Home assignment #6

Due date Wednesday, 5/25/2022 8:00PM

If you have difficulties, you may refer to the hints at the end of this file.

Q1. (40 points in total) Local procedure, Standalone Procedure and Standalone Function.

The HR plans to grant the employees one-time bonus, the amount will be decided as the policy below for whole company:

For employees that get commission\_pct equal to or greater than .25:

if salary > = 10,000 then bonus := 1,000 ;

if salary >= 7,000 (and salary < 10000) then bonus := 800 ;

else bonus := 600 ;

For employees that get commission\_pct less than .25 and greater than 0) :

if salary > = 10,000 then bonus := 700 ;

if salary >= 7,000 (and salary < 10,000) then bonus := 600 ;

else bonus := 500 ;

For employees that get no commission, (commission\_pct is null):

if salary > = 10,000 then bonus := 1500 ;

if salary >= 7,000 (and salary < 10,000 ) then bonus := 1200 ;

else bonus := 1100 ;

Below are some code you may change and reuse:

SELECT salary, NVL (commission\_pct, 0) comm\_rate

INTO sal, comm -- sal and comm are your local variables

FROM employees

Where employee\_ID = emp\_ID; -- emp\_id or anything you declare

IF comm >= .25 THEN

IF sal >= 10000 THEN bonus := 1000;

ELSIF sal >= 7000 THEN bonus := 800;

ELSE bonus := 600;

END IF; -- end of nested IF

ELSIF (comm < .25 and comm > 0) THEN

IF sal >= 10000 THEN bonus := 700;

ELSIF sal >= 7000 THEN bonus := 600;

ELSE bonus := 500;

END IF;

ELSE -- or ELSIF comm = 0 THEN

IF sal >= 10000 THEN bonus := 1500;

ELSIF sal >= 7000 THEN bonus := 1200;

ELSE bonus := 1100;

END IF;

END IF;

Q1 (a) (15 points) local (nested) procedure.

Write an anonymous PL/SQL block.

Inside this block (in the declarative section), define a local procedure called Emp\_bonus. You may refer to the Example 2c on Noted7a. That local procedure will have one IN mode parameter of number data type for receiving employee\_id, three OUT parameters, called Sal, Comm and Bonus, using number as their data type. The procedure will retrieve necessary info related to that employee from table Employees, calculate the bonus amount for that individual employee.

In the Executable section of this PL/SQL block, the program will retrieve the employee ID for those in department 80, invoke that procedure Emp\_bonus, passing the employee ID to that procedure and receive the OUT values (assign these to local variables), then print out the employee ID, the employee’s salary, commission percentage, and bonus for that employee.

Please print out the result for department 80.

Q1 (b) (15 points) Standalone Procedure.

In question (b), create a standalone procedure called Emp\_Bonus. This procedure plays the same role as the local procedure in Q1 (a).

After having successfully completed the procedure, write an anonymous PL/SQL program, call the procedure to get the bonus for each employee in department 80, print out same info as in (a).

Q1(c). (10 points) Standalone Function

Similar to Q1(b) but not same. This time, you will create a standalone function called F\_Emp\_Bonus. In this function you will have one IN parameter of emp\_id, no OUT parameter. Your function will RETURN the amount of bonus for that employee. (only one value to return)

In the anonymous PL/SQL, you will call this function repeatedly for each employee in department 80. The program will print the employee ID and bonus.

Q2. (20 points) Function returns a data type of record. This question is based on question 1(c).

In question 1 (c), we only return one scalar value of bonus back to the invoker.

In this question, you will write a PL/SQL block,

1. you need to define a data type of RECORD named E\_bonus, that has four fields

Emp\_ID number (6), -- correspondent to employee\_id

Sal number (8, 2), -- as salary

Comm number (2, 2), -- as commission\_pct

Bonus number (6) -- the result of calculated bonus

1. In the declarative section, you will define a (nested) function named F\_Emp\_Bonus, that will accept the employee\_id as IN parameter, return a record data type of E\_bonus (just defined above). The function will use the employee ID (IN parameter) to retrieve the employee’s salary and commission percentage to calculate the bonus, populate the record fields with the right values.
2. In the executable section, your program will run a loop to invoke this function E\_bonus for all the employees that work for department 80. The program will print out the related info from the records as in question 1(a).

Q3. (20 points) Function overload.

We have used some functions overloaded, such as function TO\_CHAR. This question will provide a chance for you to practice. As we do not discuss package yet, this overload must compete in one PL/SQL block, cannot be “create” two times separately.

In a PL/SQL block, define two (nested, or say local) functions with the same name as Dept\_Head\_FName with different parameter names and data types. Both functions have one IN parameter. The first function use deptno (department\_id) as its IN parameter, its data type is number. The second function has dept\_name (department\_name) as its IN parameter, the data type is varchar2. Both functions will return the full name of the head/manager of that department.

In the executable section in the PL/SQL, it will call the function twice. One uses 60 as department\_id. The second time, uses ‘IT’ as actual parameter value. Your program will print out the results.

