CSC 352 / 452: Database Programming

assignment #4 (60 Points)

**CSC 352/452-501: Due on Tuesday, 8/4/2015 at 11:59PM**

**CSC 352/452-510: Due on Wednesday, 8/5/2015 at 11:59PM**

Unless prior arrangements are made, homework turned in late but within 24 hours of the due time will be graded at 75% credit, homework turned in between 24 and 48 hours will be graded at 50% credit, and homework turned in later than 48 hours will not be accepted.

**Please note that only TEXT files will be accepted. All other file types (e.g., DOC, DOCX, RTF, PDF, JPG, or ZIP) will be rejected.**

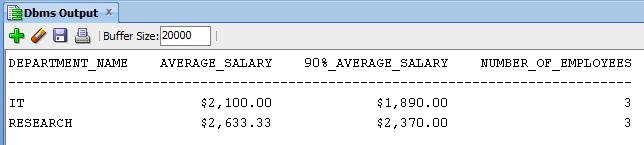
* You are not allowed to create temporary tables, views, procedures, or functions.
* Please review your assignment file before submitting it to make sure you have the correct one. It is your responsibility to ensure that you upload the correct assignment file.

**1a) (CSC 352 only - 30 points)**

Based on the tables created in Assignment #1, write a PL/SQL anonymous block to find all departments that have more than **TWO** employees whose salaries are above **90%** of the average salary of their respective departments. Your program must display the department name, the average salary of the department, 90% of the average salary of the department, and the number of the corresponding employees. Sort your output in ascending order by the department name.

Hard coding the department name (IT, RESEARCH), the number of departments (2), or the number of employees (3) will get a zero point.

The output of your program must match the following:



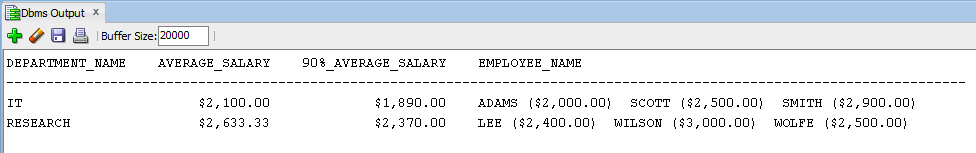
**1b) (CSC 452 only – 30 points)**

Based on the tables created in Assignment #1, write a PL/SQL anonymous block to find all departments that have more than **TWO** employees whose salary is above **90%** of the average salary of their respective departments.

Your program must display the department names, the average salary of the department, 90% of the average salary of the department, and the corresponding employee names alone with their salaries. Make sure that employees from the same department must be in the same line and separated by blank spaces (e.g., LEE ($2,400.00) WILSON ($3,000.00)). You will lose some points if you fail to do so. Sort your output in ascending order by the department name and then the employee name.

Hard coding department names (IT, RESEARCH), the number of departments (2), or the number of employees (3) will get a zero point.

The output of your program must match the following:



Hints:

* The second cursor should use a value from the first cursor in the "WHERE clause".
* You may use your variable v\_output to display the employee names,

……

v\_output := v\_output || ' '|| idx\_2.employee\_name || ' (' ||

TRIM(TO\_CHAR(ROUND(idx\_2.salary, 2), '$99,990.00')) || ')';

……

DBMS\_OUTPUT.PUT\_LINE(v\_output).

**2a) (CSC 352 only - 30 points)**

Based on the EMPLOYEE table created in Assignment #1, write a PL/SQL anonymous block that accepts an employee ID from the user input and displays all direct and indirect managers (supervisors) of the given employee.

* If the employee ID from the user input is not in the EMPLOYEE table (EMPLOYEE\_ID), you display a message telling the user that the employee ID is not in the table.
* If the employee does not have a manager, the manager name must be shown as “N/A” in your output.
* Hard coding, except the string'NULL', is not allowed in your program.
* You cannot assume that KING (employee\_id = 7839) is always the company’s top manager (the president of the company).
* Creating cleanly formatted output is a common programming requirement. The format of your output must match mine EXACTLY. For example,

No. Employee Name Manager Name

------------------------------------------

1 LANCASTER FISHER

2 FISHER CLARK

3 CLARK KING

4 KING N/A

If your output does not match mine EXACTLY, you will lose some points.

* To avoid complicating issues, you can assume that the user always enters input from keyboard that consists only of the digits 0 through 9 and Enter.

Hints:

* It is NOT necessary to use a cursor to get the output. (some SELECT-INTO statements are enough.)
* Step 1) Receive the user input (v\_eid).
* Step 2) Check whether v\_eid is in the EMPLOYEE table.
* Step 3) Find all managers of the given employee by processing your SELECT-INTO statements repeatedly. For example

WHILE -- beginning of your loop

Find the employee name and manager id based on v\_eid.

…… -- your statements

Display the employee name and manager name.

