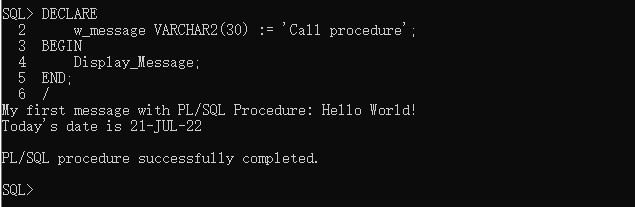
Lab 4: User-defined PL/SQL Procedures and PL/SQL Packages

1. PL/SQL Procedure
2. Definition of Procedure without Parameters (Header and Body).
3. CREATE OR REPLACE PROCEDURE Display\_Message
4. IS
5. today\_date      DATE;                           *-- Date Data type*
6. w\_hello         VARCHAR2(30) := 'Hello World!'; *-- Character Data type*
7. BEGIN
8. today\_date := SYSDATE;
9. dbms\_output.put\_line('My first message with PL/SQL Procedure: '||w\_hello);
10. dbms\_output.put\_line('Today''s date is ' || today\_date);
11. END;



• Call of Procedure without Parameters.

1. DECLARE
2. w\_message VARCHAR2(30) := 'Call procedure';
3. BEGIN
4. Display\_Message;
5. END;



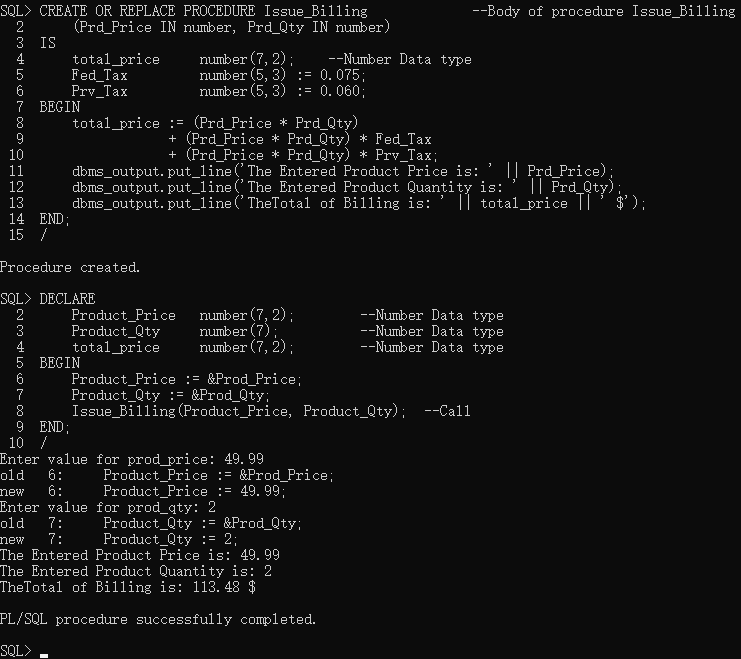
b) Definition of Procedure with Parameters (Header and Body)