Q4. (20 points)

In note6b\_Records, on page 5, << More Record Level operations >>, we mentioned that the current Oracle version does not support record level comparison. For frequently used record type, we can create a function to complete this task.

Write an anonymous PL/SQL block, it will declare the record type as below:

TYPE emp\_name IS RECORD (

f\_name varchar2 (20),

l\_name varchar2 (25)

);

Define a function called Emp\_name\_eq, that has two IN parameters with data type same as the record type emp\_name. The function will return a Boolean data type value.

The function will compare each field of the two IN parameters. If each correspondent field from two records is same then the function will return TRUE, Otherwise, the function will return FALSE.

In the executable section of the program, you will declare three variables of the type emp\_name, emp1, emp2 and emp3. Initialize emp1 using the first\_name and last\_name from employees table where employee ID of 202. Then assign emp2 same values as emp1. Assign emp3 with values of Winston Taylor.

After assigning the values to these three variables, your program will

1. invoke the function to compare emp1 and emp2, then print out the result of the comparison.
2. invoke the function to compare emp1 and emp3, then print out the result of the comparison.

\*\* Note:

You cannot pass a BOOLEAN value to the DBMS\_OUTPUT.PUTLINE

subprogram. To print a BOOLEAN value, use an IF or CASE statement to translate it to a

character value.

Hints for Q1 (a).

Declare

e\_sal number (8, 2) := 0;

e\_comm number (2, 2) := 0;

e\_bonus number (8, 2) := 0;

Cursor C IS

SELECT employee\_id from employees where department\_id in (60);

Procedure Emp\_Bonus ( Emp\_ID number,

Sal OUT Number, Comm OUT Number, Bonus OUT Number )

IS

BEGIN

SELECT salary, NVL (commission\_pct, 0) comm\_rate

INTO sal, comm

FROM employees

Where employee\_ID = emp\_ID;

IF comm >= .25 THEN

...

END IF;

END Emp\_Bonus ;

Begin

DBMS\_OUTPUT.PUT\_LINE (' Employee ID Salary Commission PCT Bonus ');

DBMS\_OUTPUT.PUT\_LINE ('------------ --------- -------------- --------- ');

For indx in C LOOP

Emp\_Bonus ( indx.employee\_id, e\_sal, e\_comm, e\_bonus );

DBMS\_OUTPUT.PUT\_LINE ( RPAD (indx.employee\_id, 12) ||

To\_char (e\_sal, '$99,999') || LPAD (e\_comm, 15) || To\_char (e\_bonus, '$99,999') );

END loop;

END;

OUTPUT

Q1 (b),

Make the procedure as an independent “create” statement.

Then write a PL/SQL block, from there invoke this procedure.

Q1 (c), only return one scalar value for this question.

Q2. Similar style as Q1 (a), the function is declared inside the PL/SQL block,

Declare

TYPE E\_bonus IS RECORD

( Emp\_id number (6),

Sal number (8, 2),

comm number (2, 2),

bonus number (6) );

v E\_bonus;

Cursor c IS

SELECT employee\_id, ...

Function F\_Emp\_Bonus ( Emp\_ID number)

RETURN E\_bonus

IS

. . . Codes: give me the emp id, I get the salary back/return to you.

RETURN r ;

END F\_Emp\_Bonus ;

Begin

For indx in C LOOP

v := F\_Emp\_Bonus ( indx.employee\_id);

DBMS\_OUTPUT.PUT\_LINE

. . .

End;

Q3.

Declare

Dpt\_id employees.department\_id%TYPE := 60;

dept\_name varchar2 (30) := 'it' ;

Dpt\_head\_fname varchar2 (47);

Function Dept\_Head\_FName

( Deptno IN number)

RETURN varchar2

IS

. . .

RETURN Dept\_Head\_fullname;

END Dept\_Head\_FName ;

Function Dept\_Head\_FName

( Dept\_name IN varchar2)

RETURN varchar2

IS

Dept\_Head\_fullname varchar2 (47);

BEGIN

. . ..

END Dept\_Head\_FName ;

BEGIN

. . .

END;

Q4.

REM Q4.

DECLARE

TYPE emp\_name IS RECORD (

f\_name varchar2(20),

l\_name varchar2(25)

);

emp1 emp\_name;

. . .

FUNCTION emp\_name\_eq ( rec1 IN emp\_name, rec2 IN emp\_name)

RETURN boolean

IS

ret boolean ;

Begin

IF ( ... ) THEN

ret := true;

else

ret := false;

END IF;

return ret;

END emp\_name\_eq;

BEGIN

...

IF emp\_name\_eq (emp1, emp2) THEN

DBMS\_OUTPUT.PUT\_LINE('The two records emp1 '''|| emp1.f\_name ||

' ' || emp1.l\_name ||''' and emp2 '''|| emp2.f\_name || ' ' ||

emp2.l\_name || ''' are same.' );

ELSE

DBMS\_OUTPUT.PUT\_LINE

END IF;

IF emp\_name\_eq (emp1, emp3) THEN

DBMS\_OUTPUT.PUT\_LINE

END IF;

END;

/