CEGEP VANIER COLLEGE
CENTRE FOR CONTINUING EDUCATION
Developing Applications using Oracle
420-987-VA

Teacher: Samir Chebbine **Lab 4** **Jul 19, 2022**

Lab 4: User-defined PL/SQL Procedures and PL/SQL Packages

Submit all PL/SQL programs and their output to be saved into a text file *Programs_Lab4Tutorial.sql* with *sql* extension.

1. PL/SQL Procedure

Complete all these following programs as explained in my **Lab 4 YouTube Video 1.** Notice all *missing* coding statements are provided in this video with explanation.

a) Definition of Procedure without Parameters (Header and Body).

The purpose of PL/SQL procedure and function is to *modularize* a PL/SQL program.
All previous PL/SQL programs are called **Anonymous procedures.**
Program PL/SQL # 1: Edit PL/SQL procedure *Display_Message* that displays date and a message.

The general syntax	Example of PL/SQL Procedure

CREATE [OR REPLACE] PROCEDURE Procedurename
[(param1, [param2, param3])]
IS
Declaration of constants, variables, cursors,
and exception
BEGIN
PL/SQL statements (if, while...)
and SQL statements
EXCEPTION
Action for error conditions
END;
	Oracle SQL*Plus
	File Edit Search Options Help
	SQL> [redacted]
	2 [redacted]
	3 today_date DATE; -- Date Data type
	4 w_hello VARCHAR2(30) := 'Hello World!'; -- Character Data Type
	5 BEGIN
	6 today_date := SYSDATE;
	7 [redacted]
	8 [redacted]
	9 END;
	10 /
	
	Procedure created.

- **Call of Procedure without Parameters.**

SQL> [redacted] My first message with PL/SQL Procedure: Hello World! Today's date is 26-DEC-11 PL/SQL procedure successfully completed.	Oracle SQL*Plus File Edit Search Options Help SQL> DECLARE 2 w_message VARCHAR2(30) := 'Call procedure!'; 3 BEGIN 4 [redacted] 5 END; 6 / My first message with PL/SQL Procedure: Hello World! Today's date is 26-DEC-11 PL/SQL procedure successfully completed.

b) Definition of Procedure with Parameters (Header and Body).

The purpose of PL/SQL procedure is to enhance *modularity and reusability* of PL/SQL program.

Program PL/SQL # 2:

Edit PL/SQL *procedure* called *Issue_Billing* that allows the end user to issue a billing for a given product (Prd_Price) and a given quantity (Prd_Qty). You have to display the total taking into account the federal and provincial taxes (Fed_Tax=7.5%, Prv_Tax=6% respectively) according to the following formula:

total_price = (Prd_Price* Prd_Qty) + (Prd_Price* Prd_Qty)* 7.5% + (Prd_Price* Prd_Qty)* 6%

Definition of Procedure with Parameters

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> ----------:----------
  2  ----------:----------
  3  --
  4    total_price       NUMBER(7,2);        -- Number Data type
  5    Fed_Tax NUMBER(5,3):=0.075;           -- Decimal Number Data type
  6    Prv_Tax NUMBER(5,3):=0.060;           -- Decimal Number Data type
  7  BEGIN
  8    total_price := (Prd_Price * Prd_Qty)
  9                 +  (Prd_Price* Prd_Qty)* Fed_Tax
 10                 +  (Prd_Price* Prd_Qty)* Prv_Tax;
 11    ----------:----------
 12    ----------:----------
 13    ----------:----------
 14  END;
 15  /

Procedure created.
```

Call of Procedure with Parameters

```
DECLARE
   Product_Price      NUMBER(7,2);
   Product_Qty        NUMBER(7);
   total_price        NUMBER(7,2);
BEGIN
    ----------:----------
    ----------:----------
    ----------:----------
END;
Enter value for prod_price: 49.99
Enter value for prod_qty: 2
The Entered Product Price is: 49.99
The Entered Product Quantity is: 2
The Total of Billing is: 113.48 $

PL/SQL procedure successfully completed.
```

2. PL/SQL Package

Remember *DBMS_OUTPUT* is a package, which includes *PUT_LINE* as a procedure. By **definition**, a package could contain objects such as Cursor, Procedure, Function, Scalar variables, and Composite variables.

The package contains a *specification* and *body* on how to call different procedures.

a) **The general syntax**

Package specification

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> ----------:----------
  2  AS
  3    ----------:----------
  4    ----------:----------
  5   (Prd_Price IN  NUMBER, Prd_Qty IN  NUMBER);
  6   END BillingPackage;
  7   /

Package created.
```

Package Body

```
Oracle SQL*Plus
File  Edit  Search  Options  Help
SQL> ----------:----------
  2  AS
  3  PROCEDURE ----------:----------
  4  ----------:----------
  5  BEGIN
  6  ----------:----------
  7
  8  END Display_Message;
  9  ------------------------------------------------------------
 10  PROCEDURE ----------:----------
 11   (Prd_Price IN  NUMBER, Prd_Qty IN  NUMBER)
 12
 13  ----------:----------
 14  BEGIN
 15  ----------:----------
 16  END Issue_Billing;
 17  ------------------------------------------------------------
 98  END BillingPackage;
 99  /

Package body created.
```

b) **Calling procedure and function from package**

You have to use dot notation in order to call any procedure specified in the package *BillingPackage*.