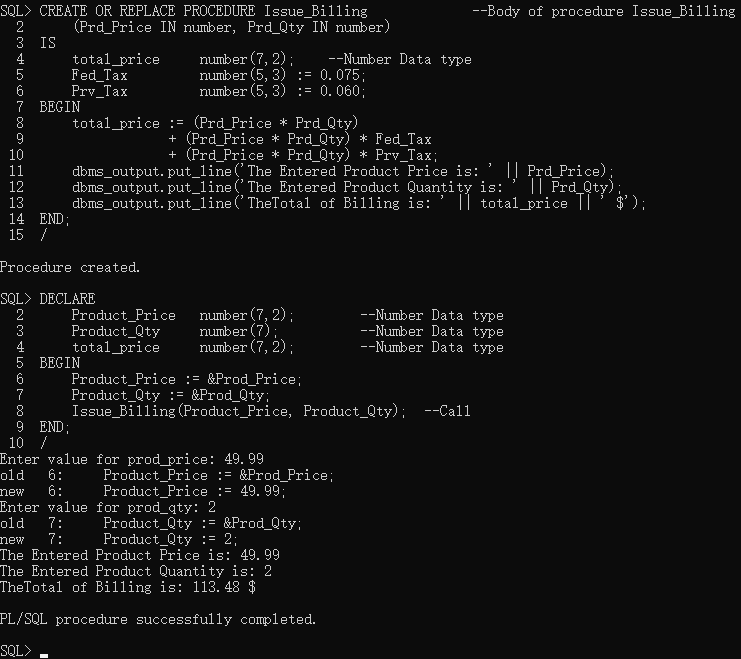
Definition of Procedure with Parameters

1. CREATE OR REPLACE PROCEDURE Issue\_Billing    *--Body of procedure Issue\_Billing*
2. (Prd\_Price IN number, Prd\_Qty IN number)
3. IS
4. total\_price     number(7,2);    *--Number Data type*
5. Fed\_Tax         number(5,3) := 0.075;
6. Prv\_Tax         number(5,3) := 0.060;
7. BEGIN
8. total\_price := (Prd\_Price \* Prd\_Qty)
9. + (Prd\_Price \* Prd\_Qty) \* Fed\_Tax
10. + (Prd\_Price \* Prd\_Qty) \* Prv\_Tax;
11. dbms\_output.put\_line('The Entered Product Price is: ' || Prd\_Price);
12. dbms\_output.put\_line('The Entered Product Quantity is: ' || Prd\_Qty);
13. dbms\_output.put\_line('TheTotal of Billing is: ' || total\_price || ' $');
14. END;

Call of Procedure with Parameters

1. DECLARE
2. Product\_Price   number(7,2);        *--Number Data type*
3. Product\_Qty     number(7);          *--Number Data type*
4. total\_price     number(7,2);        *--Number Data type*
5. BEGIN
6. Product\_Price := &Prod\_Price;
7. Product\_Qty := &Prod\_Qty;
8. Issue\_Billing(Product\_Price, Product\_Qty);  *--Call*
9. END;





2. PL/SQL Package

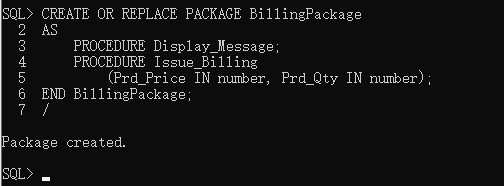
a) The general syntax

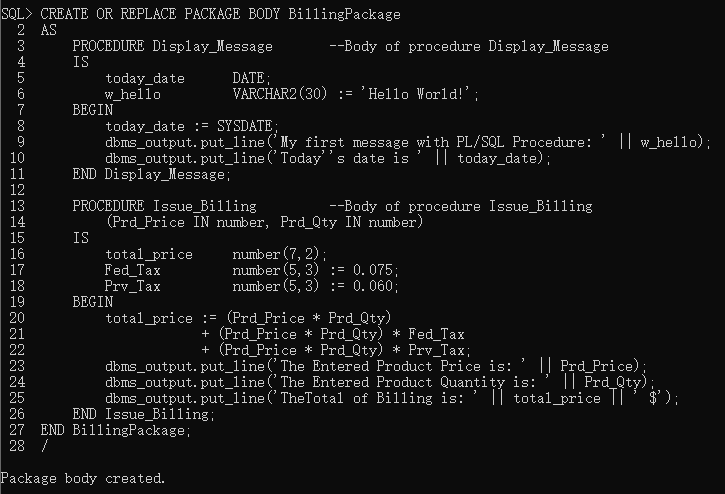
Package specification

1. CREATE OR REPLACE PACKAGE BillingPackage
2. AS
3. PROCEDURE Display\_Message;
4. PROCEDURE Issue\_Billing
5. (Prd\_Price IN number, Prd\_Qty IN number);
6. END BillingPackage;

