10

# Stored Procedures and Functions

# Objectives

After completing this lesson, you should be able to:

- Differentiate between anonymous blocks and subprograms
- Create a simple procedure and invoke it from an anonymous block
- Create a simple function
- Create a simple function that accepts a parameter
- Differentiate between procedures and functions

# Course Roadmap

PL SQL

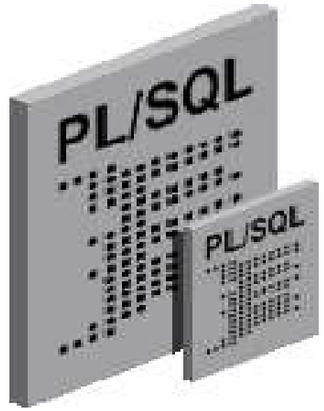
- Lesson 6: Writing Control Statements
- Lesson 7: Working with Composite DataTypes
- Lesson 8: Using Explicit Cursors
- Lesson 9: Exception Handling
- **Lesson 10: Stored Procedures and Functions** — You are here!

# Agenda

- Introducing procedures and functions
- Previewing procedures
- Previewing functions

# Procedures and Functions

- Are named PL/SQL blocks
- Are called PL/SQL subprograms
- Have block structures similar to anonymous blocks:
  - Optional declarative section (without the `DECLARE` keyword)
  - Mandatory executable section
  - Optional section to handle exceptions



# Differences Between Anonymous  Blocks and Subprograms

| Anonymous Blocks | Subprograms |
|---|---|
| Unnamed PL/SQL blocks | Named PL/SQL blocks |
| Compiled every time | Compiled only once |
| Not stored in the database | Stored in the database |
| Cannot be invoked by other applications | Named and, therefore, can be invoked by other applications |
| Do not return values | If functions, must return values |
| Cannot take parameters | Can take parameters |

# Agenda

- Introducing procedures and functions
- Previewing procedures
- Previewing functions

# Procedure: Syntax

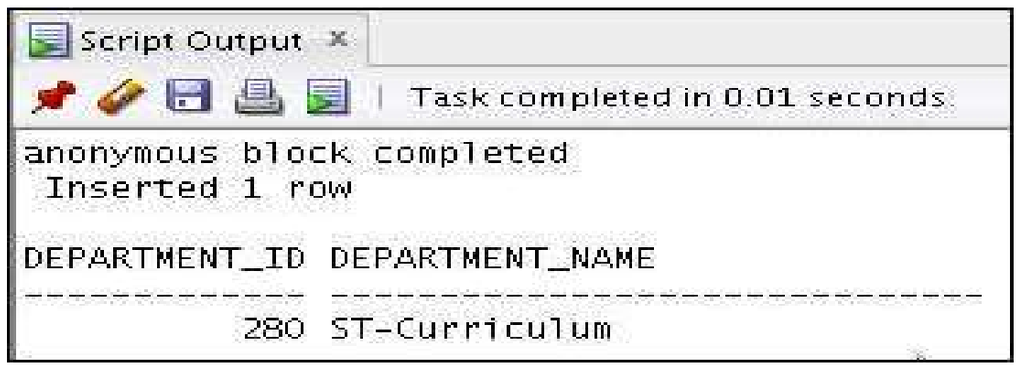
```
CREATE [OR REPLACE] PROCEDURE procedure_name
 [(argument1 [mode1] datatype1,
  argument2 [mode2] datatype2,
  . . .)]
IS|AS
procedure_body;
```

# Creating a Procedure

```
...
CREATE TABLE dept AS SELECT * FROM departments;
CREATE PROCEDURE add_dept IS
 v_dept_id dept.department_id%TYPE;
 v_dept_name dept.department_name%TYPE;
BEGIN
 v_dept_id:=280;
 v_dept_name:='ST-Curriculum';
 INSERT INTO dept(department_id,department_name)
 VALUES(v_dept_id,v_dept_name);
 DBMS_OUTPUT.PUT_LINE(' Inserted '|| SQL%ROWCOUNT ||' row ');
END;
```

# Invoking a Procedure

```
...
BEGIN
 add_dept;
END;
/
SELECT department_id, department_name FROM dept WHERE
department_id=280;
```



# Agenda

- Introducing procedures and functions
- Previewing procedures
- Previewing functions

