CSC 352 / 452: Database Programming

assignment #4 (60 Points)

**CSC 352/452-501: Due on Tuesday, 8/1/2017 at 11:59PM**

**CSC 352/452-510: Due on Wednesday, 8/2/2017 at 11:59PM**

Unless prior arrangements are made, homework turned in late will not be accepted. However, homework turned in within 24 hours late will be graded at 50% credit.

**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. In D2L, only the most recent submission is kept.**

* 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 upload the correct assignment file.

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

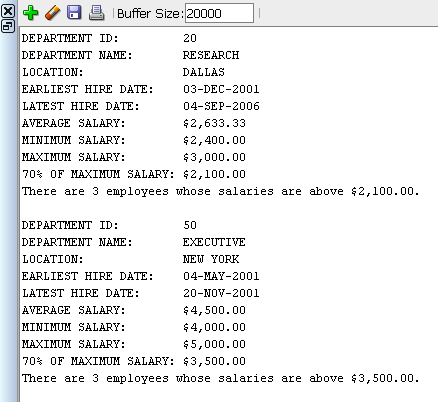
Based on the tables created in Assignment #1, write a PL/SQL anonymous block to find all departments that have more than **TWO** employees whose **salaries** are above **70%** of the maximum (highest) salary of their respective departments. For each of these departments, your program must display the department ID, department name, and location of the department, the earliest and latest hire dates in the department, the average salary of the department, the minimum salary of the department, the maximum salary of the department, 70% of the maximum salary of the department, and the number of the corresponding employees. Sort your output in ascending order by the department ID.

Hard coding the department ID (e.g., 20, 50), the department name (e.g., EXECUTIVE, RESEARCH), the number of departments (e.g., 2), the number of employees (e.g., 3), or the maximum salary will receive 0 points.

Hint: using 1 cursor is enough.

*(This question should take less than 45 minutes to solve.)*

The output of your program must match the following:



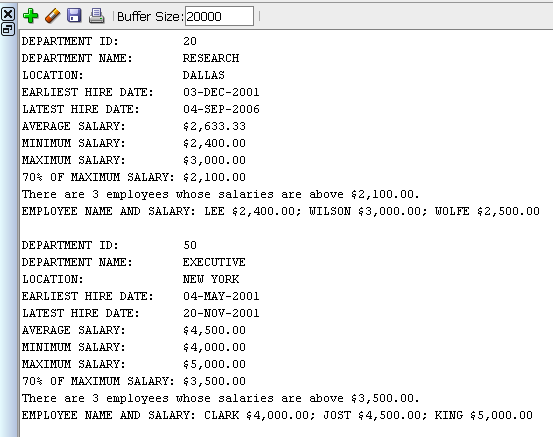
**1b) (CSC 452 only – 30 points)**

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 **salaries** are above **70%** of the maximum (highest) salary of their respective departments. For each of these departments, your program must display the department ID, department name, and location of the department, the earliest and latest hire dates in the department, the average salary of the department, the minimum salary of the department, the maximum salary of the department, 70% of the maximum salary of the department, the number of the corresponding employees, and the corresponding employee names alone with their salaries. Make sure that employees from the same department must be in the same line and each name-salary pair is separated by a semicolon (;). (e.g., LEE $2,400.00; WILSON $3,000.00; WOLFE $2,500.00). You will lose some points if you fail to do so. Sort your output in ascending order by the department ID and then the employee name.

Hard coding the department ID (e.g., 20, 50), the department name (e.g., EXECUTIVE, RESEARCH), the number of departments (e.g., 2), the number of employees (e.g., 3), or the maximum salary will receive 0 points.

*(This question should take less than 60 minutes to solve.)*

The output of your program must match the following:



Hints:

* You need to use two cursors to get the output. The examples on pages 23-29 tell you how to declare and process two cursors.
* 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'));

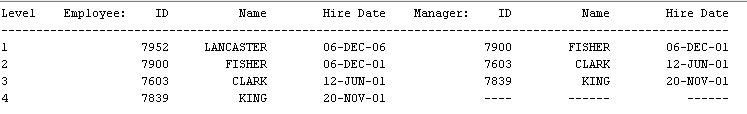
……

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 (ID, name, and job title) must be shown as “------” in your output.
* 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,



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 may be enough.) The main part is as follows:

LOOP *-- beginning of your loop*

Find the employee name and manager\_ID based on v\_user\_input\_eid.

…… *-- your statements*

Display the employee (ID, name, and job title) and manager (ID, name, and job title)

(or “------”).

Exit the loop when the manager ID is NULL.

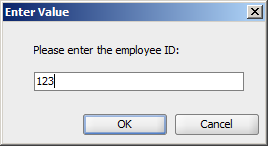
Update v\_user\_input\_eid (v\_user\_input\_eid <- manager ID).