Package Body

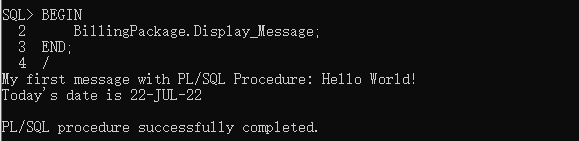
1. CREATE OR REPLACE PACKAGE BODY BillingPackage
2. AS
3. PROCEDURE Display\_Message       *--Body of procedure Display\_Message*
4. IS
5. today\_date      DATE;
6. w\_hello         VARCHAR2(30) := 'Hello World!';
7. BEGIN
8. today\_date := SYSDATE;
9. dbms\_output.put\_line('My first message with PL/SQL Procedure: ' || w\_hello);
10. dbms\_output.put\_line('Today''s date is ' || today\_date);
11. END Display\_Message;
13. PROCEDURE Issue\_Billing         --Body of procedure Issue\_Billing
14. (Prd\_Price IN number, Prd\_Qty IN number)
15. IS
16. total\_price     number(7,2);
17. Fed\_Tax         number(5,3) := 0.075;
18. Prv\_Tax         number(5,3) := 0.060;
19. BEGIN
20. total\_price := (Prd\_Price \* Prd\_Qty)
21. + (Prd\_Price \* Prd\_Qty) \* Fed\_Tax
22. + (Prd\_Price \* Prd\_Qty) \* Prv\_Tax;
23. dbms\_output.put\_line('The Entered Product Price is: ' || Prd\_Price);
24. dbms\_output.put\_line('The Entered Product Quantity is: ' || Prd\_Qty);
25. dbms\_output.put\_line('TheTotal of Billing is: ' || total\_price || ' $');
26. END Issue\_Billing;
27. END BillingPackage;



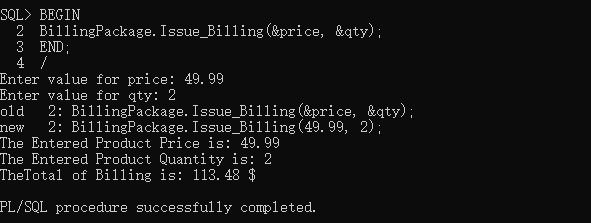


b) Calling procedure and function from package

1. BEGIN
2. BillingPackage.Display\_Message;
3. END;



1. BEGIN
2. BillingPackage.Issue\_Billing(&price, &qty);
3. END;



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

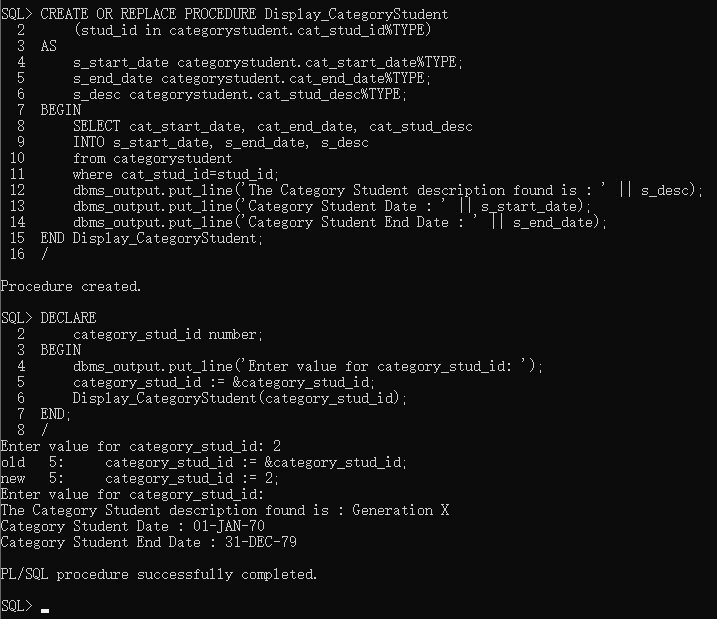
a) Create a Procedure to be named Display\_CategoryStudent that accepts one formal parameter(Cat\_Stud\_ID) to display Category Student description, start date, and end date of givenstudent category id found in CategoryStudent table.