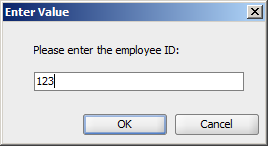
Exit the loop when the manager id is NULL.

Update v\_eid using the manager\_id.

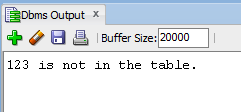
END LOOP; -- end of your

Test your program to ensure that you get the correct results.

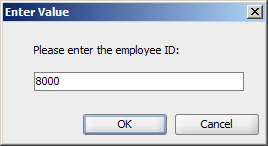
Case 1)



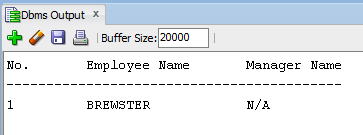
**Dbms Output:**



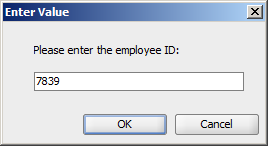
Case 2a)



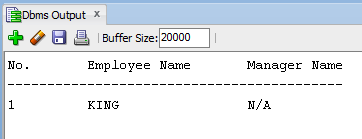
**Dbms Output:**



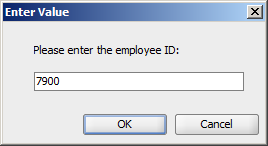
Case 2b)



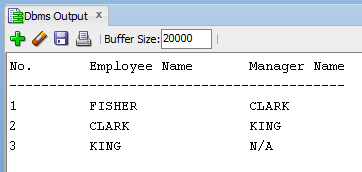
**Dbms Output:**



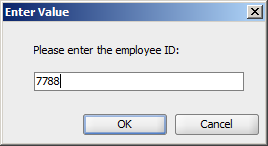
Case 3a)



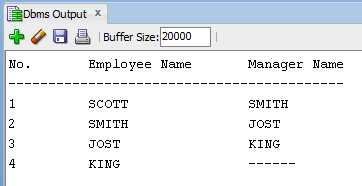
**Dbms Output:**



Case 3b)



**Dbms Output:**



**2b) (CSC 452 only - 30 points)**

The table REGION contains some regions of the world.

CREATE TABLE region

(

PARENT\_NAME VARCHAR2(30),

CHILD\_NAME VARCHAR2(30)

);

/

INSERT INTO region VALUES(NULL,'Asia');

INSERT INTO region VALUES(NULL,'Australia');

INSERT INTO region VALUES(NULL,'Europe');

INSERT INTO region VALUES(NULL,'North America');

INSERT INTO region VALUES('Asia','China');

INSERT INTO region VALUES('Asia','Japan');

INSERT INTO region VALUES('Australia','New South Wales');

INSERT INTO region VALUES('New South Wales','Sydney');

INSERT INTO region VALUES('Canada','Ontario');

INSERT INTO region VALUES('China','Beijing');

INSERT INTO region VALUES('England','London');

INSERT INTO region VALUES('Europe','United Kingdom');

INSERT INTO region VALUES('Illinois','Aurora');

INSERT INTO region VALUES('Illinois','Cook County');

INSERT INTO region VALUES('Illinois','Rockford');

INSERT INTO region VALUES('Wisconsin','Madison');

INSERT INTO region VALUES('Japan','Osaka');

INSERT INTO region VALUES('Japan','Tokyo');

INSERT INTO region VALUES('North America','Canada');

INSERT INTO region VALUES('North America','United States');

INSERT INTO region VALUES('Ontario','Ottawa');

INSERT INTO region VALUES('Ontario','Toronto');

INSERT INTO region VALUES('United States','Colorado');

INSERT INTO region VALUES('United States','Illinois');

INSERT INTO region VALUES('United States','Texas');

INSERT INTO region VALUES('United Kingdom','England');

INSERT INTO region VALUES('Texas','Rockford');

INSERT INTO region VALUES('Colorado','Aurora');

INSERT INTO region VALUES('Cook County','Chicago');

COMMIT;

Based on the REGION table, write a PL/SQL anonymous block that accepts a region name (CHILD\_NAME) from the user input and displays all its parent regions.

Hard coding, except the string'NULL', is not allowed in your program. You cannot assume that the maximum number of region names is always 5.

Hints:

* Step 1) Define a cursor to find the parent for a given child. (One cursor is enough.)
* Step 2) Receive the user input and display it.

Is the user input in the region table (CHILD\_NAME)?

* Step 3) Find all the parents by processing your cursor repeatedly. For example

WHILE -- beginning of your loop

open your cursor

your statements

(e.g., v\_output := v\_output || '--> ' || ...;)

close your cursor

your statements

(e.g., v\_child := v\_parent;)

END LOOP; -- end of your

* Use the UPPER function to compare two region names.
* You may use your variable v\_output to display your output (e.g., v\_output

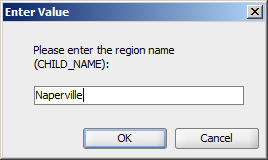
:= v\_output || '…'; DBMS\_OUTPUT.PUT\_LINE(v\_output)).