For example: *BillingPackage. Display_Message* or *BillingPackage.Issue_Billing (&price, &qty)*

3. **Execute the script file *Registration.sql* (Lab 2) for creating tables of Registration System.**

a) Create a Procedure to be named *Display_CategoryStudent* that accepts one formal parameter (*Cat_Stud_ID*) to display *Category Student description*, *start date*, and *end date* of given student category id found in *CategoryStudent* table.

Execute the procedure *Display_CategoryStudent* by calling it from an *anonymous* PL/SQL program as shown in Figure 1.

```
 16  END Display_CategoryStudent;
 17  /

Procedure created.

SQL> DECLARE
  2    [redacted]
  3  BEGIN
  4    [redacted]
  5    [redacted]
  6  END;
  7  /
Enter value for category_stud_id: 3
The Category Student description found is : Generation Y
Category Student Date : 01/JAN/1980
Category Student End Date : 31/DEC/1989

PL/SQL procedure successfully completed.
```

Figure 1

```
 11  END doCalc_Cost_Tuition;
 12  /

Procedure created.

SQL> DECLARE
  2    [redacted]
  3    [redacted]
  4    [redacted]
  5    [redacted]
  6  BEGIN
  7    [redacted]
  8    [redacted]
  9    [redacted]
 10    [redacted]
 11  END;
 12  /
Enter value for course_price: 120
Enter value for num_course: 3
Enter value for cost_manual: 250
The Total Cost of Tuition Corresponding to
Course Price: 120$
Number of Courses: 3
Cost Manual: 250
is: 610$

PL/SQL procedure successfully completed.
```

Figure 2

b) Create a Procedure to be named *doCalc_Cost_Tuition* that accepts three arguments (*course_price*, *num_course*,*cost_manual*) and computes the cost of a tuition (*cost_work*) according to the following formula:
*cost_tuition = (course_price * num_course) + cost_manual*
Execute the procedure *doCalc_Cost_Tuition* by calling it from an *anonymous* PL/SQL program as shown in Figure 2.

c) Create a package to be named *RegistrationPackage* that contains all previous procedures (*Display_CategoryStudent*, *doCalc_Cost_Tuition*) as shown in Figure 3 and 4.

Package specification	Package Body	Package Procedure call
7 END RegistrationPackage; 8 / Package created.	60 END RegistrationPackage; 61 / Package body created.	SQL> BEGIN 2 RegistrationPackage.Display_CategoryStudent [redacted] 3 RegistrationPackage.[redacted] 4 END; 5 / Enter value for category_stud_id: 2 Enter value for course_price: 120 Enter value for num_course: 3 Enter value for cost_manual: 250 The Category Student description found is : Generation X Category Student Date : 01/JAN/1970 Category Student End Date : 31/DEC/1979 The Total Cost of Tuition Corresponding to Course Price: 120$ Number of Courses: 3 Cost Manual: 250 is: 610$ PL/SQL procedure successfully completed.
Figure 3	Figure 4	Figure 5

d) Call procedures, *Display_CategoryStudent*, *doCalc_Cost_Tuition* in order from the package

RegistrationPackage as shown in Figure 5.

4. Review Questions

A. Write necessary PL/SQL statements to create the following components:

1. A heading of PL/SQL procedure named *Calculate_ProjectContribution* with two parameters *Project_Name* of type *varchar2(30)* and *Project_SDate* of type *Date()*.

2. A PL/SQL package specification named *ProjectPackage* which contains the named procedure *Calculate_ProjectContribution*.

3. A variable cursor named *CategoryEmployee_row* of type *CategoryEmployee_cursor* to reference a given record.

4. Declare a cursor named *course_cursor* that self-join a table *course* (of *Registration* script) to display *course names* and its *course pre-requisites*.

5. Declare variable named *vsalary* of the same type as field *Salary* from *employee* table.

6. A prompt statement to input a value of salary assigned to previous variable *vsalary*.

7. Declare a cursor named *studentgrade_cursor* that displays student information (s_last, s_first, s_class, birthday) and their grades (from *Registration* script).

8. Declare a cursor named *DeptFacultyStudent_cursor* that displays department information (DeptId, DeptName, Location) and its faculty members along with their supervised students (from *Registration* script).

B. Multiple choice (only one answer per question is valid)

1. The user-defined PL/SQL package is used to
   a. reduce the number of statements b. insert SQL query
   c. enhance reusability d. show error

2. An advantage of declaring user-defined PL/SQL procedure is to
   a. create modular program b. fasten the program execution
   c. ease the debugging of program d. all of the above

3. A user-defined PL/SQL procedure refers to
   a. query tables b. action to be executed
   c. using cursors d. fields to be used

4. A user-defined PL/SQL package may contain ____
   a. only one procedure b. at least two procedures
   c. many procedures d. only scalar variables

5. An explicit cursor is used to fetch ____record(s)
   a. one b. at least two
   c. variable d. multiple

6. An anchored variable uses key word ____
   a. TYPE b. CURSOR
   c. ROWTYPE d. FETCH