Procedure Display\_CategoryStudent

1. CREATE OR REPLACE PROCEDURE Display\_CategoryStudent
2. (stud\_id in categorystudent.cat\_stud\_id%TYPE)
3. AS
4. s\_start\_date categorystudent.cat\_start\_date%TYPE;
5. s\_end\_date categorystudent.cat\_end\_date%TYPE;
6. s\_desc categorystudent.cat\_stud\_desc%TYPE;
7. BEGIN
8. SELECT cat\_start\_date, cat\_end\_date, cat\_stud\_desc
9. INTO s\_start\_date, s\_end\_date, s\_desc
10. from categorystudent
11. where cat\_stud\_id=stud\_id;
12. dbms\_output.put\_line('The Category Student description found is : ' || s\_desc);
13. dbms\_output.put\_line('Category Student Date : ' || s\_start\_date);
14. dbms\_output.put\_line('Category Student End Date : ' || s\_end\_date);
15. END Display\_CategoryStudent;

Calling Procedure

1. DECLARE
2. category\_stud\_id number;
3. BEGIN
4. category\_stud\_id := &category\_stud\_id;
5. Display\_CategoryStudent(category\_stud\_id);
6. END;



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

Procedure doCalc\_Cost\_Tuition

1. CREATE OR REPLACE PROCEDURE doCalc\_Cost\_Tuition
2. (course\_price IN number, num\_course IN number,
3. cost\_manual IN number,
4. cost\_tuition OUT number)
5. AS
6. BEGIN
7. cost\_tuition := (course\_price \* num\_course) + cost\_manual;
8. END doCalc\_Cost\_Tuition;

Calling Procedure

1. DECLARE
2. course\_price    number;
3. num\_course      number;
4. cost\_manual     number;
5. cost\_tuition    number;
6. BEGIN
7. course\_price := &course\_price;
8. num\_course := &num\_course;
9. cost\_manual := &cost\_manual;
11. doCalc\_Cost\_Tuition(course\_price, num\_course, cost\_manual, cost\_tuition);
12. dbms\_output.put\_line('The Total Cost of Tuition Corresponding to');
13. dbms\_output.put\_line('Course Price: ' || course\_price || '$');
14. dbms\_output.put\_line('Number of Courses: ' || num\_course);
15. dbms\_output.put\_line('Cost Manual: ' || cost\_manual);
16. dbms\_output.put\_line('is: ' || cost\_tuition || '$');
17. END;



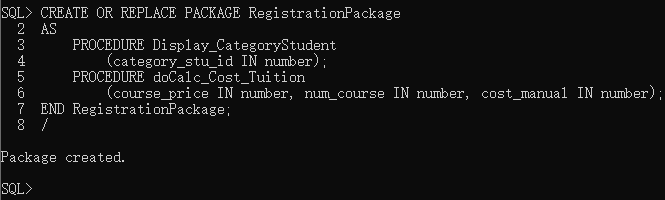
c) Create a package to be named RegistrationPackage that contains all previous procedures(Display\_CategoryStudent, doCalc\_Cost\_Tuition)

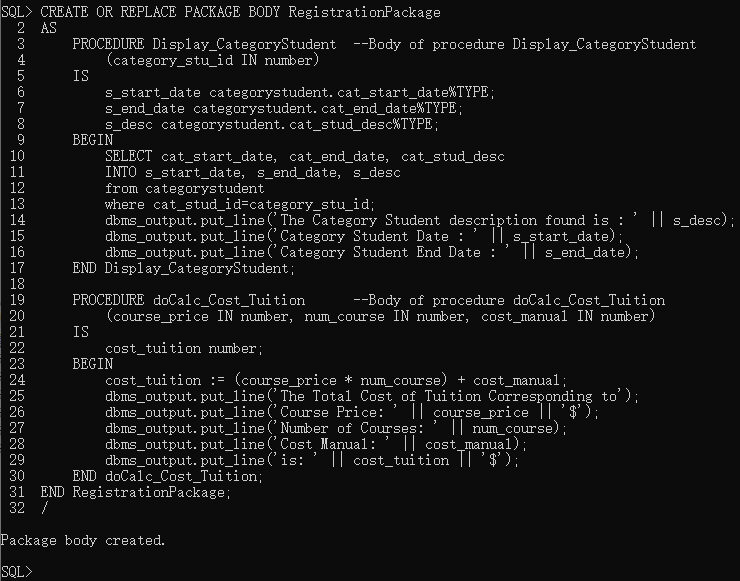
Package specification

1. **CREATE** OR REPLACE PACKAGE RegistrationPackage
2. **AS**
3. **PROCEDURE** Display\_CategoryStudent
4. (category\_stu\_id IN number);
5. **PROCEDURE** doCalc\_Cost\_Tuition
6. (course\_price IN number, num\_course IN number, cost\_manual IN number);
7. **END** RegistrationPackage;