Make sure that the region names in your output are in the same line and separated by “-->”.

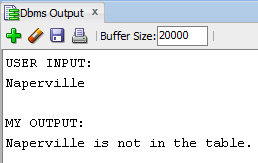
Creating cleanly formatted output is a common programming requirement. The format of your output must match mine EXACTLY. If your output does not match mine EXACTLY (e.g., missing “USER INPUT:”, “MY OUTPUT:”, “[”, “]”, “(1)”, or “-->” in your output), you will lose some points.

Test your program to ensure that you get the correct results.

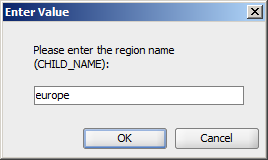
Case 1)



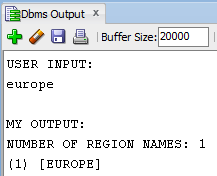
**Dbms Output:**



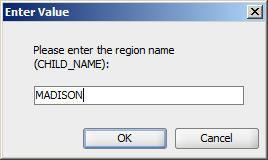
Case 2)



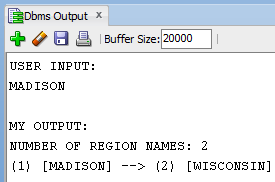
**Dbms Output:**



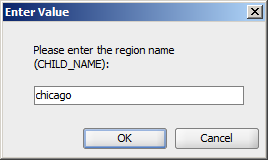
Case 3)



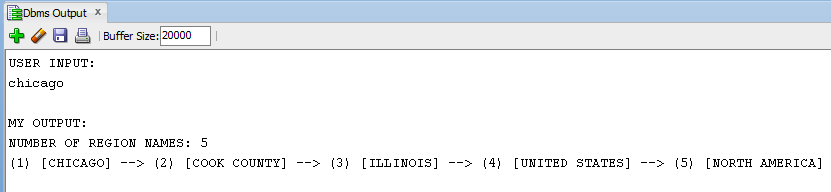
**Dbms Output:**



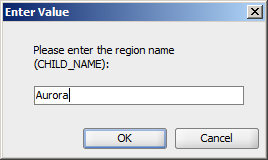
Case 4)



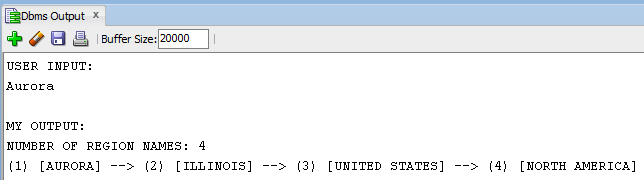
**Dbms Output:**



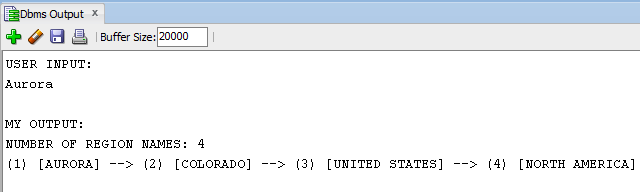
Case 5)



**Dbms Output:**



Or



With input Aurora, we cannot guarantee that the Illinois city comes up before the Colorado city. (We are relying on Oracle's internal row order). Please take a look the “optional question” because it is deterministic.

**Please submit a text file containing all the source codes to D2L before or on due date.**

**Optional Question**

**Just for fun (no credit, no extra credit, no need to submit, just for if you are a curious person and like database programming).**

We have duplicate city names in the CHILD\_NAME column. Modify your program created in (2) such that all parent regions of the duplicate city names can be displayed.

Test your program to ensure that you get the correct results.

Case 1) User input: Naperville

Case 2) User input: Europe

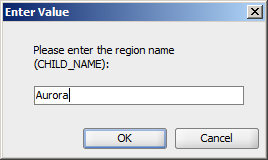
Case 3) User input: Madison

Case 4) User input: Chicago

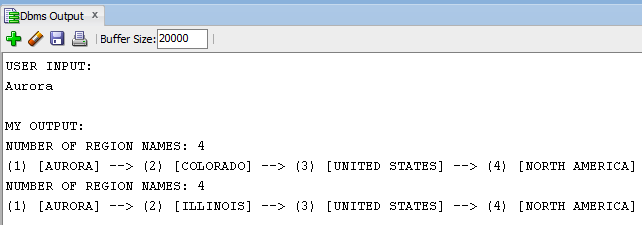
Case 5) User input: Aurora

Case 6) User input: Rockford

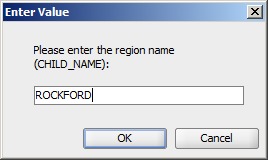
Case 5) User input: aurora



**Dbms Output:**



Case 6) User input: Rockford



**Dbms Output:**

