P
L
/
S
Q
L

Oracle11*g* : PL/SQL Programming

Chapter 6

Functions

Chapter Objectives

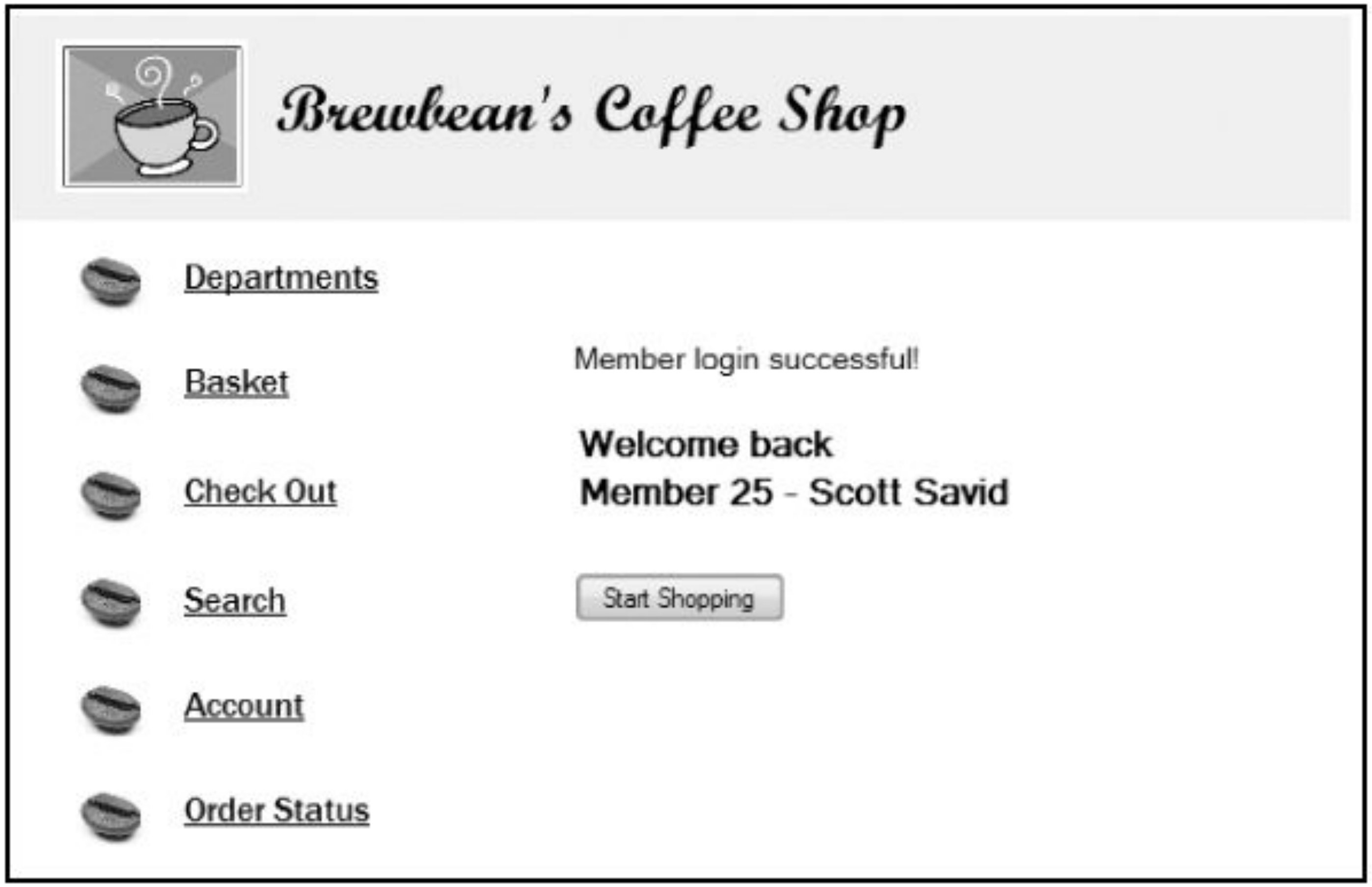
- After completing this lesson, you should be able to understand:
  - Functions
  - Creating a stored function
  - Using OUT parameters in functions
  - Including multiple RETURN statements in a function
  - Using a RETURN statement in a procedure
  - Using constraints of actual and formal parameters

Chapter Objectives (continued)

- After completing this lesson, you should be able to understand (continued):
  - Understanding and controlling how parameter values are passed
  - Working with function purity levels
  - Additional program unit options
  - Referencing the data dictionary for program units
  - Deleting program units

Brewbean’s Challenge

- Need program module to check a user login

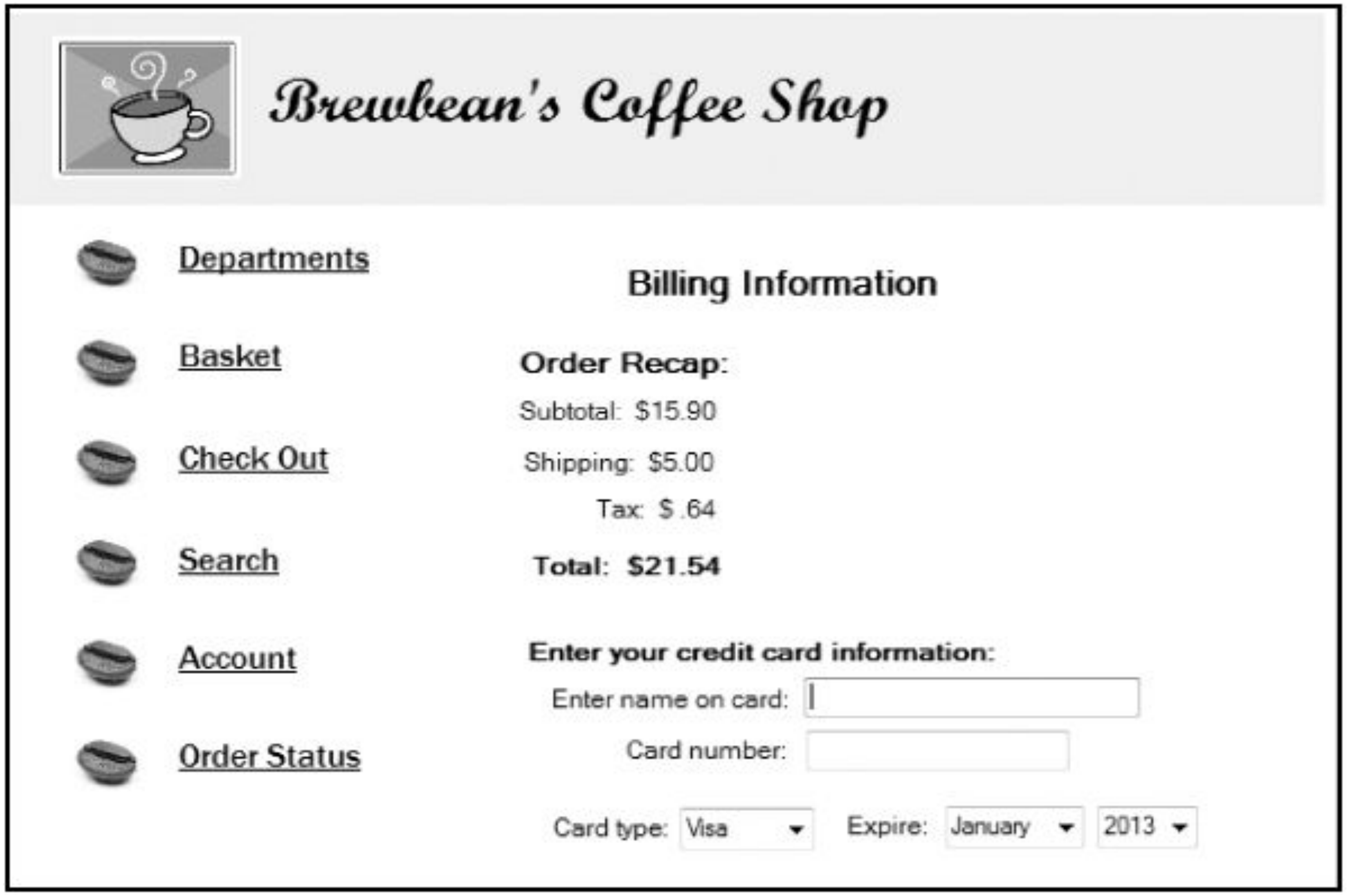


Member login successful!