Package Body

1. **CREATE** OR REPLACE PACKAGE BODY RegistrationPackage
2. **AS**
3. **PROCEDURE** Display\_CategoryStudent  --Body of procedure Display\_CategoryStudent
4. (category\_stu\_id IN number)
5. **IS**
6. s\_start\_date categorystudent.cat\_start\_date%TYPE;
7. s\_end\_date categorystudent.cat\_end\_date%TYPE;
8. s\_desc categorystudent.cat\_stud\_desc%TYPE;
9. **BEGIN**
10. **SELECT** cat\_start\_date, cat\_end\_date, cat\_stud\_desc
11. **INTO** s\_start\_date, s\_end\_date, s\_desc
12. **from** categorystudent
13. **where** cat\_stud\_id=category\_stu\_id;
14. dbms\_output.put\_line('The Category Student description found is : ' || s\_desc);
15. dbms\_output.put\_line('Category Student Date : ' || s\_start\_date);
16. dbms\_output.put\_line('Category Student End Date : ' || s\_end\_date);
17. **END** Display\_CategoryStudent;
18. **PROCEDURE** doCalc\_Cost\_Tuition      --Body of procedure doCalc\_Cost\_Tuition
19. (course\_price IN number, num\_course IN number, cost\_manual IN number)
20. **IS**
21. cost\_tuition number;
22. **BEGIN**
23. cost\_tuition := (course\_price \* num\_course) + cost\_manual;
24. dbms\_output.put\_line('The Total Cost of Tuition Corresponding to');
25. dbms\_output.put\_line('Course Price: ' || course\_price || '$');
26. dbms\_output.put\_line('Number of Courses: ' || num\_course);
27. dbms\_output.put\_line('Cost Manual: ' || cost\_manual);
28. dbms\_output.put\_line('is: ' || cost\_tuition || '$');
29. **END** doCalc\_Cost\_Tuition;
30. **END** RegistrationPackage;

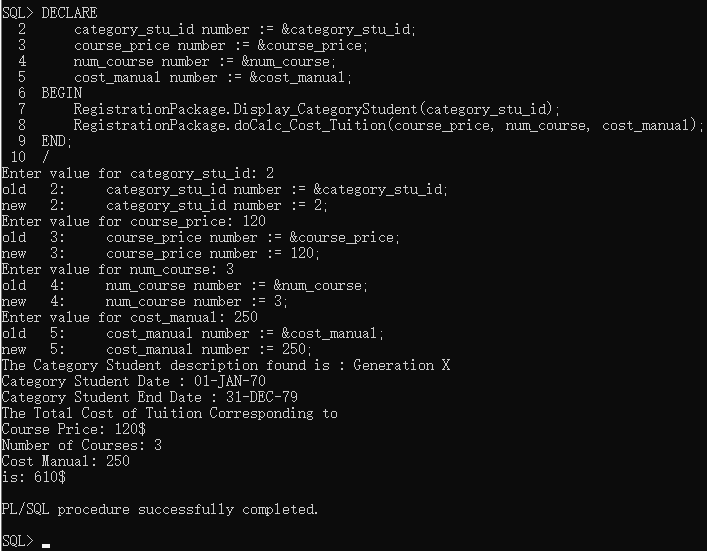




d) Call procedures, Display\_CategoryStudent, doCalc\_Cost\_Tuition in order from the package

Calling Procedure

1. **DECLARE**
2. category\_stu\_id number := &category\_stu\_id;
3. course\_price number := &course\_price;
4. num\_course number := &num\_course;
5. cost\_manual number := &cost\_manual;
6. **BEGIN**
7. RegistrationPackage.Display\_CategoryStudent(category\_stu\_id);
8. RegistrationPackage.doCalc\_Cost\_Tuition(course\_price, num\_course, cost\_manual);
9. **END**;



