**CITC 2340 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 Anonymous Code Blocks* [Assignments] submission folder. Use the results document as a study aid for the final exam.

1. From the online 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 &).

SQL> set serveroutput on;

SQL>

SQL> -- Example 1

SQL> --

SQL> declare

2 width integer;

3 height integer := 2;

4 area integer;

5 begin

6 area := 6;

7 width := area/height;

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

9 exception

10 when zero\_divide then

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

12 end;

13 /

width = 3

PL/SQL procedure successfully completed.

SQL>

SQL> declare

2 width integer;

3 height integer;

4 area integer;

5 begin

6 --area := 6;

7 width := &area/&height;

8 -- use the & to prompt for numeric input

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

10 exception

11 when zero\_divide then

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

13 end;

14 /

old:declare

width integer;

height integer;

area integer;

begin

--area := 6;

width := &area/&height;

-- use the & to prompt for numeric input

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

exception

when zero\_divide then

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

end;

new:declare

width integer;

height integer;

area integer;

begin

--area := 6;

width := 25/5;

-- use the & to prompt for numeric input

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

exception

when zero\_divide then

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

end;

width = 5

PL/SQL procedure successfully completed.

SQL> -- Try to divide by 0

SQL> declare

2 width integer;

3 height integer;

4 area integer;

5 begin

6 width := &area/&height;

7 -- use the & to prompt for numeric input

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

9 exception

10 when zero\_divide then

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

12 end;

13 /

old:declare

width integer;

height integer;

area integer;

begin

width := &area/&height;

-- use the & to prompt for numeric input

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

exception

when zero\_divide then

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

end;

new:declare

width integer;

height integer;

area integer;

begin

width := 30/0;

-- use the & to prompt for numeric input

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

exception

when zero\_divide then

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

end;

Division by zero

PL/SQL procedure successfully completed.

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

SQL> -- Example 2a.

SQL> --

SQL> begin

2 dbms\_output.put\_line ('I am a PL/SQL guru');

3 end;

4 /

I am a PL/SQL guru

PL/SQL procedure successfully completed.

-- change it to display my name

SQL>

SQL> begin

2 dbms\_output.put\_line ('Mihaela is a PL/SQL guru');

3 end;

4 /

Mihaela is a PL/SQL guru

PL/SQL procedure successfully completed.

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

SQL> declare

2 month char(3);

3 begin

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

5 if (month = 'JAN') then

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

7 elsif (month = 'FEB') then

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

9 elsif (month = 'MAR') then

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

11 elsif (month = 'APR') then

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

13 elsif (month = 'MAY') then

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

15 elsif (month = 'JUN') then

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

17 else

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

19 end if;

20 end;

21 /

April is for Fools

PL/SQL procedure successfully completed.

SQL> -- this code block pulls the month from sysdat. By using the format 'DD-MON-YY', the code looks for the

SQL> --3 characters starting in the 4th position (to\_char(sysdate, 'DD-MON-YY'), 4, 3), then it compares to all month

SQL> -- names in caps, 3-letter format, and chooses which output to print

SQL>

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.

1. Test your code for the current month.

SQL> declare

2 month char(3);

3 begin

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

5 if (UPPER(month) = 'JAN') then

6 dbms\_output.put\_line ('January has Grandpa''s Birthday');

7 elsif (UPPER(month) = 'FEB') then

8 dbms\_output.put\_line ('February has Valentine''s Day');

9 elsif (UPPER(month) = 'MAR') then

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

11 elsif (UPPER(month) = 'APR') then

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

13 elsif (UPPER(month) = 'MAY') then

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

15 elsif (UPPER(month) = 'JUN') then

16 dbms\_output.put\_line ('June is for Katie''s birthday');

17 elsif (UPPER(month) = 'JUL') then

18 dbms\_output.put\_line ('July is for Romanian vacation');

19 elsif (UPPER(month) = 'AUG') then

20 dbms\_output.put\_line ('August is for Gradmas'' Birthdays');

21 elsif (UPPER(month) = 'SEP') then

22 dbms\_output.put\_line ('September is for Aine, Layla, and Daddy''s Birthdays');

23 elsif (UPPER(month) = 'OCT') then

24 dbms\_output.put\_line ('October is for Hacker Halted');

25 elsif (UPPER(month) = 'NOV') then

26 dbms\_output.put\_line ('November is for Mommy''s Birthday');

27 elsif (UPPER(month) = 'DEC') then

28 dbms\_output.put\_line ('December is for Christmas');

29 else

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

31 end if;

32 end;

33 /

April is for Anniversary

PL/SQL procedure successfully completed.