Welcome back
Member 25 - Scott Savid

Brewbean's Challenge (continued)

- Need program module to calculate shipping cost based on the number of items in the basket



Billing Information

Order Recap:

Subtotal: $15.90

Shipping: $5.00

Tax: $ .64

Total: $21.54

Enter your credit card information:

Enter name on card:

Card number:

Card type: Visa Expire: January 2013

Introduction to Functions

- A function is similar to a procedure in that it can accomplish a task and retrieve/return values
- A function is part of an expression, not an entire statement such as a procedure
- Can be used in both PL/SQL and SQL statements
- Same as Oracle-supplied functions (ROUND, TO_CHAR)
- Contains a RETURN statement

Example of Oracle-Supplied Function

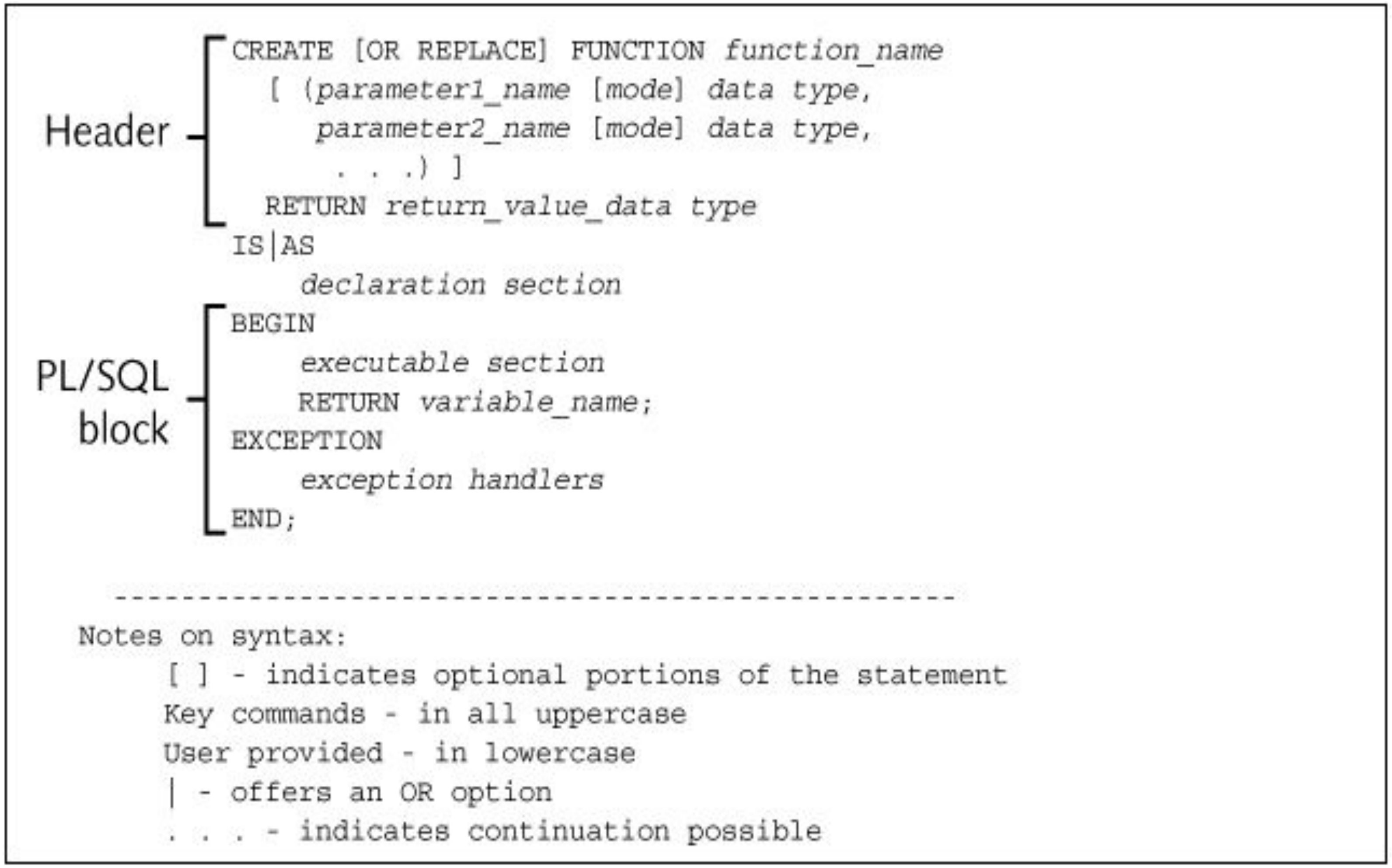
- SQL

```
SELECT idProduct, price, ROUND(price, 0)
  FROM bb_product
  WHERE idProduct < 4;
```

- PL/SQL

```
DECLARE
   v_amt1 number(5,2);
   v_amt2 number(3,0);
BEGIN
   v_amt1 := 32.50;
   v_amt2 := ROUND(v_amt1,0);
   DBMS_OUTPUT.PUT_LINE(v_amt2);
END;
```