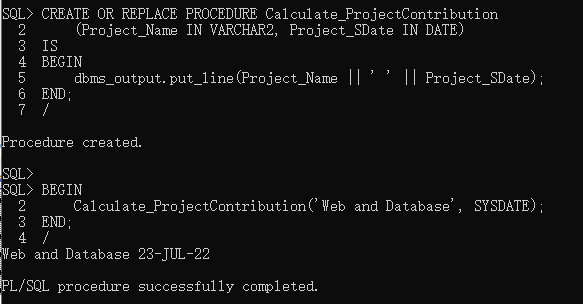
4. Review Questions

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

1) A heading of PL/SQL procedure named **Calculate\_ProjectContribution** with twoparameters **Project\_Name** of type **varchar2(30)** and **Project\_SDate** of type **Date().**

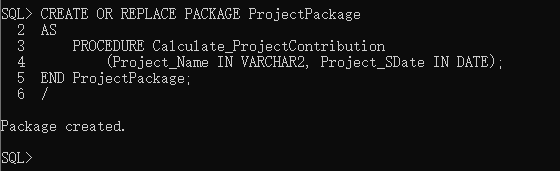
1. **CREATE** OR REPLACE **PROCEDURE** Calculate\_ProjectContribution
2. (Project\_Name IN VARCHAR2, Project\_SDate IN **DATE**)
3. **IS**
4. **BEGIN**
5. dbms\_output.put\_line(Project\_Name || ' ' || Project\_SDate);
6. **END**;

Tips: Parameters cannot be declared specific length in Oracle procedure, the correct way is to not specify the length of the parameter.



2) A PL/SQL package specification named **ProjectPackage** which contains the namedprocedure **Calculate\_ProjectContribution**.

1. **CREATE** OR REPLACE PACKAGE ProjectPackage
2. **AS**
3. **PROCEDURE** Calculate\_ProjectContribution
4. (Project\_Name IN VARCHAR2, Project\_SDate IN **DATE**);
5. **END** ProjectPackage;

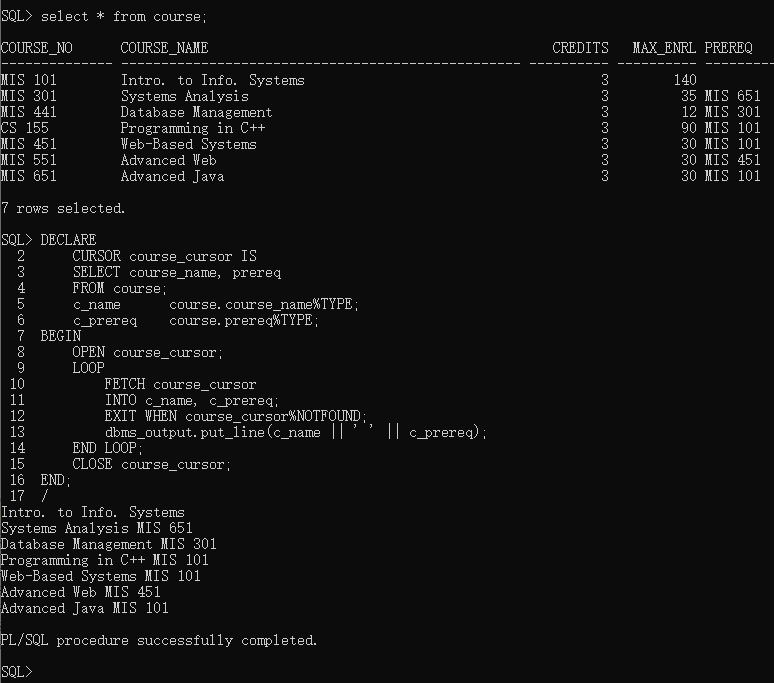


3) A variable cursor named **CategoryEmployee\_row** of type **CategoryEmployee\_cursor**to reference a given record.

1. **DECLARE**
2. CategoryEmployee\_row CategoryEmployee\_cursor%ROWTYPE;
3. **BEGIN**
4. **END**;

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**.

1. **DECLARE**
2. **CURSOR** course\_cursor **IS**
3. **SELECT** course\_name, prereq
4. **FROM** course;
5. **BEGIN**
6. **OPEN** course\_cursor;
7. **CLOSE** course\_cursor;
8. **END**;



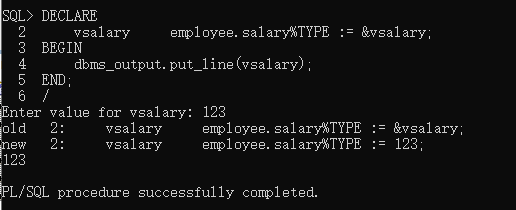
5) Declare variable named **vsalary** of the same type as field **Salary** from **employee** table.