END LOOP; *-- end of your loop*

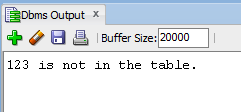
*(This question should take less than 45 minutes to solve.)*

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

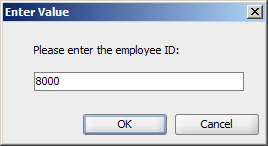
Case 1)



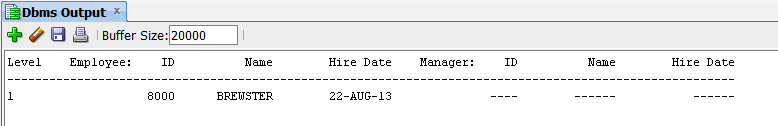
**Dbms Output:**



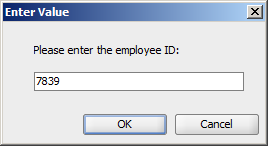
Case 2a)



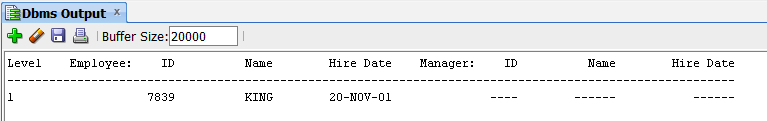
**Dbms Output:**



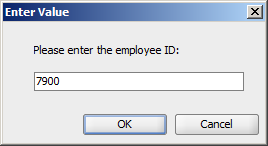
Case 2b)



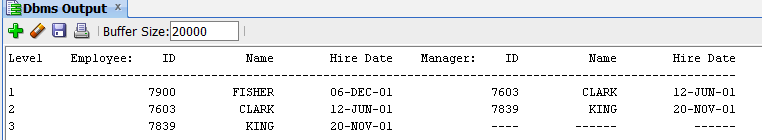
**Dbms Output:**



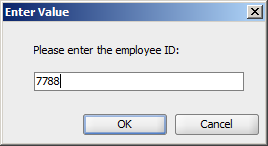
Case 3a)



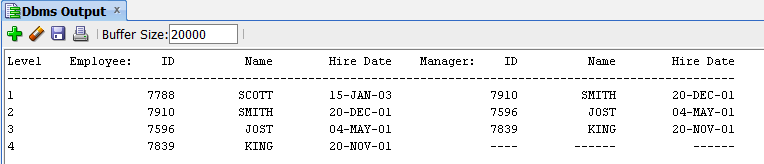
**Dbms Output:**



Case 3b)



**Dbms Output:**



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

The table WORLD\_REGION contains some regions of the world.

CREATE TABLE world\_region

(

PARENT\_REGION VARCHAR2(30) NOT NULL,

CHILD\_REGION VARCHAR2(30) NOT NULL

);

/

INSERT INTO world\_region VALUES('\*\*\*','Asia');

INSERT INTO world\_region VALUES('\*\*\*','Australia');

INSERT INTO world\_region VALUES('\*\*\*','Europe');

INSERT INTO world\_region VALUES('\*\*\*','North America');

INSERT INTO world\_region VALUES('Asia','China');

INSERT INTO world\_region VALUES('Asia','Japan');

INSERT INTO world\_region VALUES('Australia','New South Wales');

INSERT INTO world\_region VALUES('New South Wales','Sydney');

INSERT INTO world\_region VALUES('Canada','Ontario');

INSERT INTO world\_region VALUES('China','Beijing');

INSERT INTO world\_region VALUES('England','London');

INSERT INTO world\_region VALUES('Europe','United Kingdom');

INSERT INTO world\_region VALUES('Illinois','Aurora');

INSERT INTO world\_region VALUES('Illinois','Cook County');

INSERT INTO world\_region VALUES('Illinois','Rockford');

INSERT INTO world\_region VALUES('Wisconsin','Madison');

INSERT INTO world\_region VALUES('Japan','Osaka');

INSERT INTO world\_region VALUES('Japan','Tokyo');

INSERT INTO world\_region VALUES('North America','Canada');

INSERT INTO world\_region VALUES('North America','United States');

INSERT INTO world\_region VALUES('Ontario','Ottawa');

INSERT INTO world\_region VALUES('Ontario','Toronto');

INSERT INTO world\_region VALUES('United States','Colorado');

INSERT INTO world\_region VALUES('United States','Illinois');

INSERT INTO world\_region VALUES('United States','Texas');

INSERT INTO world\_region VALUES('United Kingdom','England');

INSERT INTO world\_region VALUES('Texas','Rockford');

INSERT INTO world\_region VALUES('Colorado','Aurora');

INSERT INTO world\_region VALUES('Cook County','Chicago');

COMMIT;

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

You will receive 0 points if you use different table/column names.

You cannot assume that the maximum number of region names is 5.