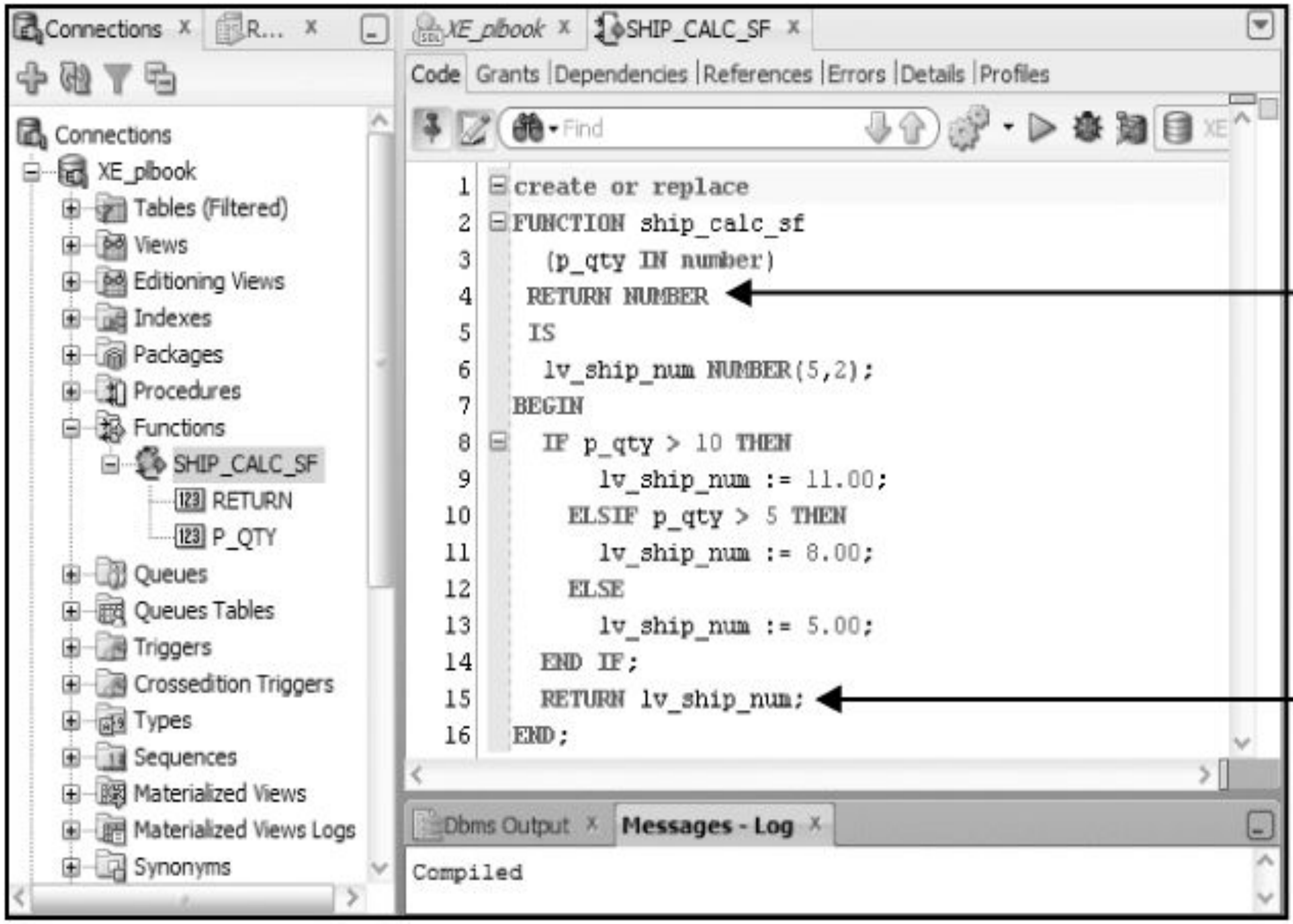
Function Create Statement


Header
PL/SQL block

```
CREATE [OR REPLACE] FUNCTION function_name
  [ (parameter1_name [mode] data type,
     parameter2_name [mode] data type,
      . . .) ]
  RETURN return_value_data type
IS|AS
    declaration section
BEGIN
    executable section
    RETURN variable_name;
EXCEPTION
    exception handlers
END;

 ---------------------------------------------------
Notes on syntax:
     [ ] - indicates optional portions of the statement
     Key commands - in all uppercase
     User provided - in lowercase
     | - offers an OR option
     . . . - indicates continuation possible
```

Function Example



```
create or replace
FUNCTION ship_calc_sf
  (p_qty IN number)
 RETURN NUMBER
 IS
  lv_ship_num NUMBER(5,2);
BEGIN
  IF p_qty > 10 THEN
      lv_ship_num := 11.00;
    ELSIF p_qty > 5 THEN
      lv_ship_num := 8.00;
    ELSE
      lv_ship_num := 5.00;
  END IF;
  RETURN lv_ship_num;
END;
```

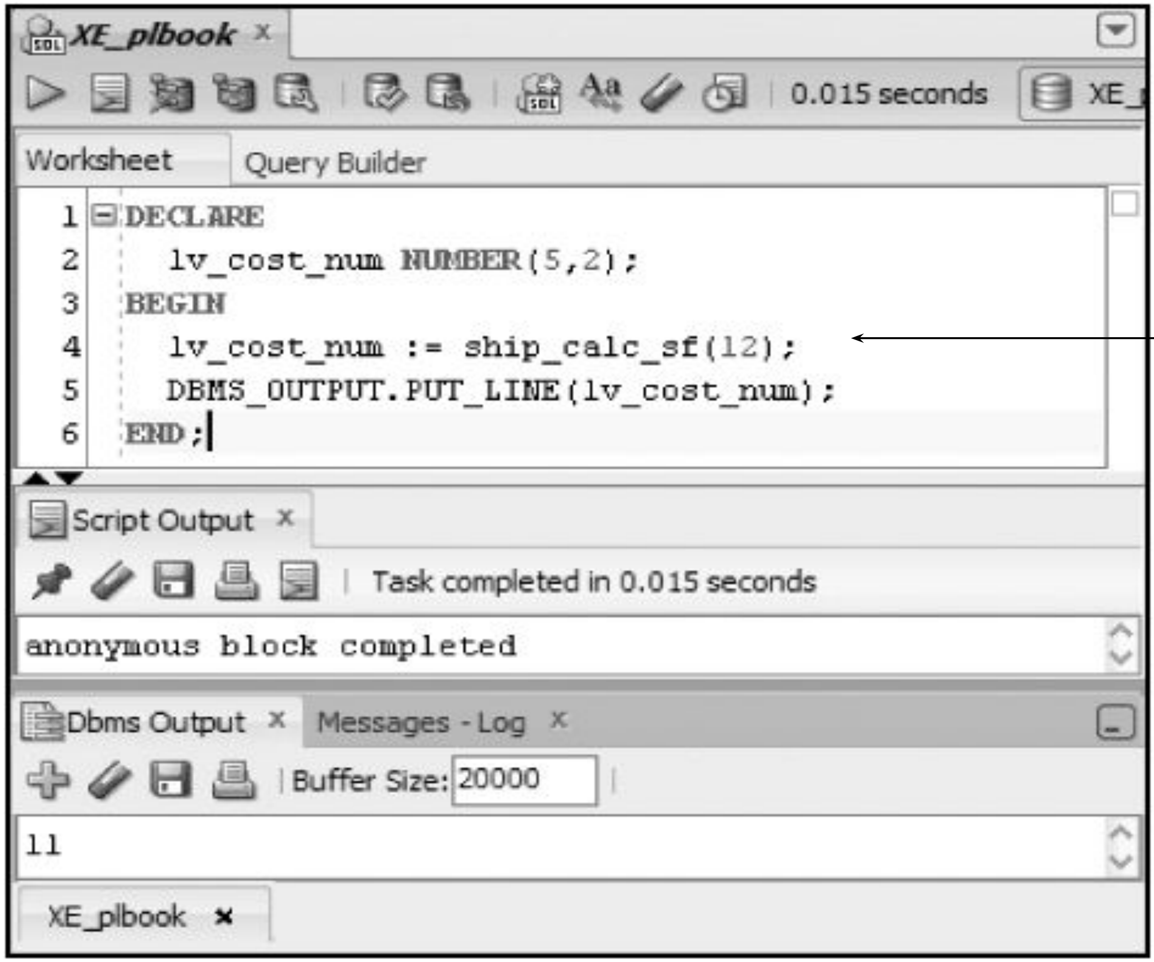
```
Compiled
```

Declares the data type of the return value

RETURN statement indicates the value to be returned

Invoking a Function from a Block

•An assignment statement is used – a function RETURNS a value!