1. **DECLARE**
2. vsalary     employee.salary%TYPE;

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

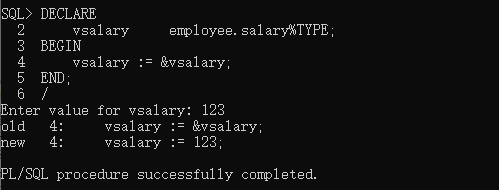
**Solution I**

1. **DECLARE**
2. vsalary     employee.salary%TYPE := &vsalary;
3. **BEGIN**
4. dbms\_output.put\_line(vsalary);
5. **END**;



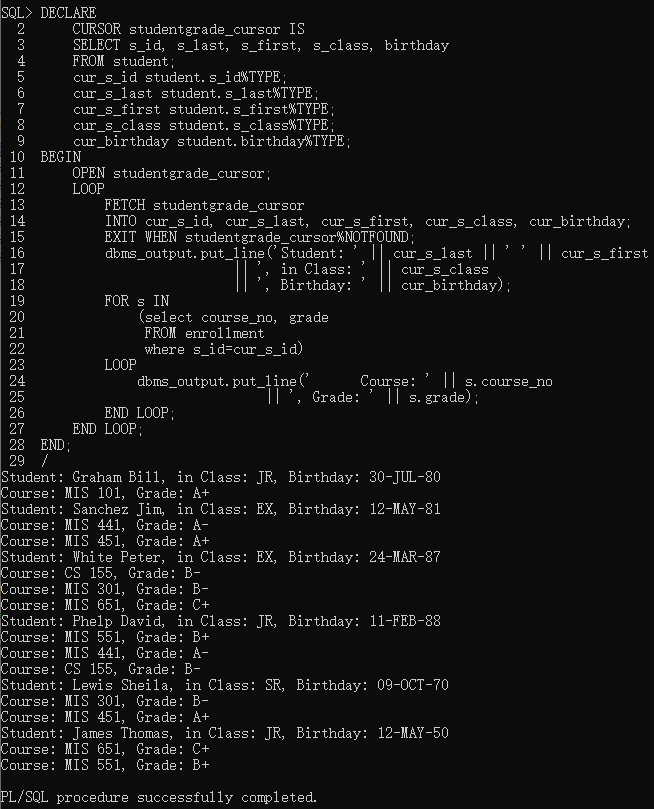
**Solution II**

1. **DECLARE**
2. vsalary     employee.salary%TYPE;
3. **BEGIN**
4. vsalary := &vsalary;
5. **END**;



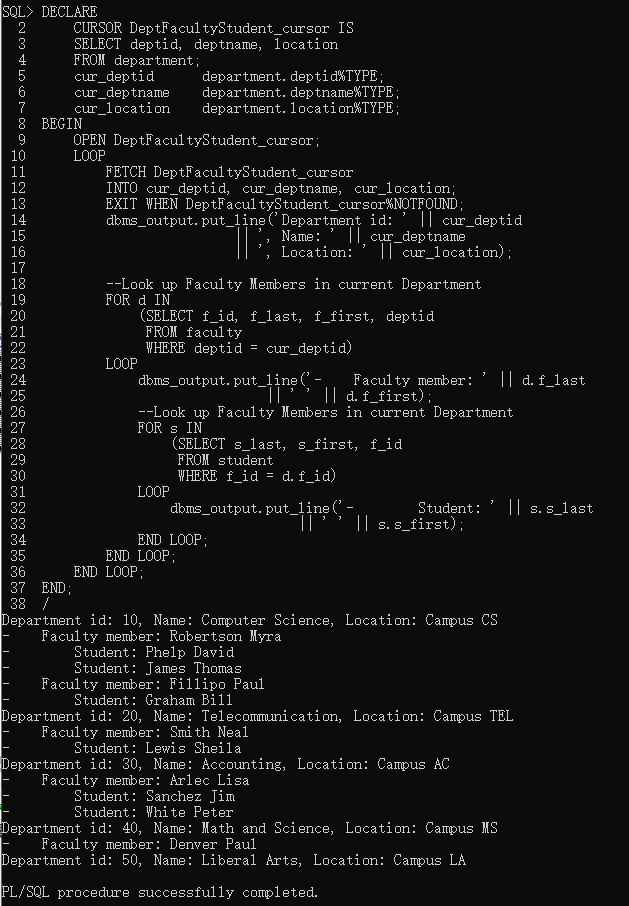
7) Declare a cursor named **studentgrade\_cursor** that displays student information (s\_last,s\_first, s\_class, birthday) and their grades (from **Registration** script).

