '||friday||' births on Friday');

else

dbms\_output.put\_line ('There was '||friday||' birth on Friday');

end if;

if (saturday > 1 or saturday = 0) then

dbms\_output.put\_line ('There were '||saturday||' births on Saturday');

else

dbms\_output.put\_line ('There was '||saturday||' birth on Saturday');

end if;

end;

/

anonymous block completed

There was 1 birth on Sunday

There were 3 births on Monday

There was 1 birth on Tuesday

There were 3 births on Wednesday

There were 0 births on Thursday

There were 2 births on Friday

There was 1 birth on Saturday

12. Review and run example 2e. Does it give the expected result? Why or why not?

This seems like a trick question. If I wasn’t paying attention then I would say this gave unexpected results because there is an employee with the last name of Hobbs in the EMPLOYEES table. However, I was paying attention and noticed that the WHERE clause was looking for ‘HOBBS’ when in the table it is ‘Hobbs’. The upper function could have been used to fix this issue.

13. Copy 2e. to a new 2f. Modify the code to allow the first and last names to be input from the command line. Add the first and last name of the person you are trying to locate to the .*put\_line* messages. Show me that it works for existing names and gives errors for names that it cannot find.

declare

lname employees.last\_name%type;

fname employees.first\_name%type;

entered\_lname employees.last\_name%type := '&var\_lname';

entered\_fname employees.first\_name%type := '&var\_fname';

begin

select last\_name, first\_name

into lname, fname

from employees

where UPPER(last\_name) = UPPER(entered\_lname) and UPPER(first\_name) = UPPER(entered\_fname);

dbms\_output.put\_line ('EMPLOYEE ' ||entered\_fname||' '||entered\_lname|| ' LOCATED');

exception

when no\_data\_found then

dbms\_output.put\_line ('EMPLOYEE ' ||entered\_fname||' '||entered\_lname|| ' NOT FOUND');

end;

/

(When name is found)

EMPLOYEE James Smith LOCATED

(When name is not found)

EMPLOYEE Candace Williford NOT FOUND