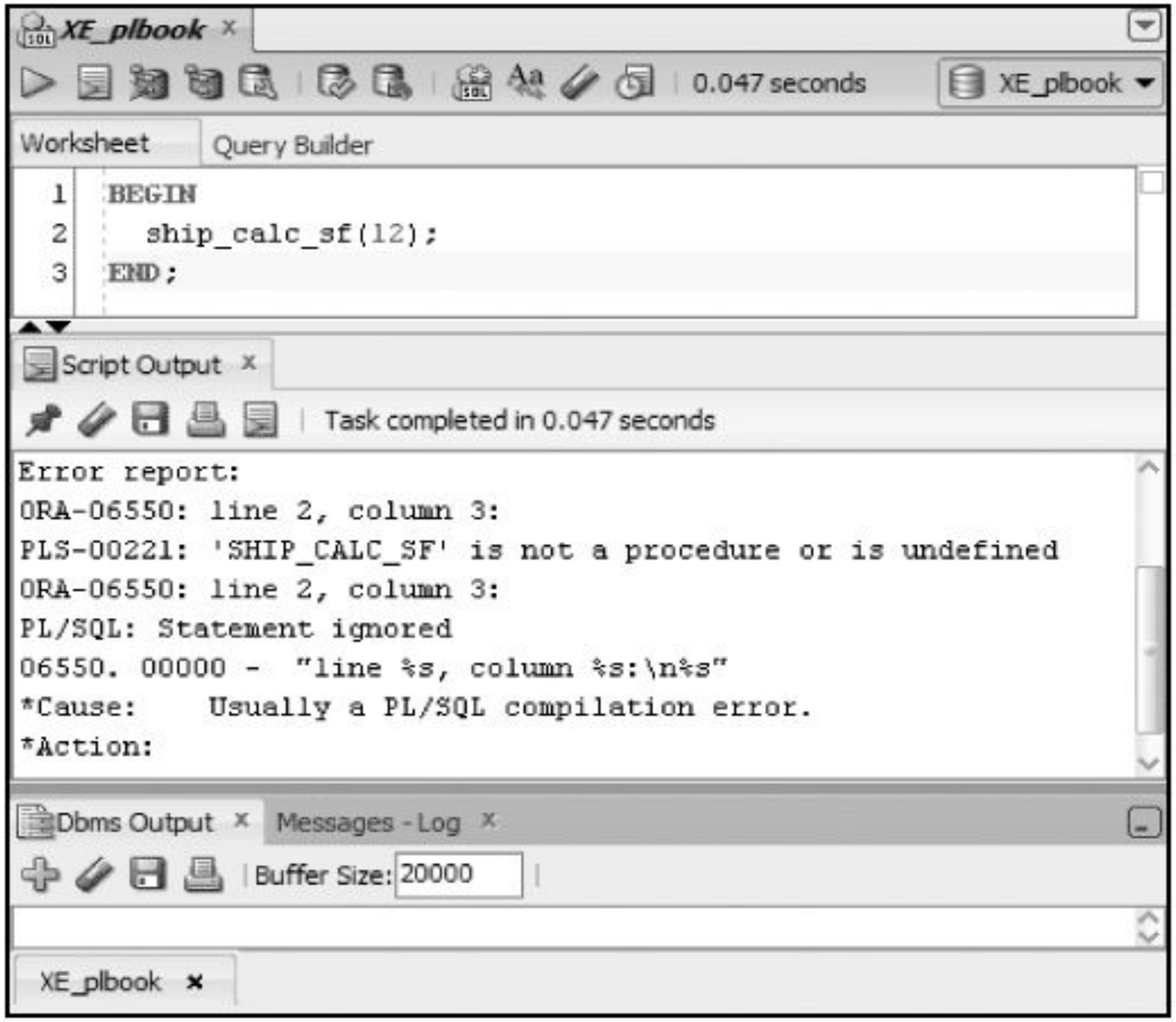


```
DECLARE
  lv_cost_num NUMBER(5,2);
BEGIN
  lv_cost_num := ship_calc_sf(12);
  DBMS_OUTPUT.PUT_LINE(lv_cost_num);
END;
```

```
anonymous block completed
```

```
11
```

Attempt to Invoke Stand-alone

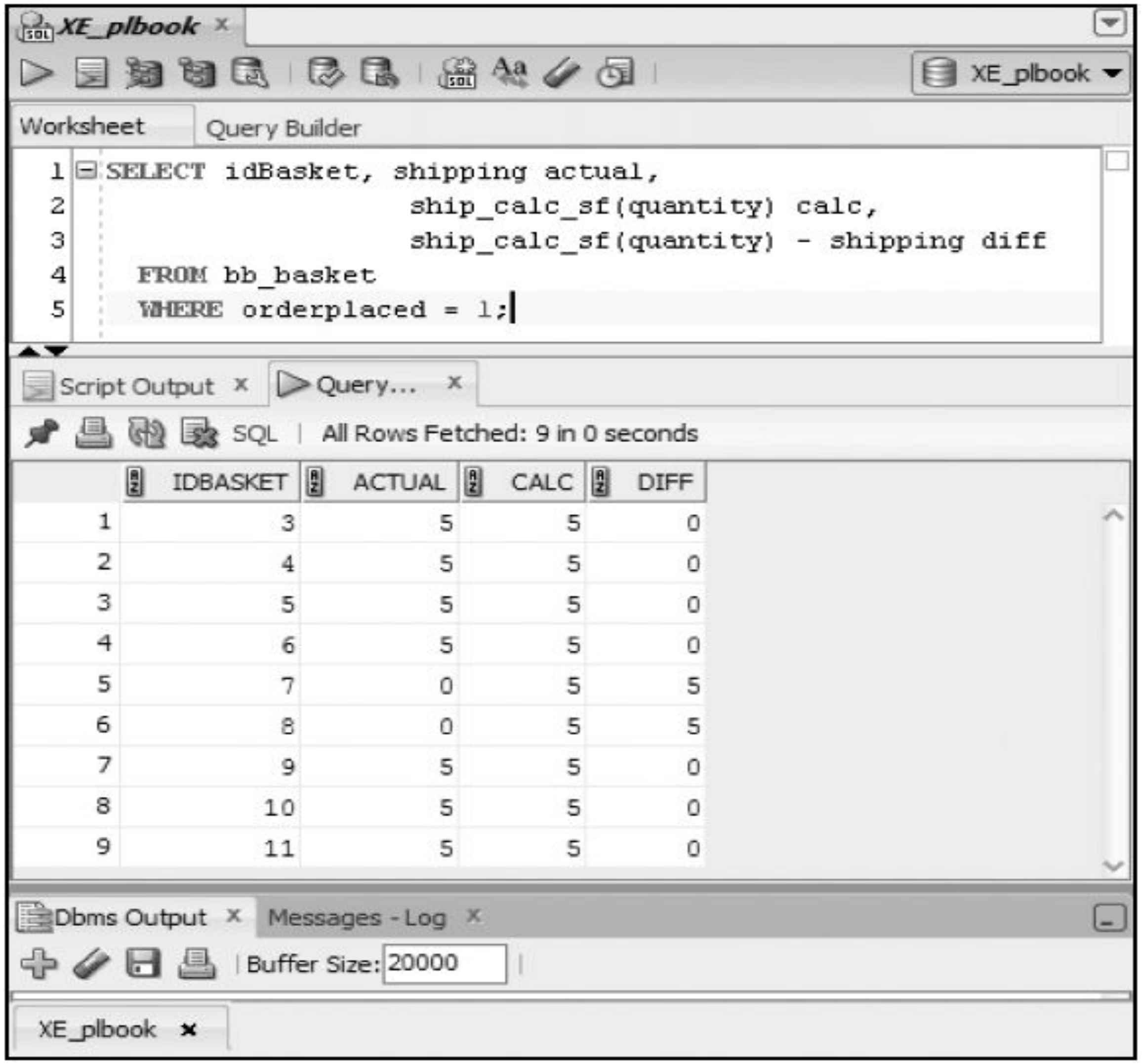


```
BEGIN
  ship_calc_sf(12);
END;
```

```
Error report:
ORA-06550: line 2, column 3:
PLS-00221: 'SHIP_CALC_SF' is not a procedure or is undefined
ORA-06550: line 2, column 3:
PL/SQL: Statement ignored
06550. 00000 -  "line %s, column %s:\n%s"
*Cause:    Usually a PL/SQL compilation error.
*Action:
```