1. **DECLARE**
2. **CURSOR** studentgrade\_cursor **IS**
3. **SELECT** s\_id, s\_last, s\_first, s\_class, birthday
4. **FROM** student;
5. cur\_s\_id student.s\_id%TYPE;
6. cur\_s\_last student.s\_last%TYPE;
7. cur\_s\_first student.s\_first%TYPE;
8. cur\_s\_class student.s\_class%TYPE;
9. cur\_birthday student.birthday%TYPE;
10. **BEGIN**
11. **OPEN** studentgrade\_cursor;
12. LOOP
13. **FETCH** studentgrade\_cursor
14. **INTO** cur\_s\_id, cur\_s\_last, cur\_s\_first, cur\_s\_class, cur\_birthday;
15. EXIT **WHEN** studentgrade\_cursor%NOTFOUND;
16. dbms\_output.put\_line('Student: ' || cur\_s\_last || ' ' || cur\_s\_first
17. || ', in Class: ' || cur\_s\_class
18. || ', Birthday: ' || cur\_birthday);
19. **FOR** s IN
20. (**select** course\_no, grade
21. **FROM** enrollment
22. **where** s\_id=cur\_s\_id)
23. LOOP
24. dbms\_output.put\_line('      Course: ' || s.course\_no
25. || ', Grade: ' || s.grade);
26. **END** LOOP;
27. **END** LOOP;
28. **END**;



8) Declare a cursor named **DeptFacultyStudent\_cursor** that displays departmentinformation (DeptId, DeptName, Location) and its faculty members along with theirsupervised students (from **Registration** script).

1. **DECLARE**
2. **CURSOR** DeptFacultyStudent\_cursor **IS**
3. **SELECT** deptid, deptname, location
4. **FROM** department;
5. cur\_deptid      department.deptid%TYPE;
6. cur\_deptname    department.deptname%TYPE;
7. cur\_location    department.location%TYPE;
8. **BEGIN**
9. **OPEN** DeptFacultyStudent\_cursor;
10. LOOP
11. **FETCH** DeptFacultyStudent\_cursor
12. **INTO** cur\_deptid, cur\_deptname, cur\_location;
13. EXIT **WHEN** DeptFacultyStudent\_cursor%NOTFOUND;
14. dbms\_output.put\_line('Department id: ' || cur\_deptid
15. || ', Name: ' || cur\_deptname
16. || ', Location: ' || cur\_location);
18. --Look up Faculty Members in current Department
19. **FOR** d IN
20. (**SELECT** f\_id, f\_last, f\_first, deptid
21. **FROM** faculty
22. **WHERE** deptid = cur\_deptid)
23. LOOP
24. dbms\_output.put\_line('      Faculty member: ' || d.f\_last
25. || ' ' || d.f\_first);
26. --Look up Faculty Members in current Department
27. **FOR** s IN
28. (**SELECT** s\_last, s\_first, f\_id
29. **FROM** student
30. **WHERE** f\_id = d.f\_id)
31. LOOP
32. dbms\_output.put\_line('            Student: ' || s.s\_last
33. || ' ' || s.s\_first);
34. **END** LOOP;
35. **END** LOOP;
36. **END** LOOP;
37. **END**;



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

**C** 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

**D** 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

**B** 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

**C** 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

**D** 5. An explicit cursor is used to fetch \_\_\_\_\_record(s)

a. one b. at least two

c. variable d. multiple

**A** 6. An anchored variable uses key word \_\_\_\_\_

a. TYPE b. CURSOR

c. ROWTYPE d. FETCH