1. Write a test code block that can test this code block for months other than the current month to be sure it will work for all months. (Hint: You have to modify the line of code that retrieves SYSDATE.)

SQL> set echo on;

SQL> set SERVEROUTPUT ON;

SQL> declare

2 month char(3);

3 begin

4 month := '&month';

5 if (UPPER(month) = 'JAN') then

6 dbms\_output.put\_line ('January has Grandpa''s Birthday');

7 elsif (UPPER(month) = 'FEB') then

8 dbms\_output.put\_line ('February has Valentine''s Day');

9 elsif (UPPER(month) = 'MAR') then

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

11 elsif (UPPER(month) = 'APR') then

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

13 elsif (UPPER(month) = 'MAY') then

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

15 elsif (UPPER(month) = 'JUN') then

16 dbms\_output.put\_line ('June is for Katie''s birthday');

17 elsif (UPPER(month) = 'JUL') then

18 dbms\_output.put\_line ('July is for Romanian vacation');

19 elsif (UPPER(month) = 'AUG') then

20 dbms\_output.put\_line ('August is for Gradmas'' Birthdays');

21 elsif (UPPER(month) = 'SEP') then

22 dbms\_output.put\_line ('September is for Aine, Layla, and Daddy''s Birthdays');

23 elsif (UPPER(month) = 'OCT') then

24 dbms\_output.put\_line ('October is for Hacker Halted');

25 elsif (UPPER(month) = 'NOV') then

26 dbms\_output.put\_line ('November is for Mommy''s Birthday');

27 elsif (UPPER(month) = 'DEC') then

28 dbms\_output.put\_line ('December is for Christmas');

29 else

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

31 end if;

32 end;

33 /

old:declare

month char(3);

begin

month := '&month';

if (UPPER(month) = 'JAN') then

dbms\_output.put\_line ('January has Grandpa''s Birthday');

elsif (UPPER(month) = 'FEB') then

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

elsif (UPPER(month) = 'MAR') then

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

elsif (UPPER(month) = 'APR') then

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

elsif (UPPER(month) = 'MAY') then

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

elsif (UPPER(month) = 'JUN') then

dbms\_output.put\_line ('June is for Katie''s birthday');

elsif (UPPER(month) = 'JUL') then

dbms\_output.put\_line ('July is for Romanian vacation');

elsif (UPPER(month) = 'AUG') then

dbms\_output.put\_line ('August is for Gradmas'' Birthdays');

elsif (UPPER(month) = 'SEP') then

dbms\_output.put\_line ('September is for Aine, Layla, and Daddy''s Birthdays');

elsif (UPPER(month) = 'OCT') then

dbms\_output.put\_line ('October is for Hacker Halted');

elsif (UPPER(month) = 'NOV') then

dbms\_output.put\_line ('November is for Mommy''s Birthday');

elsif (UPPER(month) = 'DEC') then

dbms\_output.put\_line ('December is for Christmas');

else

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

end if;

end;

new:declare

month char(3);

begin

month := 'jul';

if (UPPER(month) = 'JAN') then

dbms\_output.put\_line ('January has Grandpa''s Birthday');

elsif (UPPER(month) = 'FEB') then

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

elsif (UPPER(month) = 'MAR') then

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

elsif (UPPER(month) = 'APR') then

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

elsif (UPPER(month) = 'MAY') then

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

elsif (UPPER(month) = 'JUN') then

dbms\_output.put\_line ('June is for Katie''s birthday');

elsif (UPPER(month) = 'JUL') then

dbms\_output.put\_line ('July is for Romanian vacation');

elsif (UPPER(month) = 'AUG') then

dbms\_output.put\_line ('August is for Gradmas'' Birthdays');

elsif (UPPER(month) = 'SEP') then

dbms\_output.put\_line ('September is for Aine, Layla, and Daddy''s Birthdays');

elsif (UPPER(month) = 'OCT') then

dbms\_output.put\_line ('October is for Hacker Halted');

elsif (UPPER(month) = 'NOV') then

dbms\_output.put\_line ('November is for Mommy''s Birthday');

elsif (UPPER(month) = 'DEC') then

dbms\_output.put\_line ('December is for Christmas');

els

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

end if;

end;

July is for Romanian vacation

PL/SQL procedure successfully completed.

