Lesson 11

Creating Functions

Advantages of User-Defined Functions in SQL Statements

- Can extend SQL where activities are too complex, too awkward, or unavailable with SQL
- Can increase efficiency when used in the `WHERE` clause to filter data, as opposed to filtering the data in the application
- Can manipulate data values

Creating Functions: Syntax

The PL/SQL block must have at least one RETURN statement.

```
CREATE [OR REPLACE] FUNCTION function_name
 [(parameter1 [mode1] datatype1, . . .)]
RETURN datatype IS|AS
 [local_variable_declarations;
  . . .]
BEGIN
  -- actions;
  RETURN expression;
END [function_name];
```

PL/SQL Block

Creating and Invoking a Stored Function Using the CREATE FUNCTION Statement: Example

```
CREATE OR REPLACE FUNCTION get_sal
 (p_id employees.employee_id%TYPE) RETURN NUMBER IS
  v_sal employees.salary%TYPE := 0;
  BEGIN
    SELECT salary
    INTO   v_sal
    FROM   employees
    WHERE  employee_id = p_id;
    RETURN v_sal;
  END get_sal;
/
```

```
FUNCTION get_sal Compiled.
```

```
-- Invoke the function as an expression or as
-- a parameter value.

EXECUTE dbms_output.put_line(get_sal(100))
```

```
anonymous block completed
24000
```

Using Different Methods for Executing Functions

```
-- As a PL/SQL expression, get the results using host variables

VARIABLE b_salary NUMBER
EXECUTE :b_salary := get_sal(100)
```

```
anonymous block completed
b_salary
-----
24000
```

```
-- As a PL/SQL expression, get the results using a local
-- variable
SET SERVEROUTPUT ON
DECLARE
  sal employees.salary%type;
BEGIN
  sal := get_sal(100);
  DBMS_OUTPUT.PUT_LINE('The salary is: '|| sal);
END;
/
```

```
anonymous block completed
The salary is: 24000
```

Using Different Methods for Executing Functions

```
-- Use as a parameter to another subprogram

EXECUTE dbms_output.put_line(get_sal(100))
```

```
anonymous block completed
24000
```