Use Function in SQL

P
L
/
S
Q
L



```
SELECT idBasket, shipping actual,
                ship_calc_sf(quantity) calc,
                ship_calc_sf(quantity) - shipping diff
  FROM bb_basket
  WHERE orderplaced = 1;
```

SQL | All Rows Fetched: 9 in 0 seconds

	IDBASKET	ACTUAL	CALC	DIFF
1	3	5	5	0
2	4	5	5	0
3	5	5	5	0
4	6	5	5	0
5	7	0	5	5
6	8	0	5	5
7	9	5	5	0
8	10	5	5	0
9	11	5	5	0

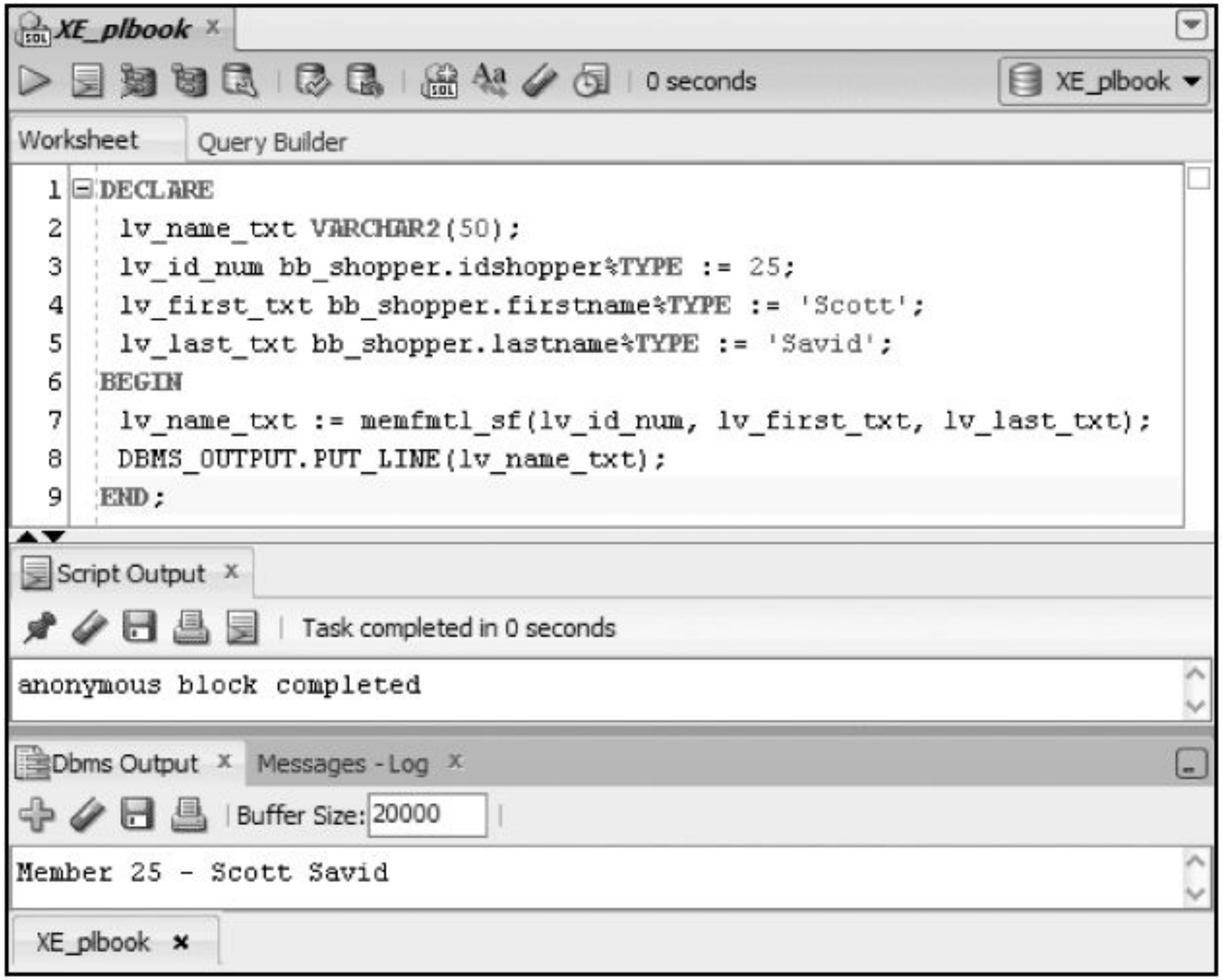
Brewbean's Member Display

P
L
/
S
Q
L

```
CREATE OR REPLACE FUNCTION memfmt1_sf
  (p_id IN NUMBER,
   p_first IN VARCHAR2,
   p_last IN VARCHAR2)
 RETURN VARCHAR2
 IS
   lv_mem_txt VARCHAR2(35);
BEGIN
 lv_mem_txt := 'Member ' || p_id || ' - ' || p_first
               || ' ' || p_last;
 RETURN lv_mem_txt;
END;
```

P
L
/
S
Q
L

Member Display Test



```
DECLARE
 lv_name_txt VARCHAR2(50);
 lv_id_num bb_shopper.idshopper%TYPE := 25;
 lv_first_txt bb_shopper.firstname%TYPE := 'Scott';
 lv_last_txt bb_shopper.lastname%TYPE := 'Savid';
BEGIN
 lv_name_txt := memfmtl_sf(lv_id_num, lv_first_txt, lv_last_txt);
 DBMS_OUTPUT.PUT_LINE(lv_name_txt);
END;
```

