ORACLE
Academy

academy.oracle.com

Database Programming with PL/SQL
8-1: Creating Procedures
Practice Activities
Vocabulary

Identify the vocabulary word for each definition below:

PL/SQL subprograms	Named PL/SQL blocks that are compiled and stored in the database.
Identifier	Indicates the DECLARE section of a subprogram.
Anonymous Block	Unnamed executable PL/SQL blocks that cannot be reused or stored in the database for later use.
Procedures	Named PL/SQL blocks that can accept parameters and are compiled and stored in the database.

Try It / Solve It

1. What is the difference between the following two pieces of code?

CODE SAMPLE A

```
DECLARE
  v_empid                employees.employee_id%TYPE := 100;
  v_percent_increase     NUMBER(2,2) := .05;
BEGIN
  UPDATE employees
    SET salary = (salary * v_percent_increase) + salary
    WHERE employee_id = v_empid;
END;
```

CODE SAMPLE B

```
CREATE PROCEDURE pay_raise
     (p_empid employees          employee_id%TYPE,
      p_percent_increase         NUMBER)
IS
BEGIN
  UPDATE employees
    SET salary = (salary * p_percent_increase) + salary
    WHERE employee_id = p_empid;
END pay_raise;
```

Code sample B is a procedure, code sample A is an anonymous block.

The procedure can be reused in other statements while code sample A can not be called or reused

2. In your own words, list the benefits of subprograms.

You can store the subprogram in the database, reuse it in other subprograms, if you want to change something you just change it in the procedure and it's changed everywhere it's called, meaning you have to change and test less, it's easier to read.

3. In your own words, describe a stored procedure.

A stored procedure is a bunch of sql statements that are given a name and can be called from other programs.

4. The remaining questions in this practice use a copy of the employees table. Create the copy by executing the following SQL statement:

```
CREATE TABLE employees_dup AS SELECT * from employees;
```

   A. Use the code below to create a procedure in Application Express. Save the definition of your procedure in case you need to modify it later. In the “Save SQL” popup, name your saved work “My name change procedure.”

```
CREATE OR REPLACE PROCEDURE name_change IS
BEGIN
  UPDATE employees_dup
    SET first_name = 'Susan'
    WHERE department_id = 80;
END name_change;
```

   B. Execute the procedure by running the following anonymous block:

```
BEGIN
  name_change;
END;
```

   C. SELECT from the table to check that the procedure has executed correctly and performed the UPDATE.

```
Select *
FROM employees_dup
WHERE department_id = 80;
```

5. Create a second procedure named pay_raise which changes the salary of all employees in employees_dup to a new value of 30000. Execute the procedure from an anonymous block, then SELECT from the table to check that the procedure has executed correctly.

6. etrieve your first name_change procedure by clicking on its name in the Saved SQL window. Modify the code to remove OR REPLACE from the CREATE statement, and introduce a deliberate error into the code, for example by misspelling a keyword: UPDAT employees_dup. Execute your code to recreate the procedure. What happens?

Name is already used.

7. Now correct the procedure code by reinserting the OR REPLACE clause and correcting your deliberate spelling error. Execute your code to recreate the procedure. Now what happens?

Procedure created.

8. Create, save, and execute a procedure which updates the salary of employees in employees_dup according to the following rules:

   - if the employee is in department 80, the new salary = 1000
   - if the employee is in department 50, the new salary = 2000
   - if the employee is in any other department, the new salary = 3000.

   You will need to include three UPDATE statements, one for each of the above rules. In a later lesson you will learn how to avoid this. Execute your procedure from an anonymous block and verify that the updates have been performed correctly.

5.

```
CREATE OR REPLACE PROCEDURE pay_raise IS
BEGIN
UPDATE employees_dup
SET salary = '30000';
END pay_raise;

BEGIN
pay_raise;
END;


SELECT * FROM employees_dup;
```

8.

```
CREATE OR REPLACE PROCEDURE new_sal IS
BEGIN
UPDATE employees_dup
SET salary = 1000
WHERE department_id = 80;
UPDATE employees_dup
SET salary = 2000
WHERE department_id = 50;
UPDATE employees_dup
SET salary = 3000
WHERE department_id NOT IN (80,50);
END new_sal;


BEGIN
new_sal;
END;


SELECT salary, department_id
FROM employees_dup;
```