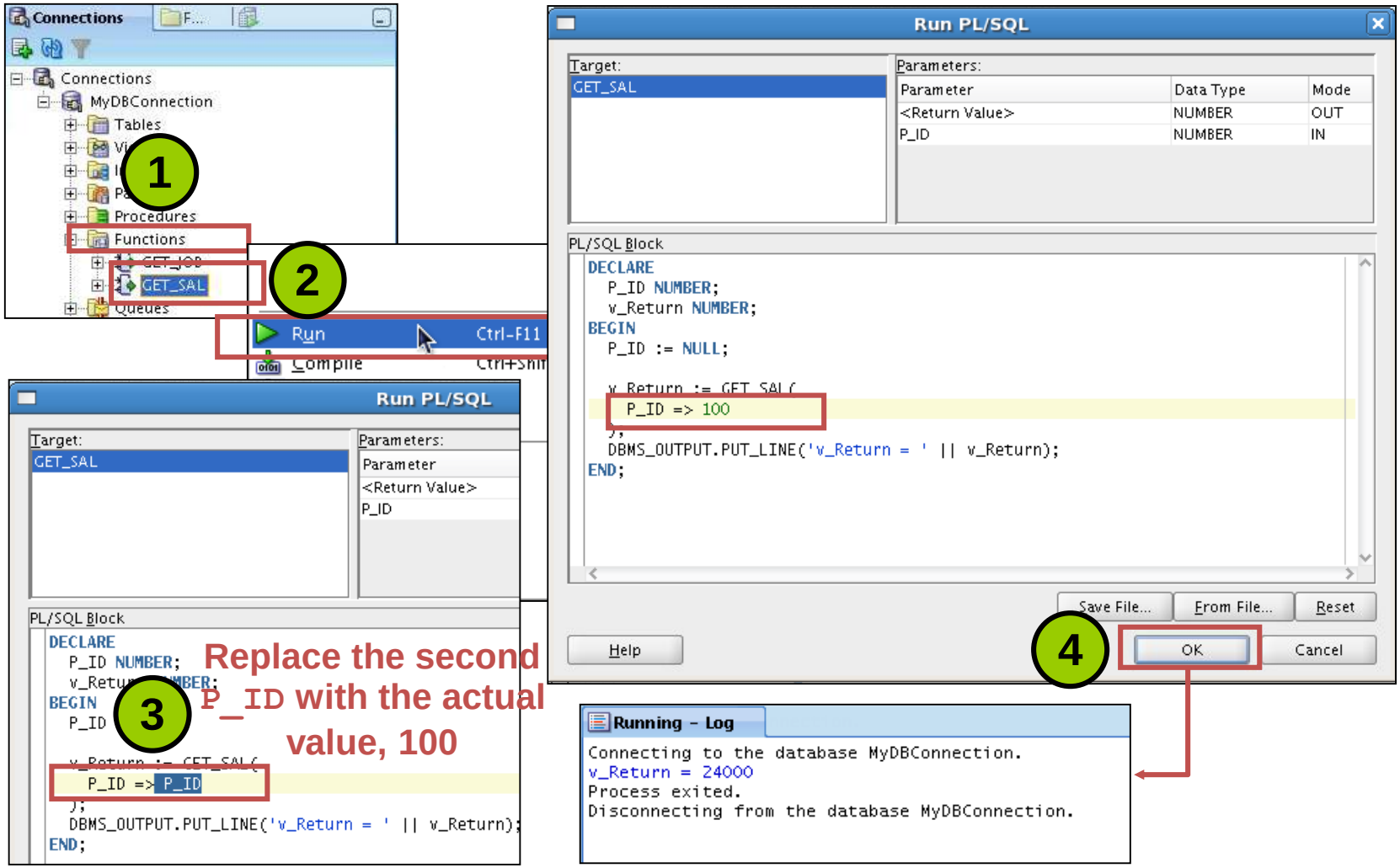
```
-- Use in a SQL statement (subject to restrictions)

SELECT job_id, get_sal(employee_id)
FROM employees;
```

```
JOB_ID     GET_SAL(EMPLOYEE_ID)
---------- --------------------
SH_CLERK   2600
SH_CLERK   2600
AD_ASST    4400
MK_MAN     13000
   ...
SH_CLERK   3100
SH_CLERK   3000

107 rows selected
```

Executing Functions Using SQL Developer



Using a Function in a SQL Expression

```
CREATE OR REPLACE FUNCTION tax(p_value IN NUMBER)
 RETURN NUMBER IS
BEGIN
   RETURN (p_value * 0.08);
END tax;
/
SELECT employee_id, last_name, salary, tax(salary)
FROM   employees
WHERE  department_id = 100;
```

```
FUNCTION tax(value Compiled.
EMPLOYEE_ID          LAST_NAME                 SALARY               TAX(SALARY)
-------------------- ------------------------- -------------------- --------------------
108                  Greenberg                 12000                960
109                  Faviet                    9000                 720
110                  Chen                      8200                 656
111                  Sciarra                   7700                 616
112                  Urman                     7800                 624
113                  Popp                      6900                 552

6 rows selected
```

Calling User-Defined Functions in SQL Statements

User-defined functions act like built-in single-row functions and can be used in:

- The `SELECT` list or clause of a query
- Conditional expressions of the `WHERE` and `HAVING` clauses
- The `CONNECT BY`, `START WITH`, `ORDER BY`, and `GROUP BY` clauses of a query
- The `VALUES` clause of the `INSERT` statement
- The `SET` clause of the `UPDATE` statement

Quiz: Choose all that is TRUE

A PL/SQL stored function:

1. Can be invoked as part of an expression
2. Must contain a `RETURN` clause in the header
3. Must return a single value
4. Must contain at least one `RETURN` statement
5. Does not contain a `RETURN` clause in the header