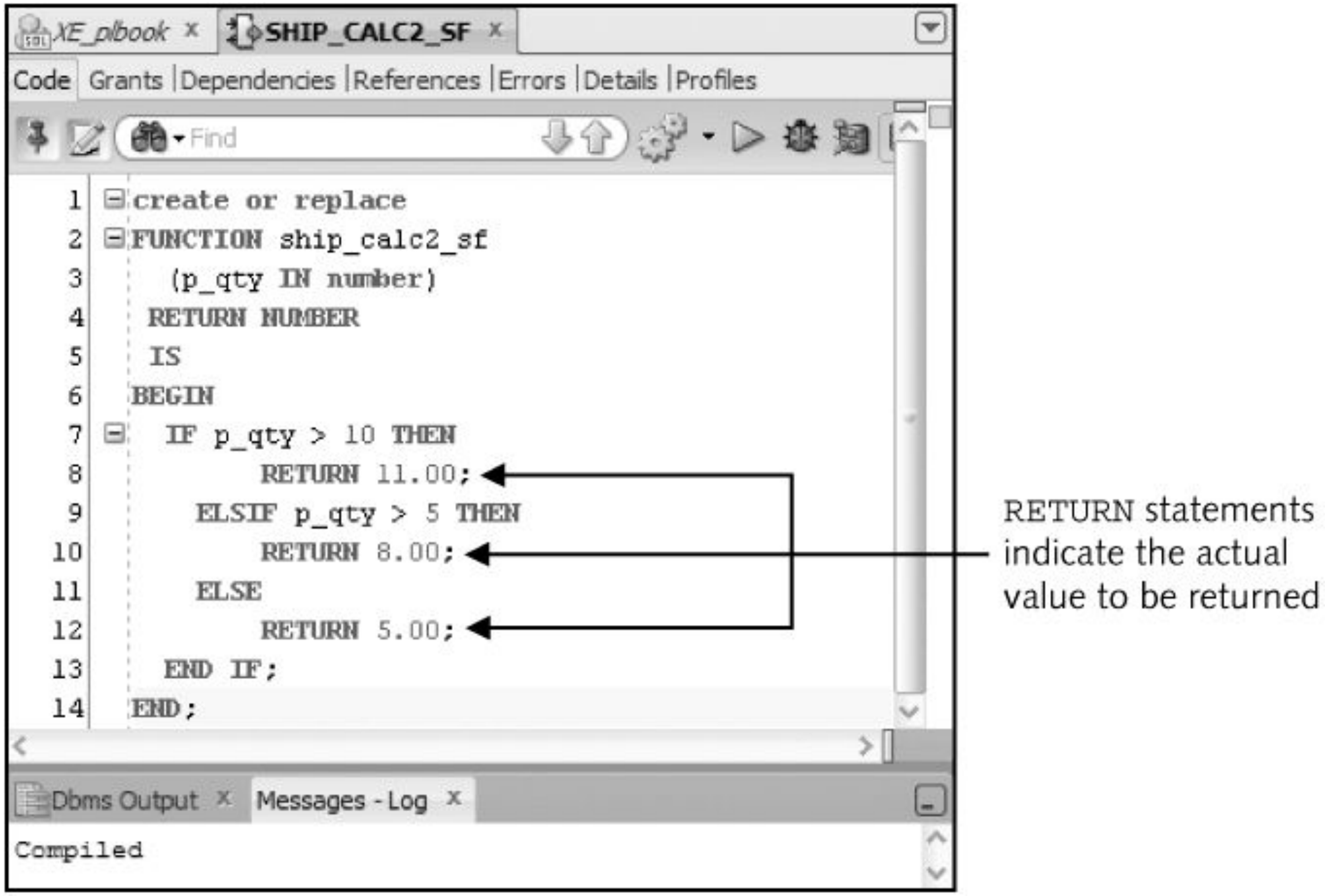
```
anonymous block completed
```

```
Member 25 - Scott Savid
```

Using OUT Mode in a Function

- OUT parameters are not typically used in functions, as:
  - Mixing OUT and RETURN values can lead to confusion
  - It prohibits the function from being used in SQL

Multiple RETURN Statements



```
create or replace
FUNCTION ship_calc2_sf
  (p_qty IN number)
 RETURN NUMBER
 IS
BEGIN
  IF p_qty > 10 THEN
        RETURN 11.00;
    ELSIF p_qty > 5 THEN
        RETURN 8.00;
    ELSE
        RETURN 5.00;
  END IF;
END;
```

```
Compiled
```

RETURN statements indicate the actual value to be returned

Note: Only one RETURN statement can execute

P
L
/
S
Q
L

RETURN Statement in a Procedure

- Different purpose than a RETURN statement in a function
- Used to change flow of execution
- Stops processing in that block and moves to the next statement after the procedure call
- Stand-alone statement with no arguments

Parameter Constraints

P
L
/
S
Q
L

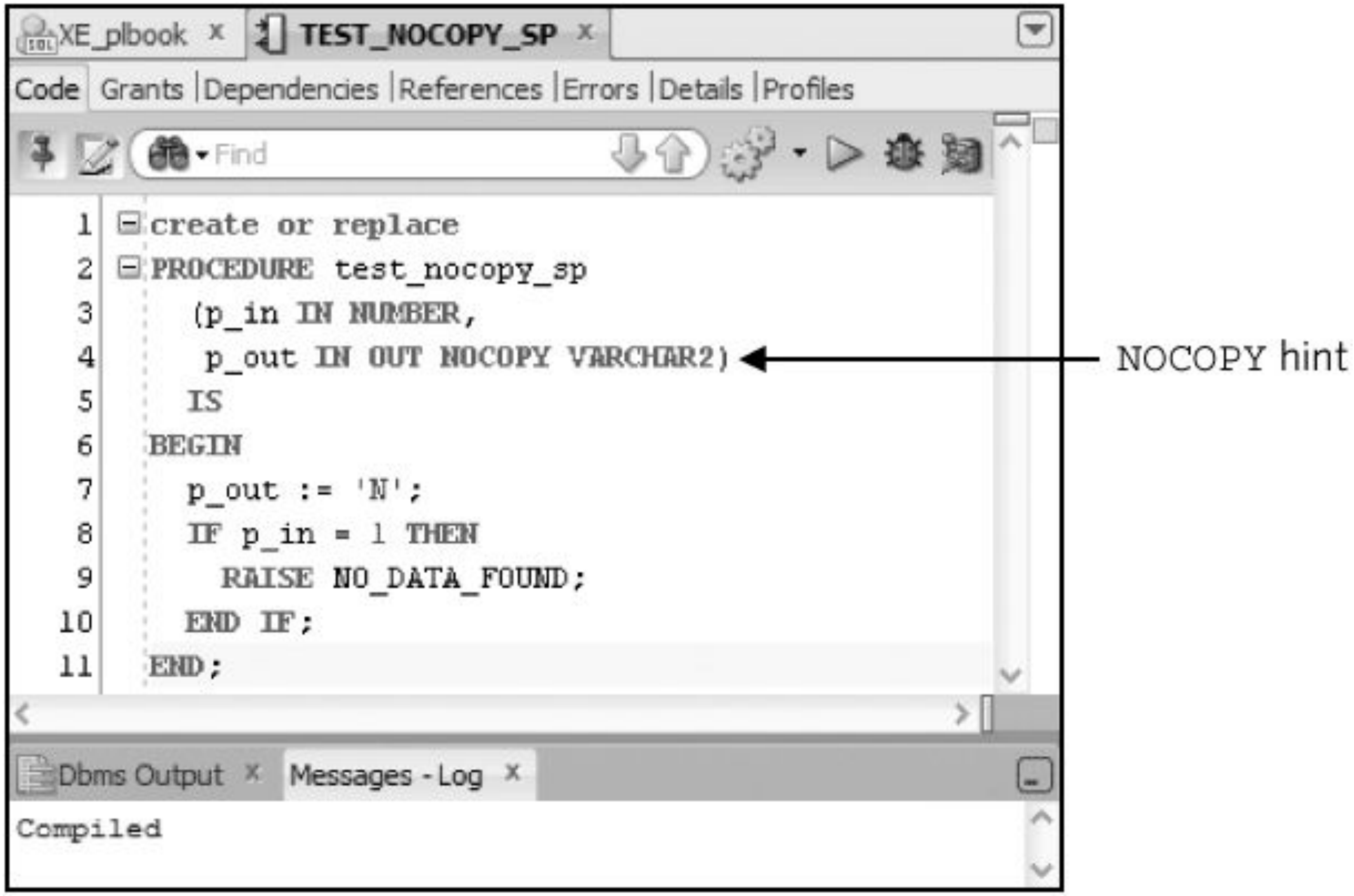
- Formal parameters – included in a program unit
- Actual parameters – arguments used in a program unit call
- Argument for an OUT parameter must be a variable to hold the value returned
- Actual parameters determine the size of the formal parameters