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

SQL> set serveroutput on;

SQL> declare

2 counter number := 1;

3 begin

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

5 loop

6 dbms\_output.put\_line (counter);

7 counter := counter + 1;

8 if (counter > 15) then

9 exit;

10 end if;

11 end loop;

12 dbms\_output.put\_line ('This is the end of the numbers that counts from 1 to 15 by 1');

13 end;

14 /

This is the beginning of the series of numbers

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

This is the end of the numbers that counts from 1 to 15 by 1

PL/SQL procedure successfully completed.

-- This code block prints all numbers from 1 to 15, counting by 1

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.

SQL> declare

2 counter number := 11;

3 begin

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

5 loop

6 dbms\_output.put\_line (counter);

7 counter := counter + 2;

8 if (counter > 33) then

9 exit;

10 end if;

11 end loop;

12 dbms\_output.put\_line ('This is the end of the numbers that counts from 11 to 33 by 2, printing only the odd numbers');

13 end;

14 /

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 2, printing only the odd numbers

PL/SQL procedure successfully completed.

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

SQL> declare

2 counter number := 1;

3 begin

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

5 FOR counter IN 1..15 -- the FOR loop will execute a specified number of times: as long as counter is between 1 and 15 inclusive

6 loop

7 dbms\_output.put\_line (counter); -- print the number value of the counter

8 end loop;

9 dbms\_output.put\_line ('This is the end of the numbers that counts from 1 to 15 by 1');

10 end;

11 /

This is the beginning of the series of numbers

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

This is the end of the numbers that counts from 1 to 15 by 1

PL/SQL procedure successfully completed.

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, etc.)

-- better code

SQL> declare

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

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

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

5 -- reserve memory to hold the birthday selected from the rental\_customer table

6 cursor birth\_days is select dob from rental\_customer; -- declare the explicit cursor

7 begin

8 open birth\_days; -- open the cursor

9 loop -- fetch the data rows

10 fetch birth\_days into bday;

11 if (birth\_days%notfound) then

12 exit; -- exit out of the loop when the pointer gets to the null row after the last row with values

13 end if;

14 day\_of\_week := rtrim(to\_char(bday, 'DAY'), ' '); -- trim out the day of the week out of the date value

15 -- add the day of the week to its respective counter

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

17 sunday := sunday + 1;

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

19 monday := monday + 1;

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

21 tuesday := tuesday + 1;

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

23 wednesday := wednesday + 1;

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

25 thursday := thursday + 1;

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

27 friday := friday + 1;

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

29 saturday := saturday + 1;

30 end if;

31 end loop;

32 close birth\_days; -- close the cursor

33 -- print the number of birtdays for each day of the week

34 if (sunday=1) then

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

36 elsif (sunday>1 OR sunday =0) then dbms\_output.put\_line ('There were '||sunday||' births on Sunday');

37 else dbms\_output.put\_line ('Error');

38 end if;

39 if (monday=1) then dbms\_output.put\_line ('There was ' || monday ||' birth on Monday');

40 elsif (monday>1 OR monday = 0) then dbms\_output.put\_line ('There were '||monday||' births on Monday');

41 else dbms\_output.put\_line ('Error');

42 end if;

43 if (tuesday=1) then dbms\_output.put\_line ('There was ' || tuesday || ' birth on Tuesday');

44 elsif (tuesday>1 OR tuesday = 0) then dbms\_output.put\_line ('There were '||tuesday||' births on Tuesday');

45 else dbms\_output.put\_line ('Error');

46 end if;

47 if (wednesday=1) then dbms\_output.put\_line ('There was ' || wednesday || ' birth on Wednesday');

48 elsif (wednesday>1 OR wednesday = 0) then dbms\_output.put\_line ('There were '||wednesday||' births on Wednesday');

49 else dbms\_output.put\_line ('Error');

50 end if;

51 if (thursday=1) then dbms\_output.put\_line ('There was ' || thursday || ' birth on Thursday');