Hints:

* You can use hard-coded string '\*\*\*' in your program.
* Define a cursor to find the parent region for a given child region (**v\_child**). (One cursor is enough.)
* Find all the parent regions by processing your cursor **repeatedly**. (The example on pages 22-23 tells you how to define a cursor with parameters and open the cursor repeatedly.) The main part is as follows:

LOOP *-- beginning of your loop*

Open your cursor.

*Your statements*

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

Close your cursor.

Y*our statements*

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

END LOOP; *-- end of your*

* Use the UPPER/LOWER function to compare two region names.
* Make sure that the region names in your output are in the same line and separated by “*==>*”.

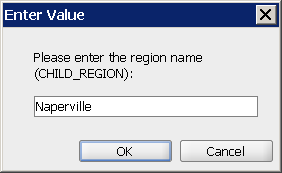
Creating cleanly formatted output is a common programming requirement. The format of your output must match mine EXACTLY. If your output does not match mine EXACTLY (e.g., missing “USER INPUT:”, “MY OUTPUT:”, “[”, “]”, “(1)”, or “*==>*” in your output), you will lose some points.

*(If you have good PL/SQL skills, your answer should be short, e.g., no more than 40 lines.)*

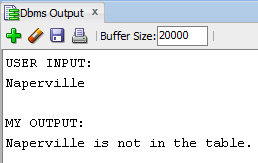
*(This question should take less than 60 minutes to solve.)*

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

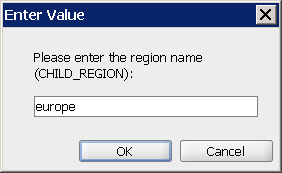
Case 1)



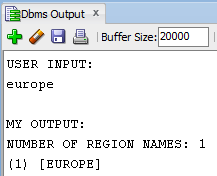
**Dbms Output:**



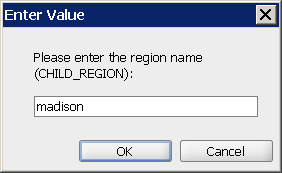
Case 2)



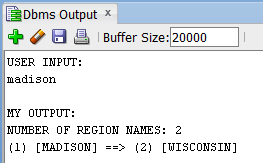
**Dbms Output:**



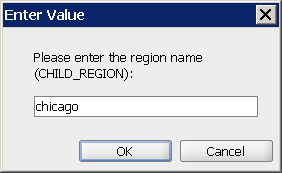
Case 3)



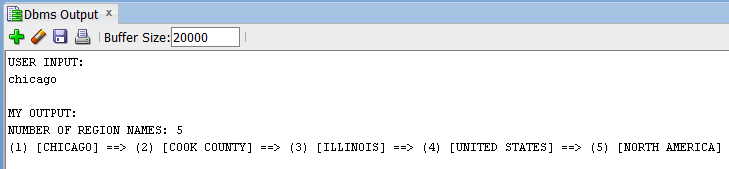
**Dbms Output:**



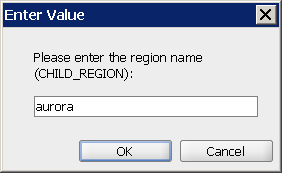
Case 4)



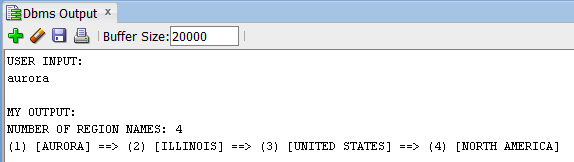
**Dbms Output:**



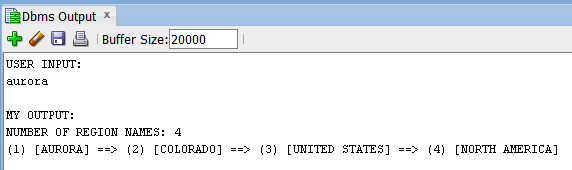
Case 5)



**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 by the 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\_REGION column. Modify your program created in (2b) 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

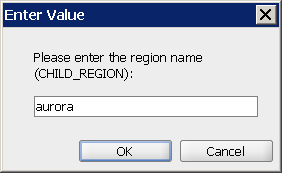
Case 4) User input: Chicago

Case 5) User input: Aurora

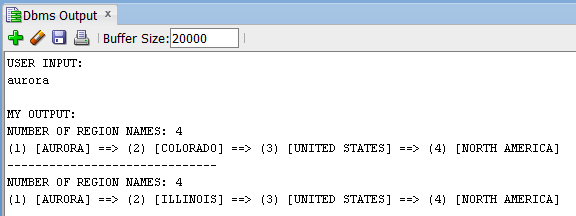
Case 6) User input: Rockford

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

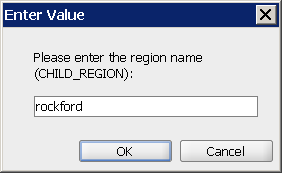
Case 5) User input: aurora



**Dbms Output:**



Case 6) User input: Rockford



**Dbms Output:**