Passing Parameter Values

P
L
/
S
Q
L

- Two techniques used to pass values between actual and formal parameters:
  1. Passed by Reference – create pointer to value in the actual parameter
  2. Passed by Value – copies value from actual to formal parameter
- Pass by value is the default
- Use a compiler hint to use pass by reference

Pass by Reference



```
create or replace
PROCEDURE test_nocopy_sp
   (p_in IN NUMBER,
    p_out IN OUT NOCOPY VARCHAR2)
   IS
BEGIN
   p_out := 'N';
   IF p_in = 1 THEN
     RAISE NO_DATA_FOUND;
   END IF;
END;
```

```
Compiled
```

Purity Levels

- Restrictions on functions used in SQL
  - If used in a remote or parallel operation, no reading or writing of packaged variables allowed
  - If used in a SELECT, VALUES, or SET clause, the function can write values to packaged variables; otherwise, it is not allowed

Purity Levels (continued)

- Restrictions on functions used in SQL (continued)
  - Functions cannot be used in a check constraint or as a default value of a table column
  - If the function calls other subprograms, the subprograms cannot break these rules
  - Must be a stored database object (or in a stored package)
  - Can use only IN parameters
  - Must be a row function (not a group function)

Purity Levels (continued)

P
L
/
S
Q
L

- Restrictions on functions used in SQL (continued)
  - Formal parameter data types must use database data types (no PL/SQL data types such as BOOLEAN are permitted)
  - Return data types must be a database data type
  - Must not issue transaction control statements to end the current transaction prior to execution
  - Cannot issue ALTER SESSION or ALTER SYSTEM commands

Purity Levels (continued)

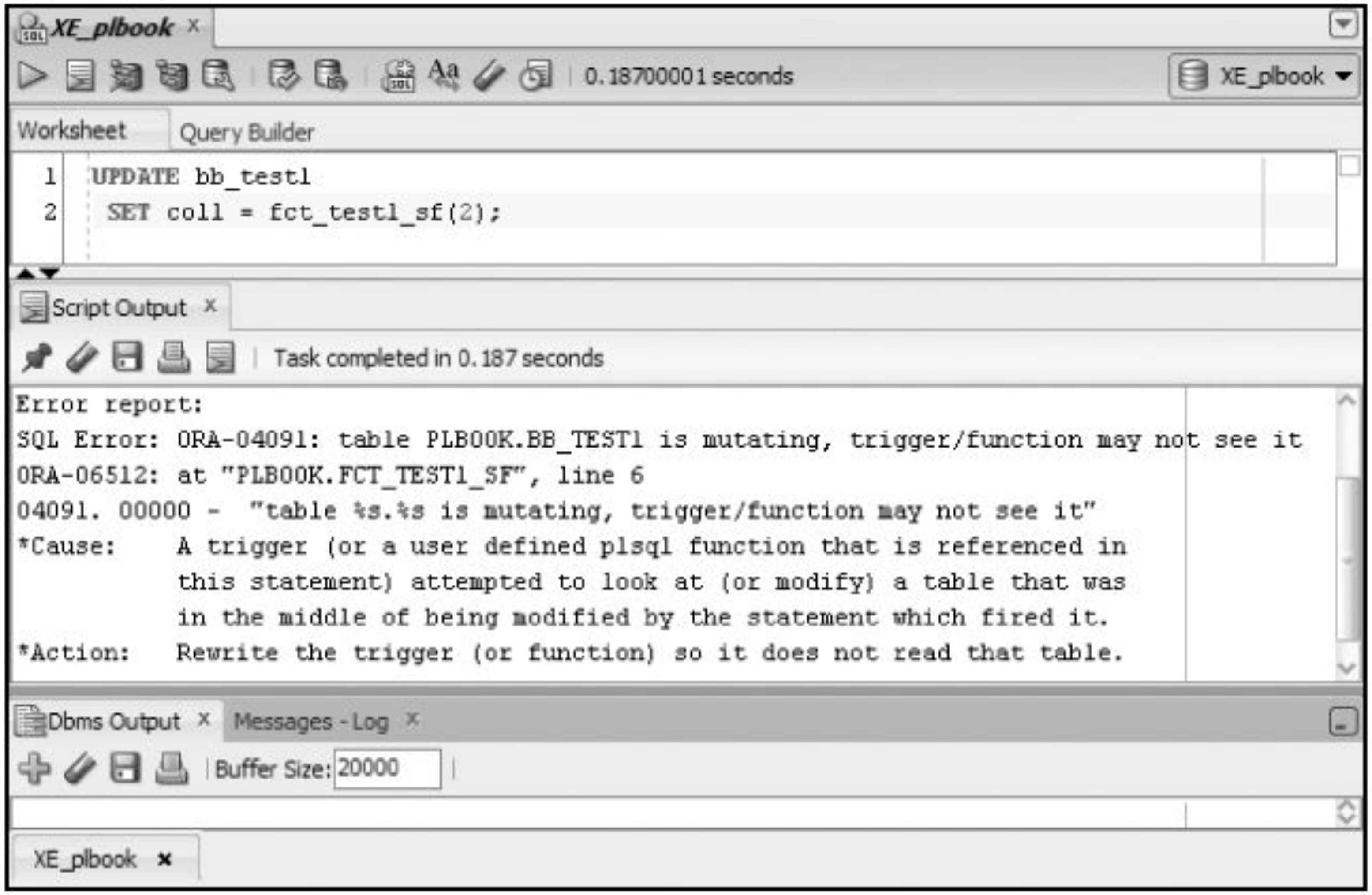
Level Acronym	Level Name	Level Description
WNDS	Writes No Database State	Function does not modify any database tables (No DML)
RNDS	Reads No Database State	Function does not read any tables (No select)
WNPS	Writes No Package State	Function does not modify any packaged variables (packaged variables are variables declared in a package specification; they are discussed in detail in Chapter 6)
RNPS	Reads No Package State	Function does not read any packaged variables

Purity Levels Test

Function that updates table bb_test1

```
CREATE OR REPLACE FUNCTION fct_test1_sf
     (p_num IN NUMBER)
      RETURN NUMBER
  IS
BEGIN
     UPDATE bb_test1
       SET col1 = p_num;
     RETURN p_num;
END;
```