52 elsif (thursday>1 OR thursday = 0) then dbms\_output.put\_line ('There were '||thursday||' births on Thursday');

53 else dbms\_output.put\_line ('Error');

54 end if;

55 if (friday=1) then dbms\_output.put\_line ('There was ' || friday || ' birth on Friday');

56 elsif (friday>1 OR friday = 0) then dbms\_output.put\_line ('There were '||friday||' births on Friday');

57 else dbms\_output.put\_line ('Error');

58 end if;

59 if (saturday=1 OR friday = 0) then dbms\_output.put\_line ('There was ' || saturday || ' birth on Saturday');

60 elsif (saturday>1) then dbms\_output.put\_line ('There were '||saturday||' births on Saturday');

61 else dbms\_output.put\_line ('Error');

62 end if;

63 end;

64 /

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

PL/SQL procedure successfully completed.

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

The result is “Employee not found,” even though there is an employee named ‘Hobbs.’ The last\_name in the where clause is written in all caps, whereas the employees table has names with an initial capital letter.

I changed the code and now it works. Thank you for your help!

-- 2.e Easy fix: UPPER(last\_name). Harder fix:

declare

lname employees.last\_name%type;

fname employees.first\_name%type;

counter integer:=0;

-- use a cursor, in case this returns more than one row

cursor c\_employees is select last\_name, first\_name from employees

where upper(last\_name) = 'HOBBS'; -- might not be unique

begin

open c\_employees;

loop

fetch c\_employees into lname, fname;

exit when c\_employees%notfound;

counter:= counter + 1;

end loop;

close c\_employees;

if (counter > 0) then

dbms\_output.put\_line ('EMPLOYEE LOCATED');

elsif (counter = 0) then

dbms\_output.put\_line ('EMPLOYEE NOT FOUND');

end if;

end;

/

PL/SQL procedure successfully completed.

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. Your code should work regardless of the case used to input the data.

-- look up Andrew Hobbs

old:declare

lname employees.last\_name%type;

fname employees.first\_name%type;

begin

select last\_name, first\_name

into lname, fname

from employees

where initcap(last\_name) = '&last\_name' AND initcap(first\_name) = '&first\_name';

dbms\_output.put\_line ('EMPLOYEE LOCATED');

exception

when no\_data\_found then

dbms\_output.put\_line ('EMPLOYEE NOT FOUND');

end;

new:declare

lname employees.last\_name%type;

fname employees.first\_name%type;

begin

select last\_name, first\_name

into lname, fname

from employees

where initcap(last\_name) = 'Hobbs' AND initcap(first\_name) = 'Andrew';

dbms\_output.put\_line ('EMPLOYEE LOCATED');

exception

when no\_data\_found then

dbms\_output.put\_line ('EMPLOYEE NOT FOUND');

end;

EMPLOYEE NOT FOUND

PL/SQL procedure successfully completed.

-- look up Fred Hobbs

SQL> declare

2 lname employees.last\_name%type;

3 fname employees.first\_name%type;

4 begin

5 select last\_name, first\_name

6 into lname, fname

7 from employees

8 where initcap(last\_name) = '&last\_name' AND initcap(first\_name) = '&first\_name';

9 dbms\_output.put\_line ('EMPLOYEE LOCATED');

10 exception

11 when no\_data\_found then

12 dbms\_output.put\_line ('EMPLOYEE NOT FOUND');

13 end;

14 /

old:declare

lname employees.last\_name%type;

fname employees.first\_name%type;

begin

select last\_name, first\_name

into lname, fname

from employees

where initcap(last\_name) = '&last\_name' AND initcap(first\_name) = '&first\_name';

dbms\_output.put\_line ('EMPLOYEE LOCATED');

exception

when no\_data\_found then

dbms\_output.put\_line ('EMPLOYEE NOT FOUND');

end;

new:declare

lname employees.last\_name%type;

fname employees.first\_name%type;

begin

select last\_name, first\_name

into lname, fname

from employees

where initcap(last\_name) = 'Hobbs' AND initcap(first\_name) = 'Fred';

dbms\_output.put\_line ('EMPLOYEE LOCATED');

exception

when no\_data\_found then

dbms\_output.put\_line ('EMPLOYEE NOT FOUND');

end;

EMPLOYEE LOCATED

PL/SQL procedure successfully completed.