# Function: Syntax

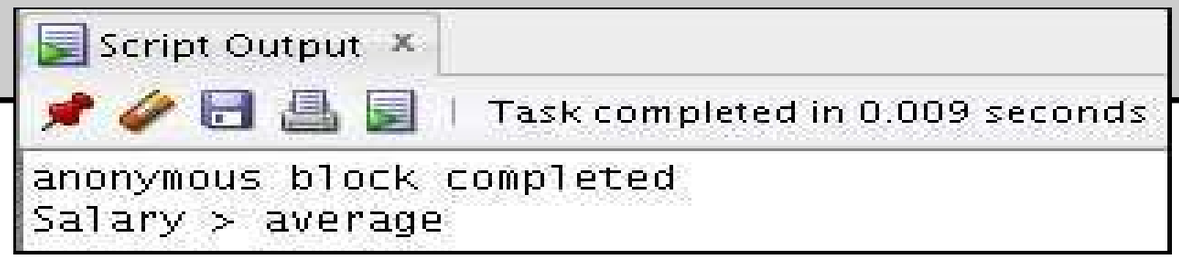
```
CREATE [OR REPLACE] FUNCTION function_name
 [(argument1 [mode1] datatype1,
  argument2 [mode2] datatype2,
  . . .)]
RETURN datatype
IS|AS
function_body;
```

# Creating a Function

```
CREATE FUNCTION check_sal RETURN Boolean IS
v_dept_id employees.department_id%TYPE;
 v_empno   employees.employee_id%TYPE;
 v_sal     employees.salary%TYPE;
 v_avg_sal employees.salary%TYPE;
BEGIN
 v_empno:=205;
 SELECT salary,department_id INTO v_sal,v_dept_id FROM employees
 WHERE employee_id= v_empno;
 SELECT avg(salary) INTO v_avg_sal FROM employees WHERE department_id=v_dept_id;
 IF v_sal > v_avg_sal THEN
  RETURN TRUE;
 ELSE
  RETURN FALSE;
 END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
   RETURN NULL;
END;
```

# Invoking a Function

```
BEGIN
 IF (check_sal IS NULL) THEN
 DBMS_OUTPUT.PUT_LINE('The function returned
  NULL due to exception');
 ELSIF (check_sal) THEN
 DBMS_OUTPUT.PUT_LINE('Salary > average');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Salary < average');
 END IF;
END;
/
```



# Passing a Parameter to the Function

```
DROP FUNCTION check_sal;
CREATE FUNCTION check_sal(p_empno employees.employee_id%TYPE) RETURN Boolean IS
 v_dept_id employees.department_id%TYPE;
 v_sal     employees.salary%TYPE;
 v_avg_sal employees.salary%TYPE;
BEGIN
 SELECT salary,department_id INTO v_sal,v_dept_id FROM employees
   WHERE employee_id=p_empno;
 SELECT avg(salary) INTO v_avg_sal FROM employees
   WHERE department_id=v_dept_id;
 IF v_sal > v_avg_sal THEN
  RETURN TRUE;
 ELSE
  RETURN FALSE;
 END IF;
EXCEPTION
  ...
```

# Invoking the Function with a Parameter

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Checking for employee with id 205');
 IF (check_sal(205) IS NULL) THEN
 DBMS_OUTPUT.PUT_LINE('The function returned
  NULL due to exception');
 ELSIF (check_sal(205)) THEN
 DBMS_OUTPUT.PUT_LINE('Salary > average');
 ELSE
 DBMS_OUTPUT.PUT_LINE('Salary < average');
 END IF;
DBMS_OUTPUT.PUT_LINE('Checking for employee with id 70');
 IF (check_sal(70) IS NULL) THEN
 DBMS_OUTPUT.PUT_LINE('The function returned
  NULL due to exception');
 ELSIF (check_sal(70)) THEN
 ...
 END IF;
END;
/
```

Subprograms:

a. Are named PL/SQL blocks and can be invoked by other applications
b. Are compiled only once
c. Are stored in the database
d. Do not have to return values if they are functions
e. Can take parameters

# Summary

In this lesson, you should have learned that:

- Create a simple procedure
- Invoke the procedure from an anonymous block
- Create a simple function
- Create a simple function that accepts parameters
- Invoke the function from an anonymous block

# Practice 10: Overview

This practice covers the following topics:

- Converting an existing anonymous block to a procedure
- Modifying the procedure to accept a parameter
- Writing an anonymous block to invoke the procedure