Purity Levels Test

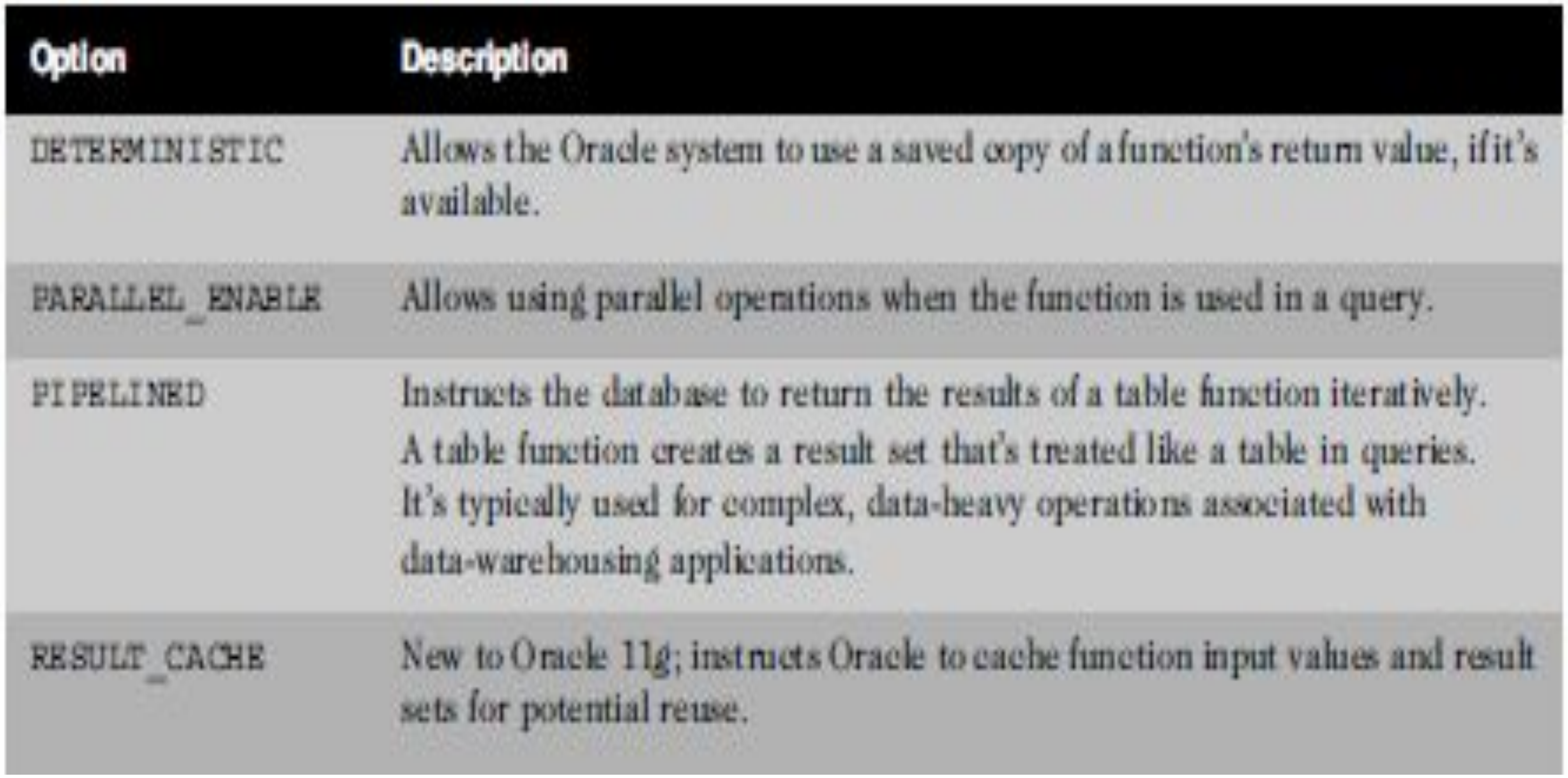


```
UPDATE bb_test1
 SET col1 = fct_test1_sf(2);
```

```
Error report:
SQL Error: ORA-04091: table PLBOOK.BB_TEST1 is mutating, trigger/function may not see it
ORA-06512: at "PLBOOK.FCT_TEST1_SF", line 6
04091. 00000 -  "table %s.%s is mutating, trigger/function may not see it"
*Cause:    A trigger (or a user defined plsql function that is referenced in
           this statement) attempted to look at (or modify) a table that was
           in the middle of being modified by the statement which fired it.
*Action:   Rewrite the trigger (or function) so it does not read that table.
```

Additional Options

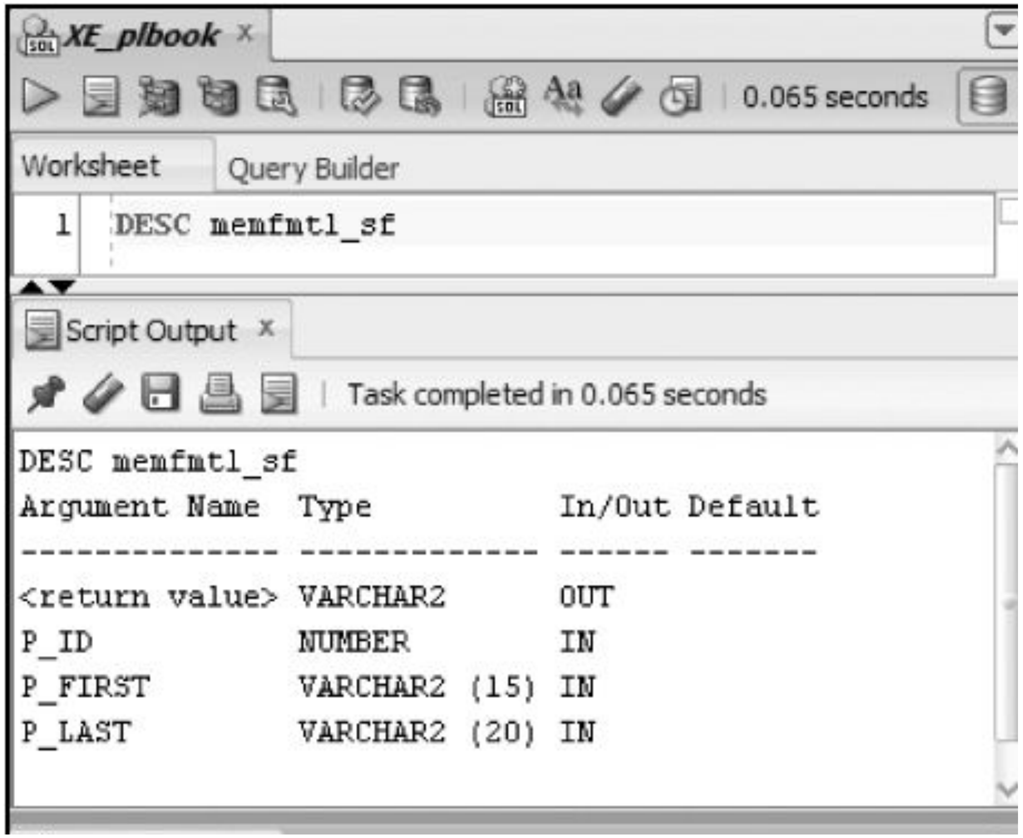
P
L
/
S
Q
L



Option	Description
DETERMINISTIC	Allows the Oracle system to use a saved copy of a function's return value, if it's available.
PARALLEL_ENABLE	Allows using parallel operations when the function is used in a query.
PIPELINED	Instructs the database to return the results of a table function iteratively. A table function creates a result set that's treated like a table in queries. It's typically used for complex, data-heavy operations associated with data-warehousing applications.
RESULT_CACHE	New to Oracle 11g; instructs Oracle to cache function input values and result sets for potential reuse.

Data Dictionary Information

- DESCRIBE identifies parameters and return value data type

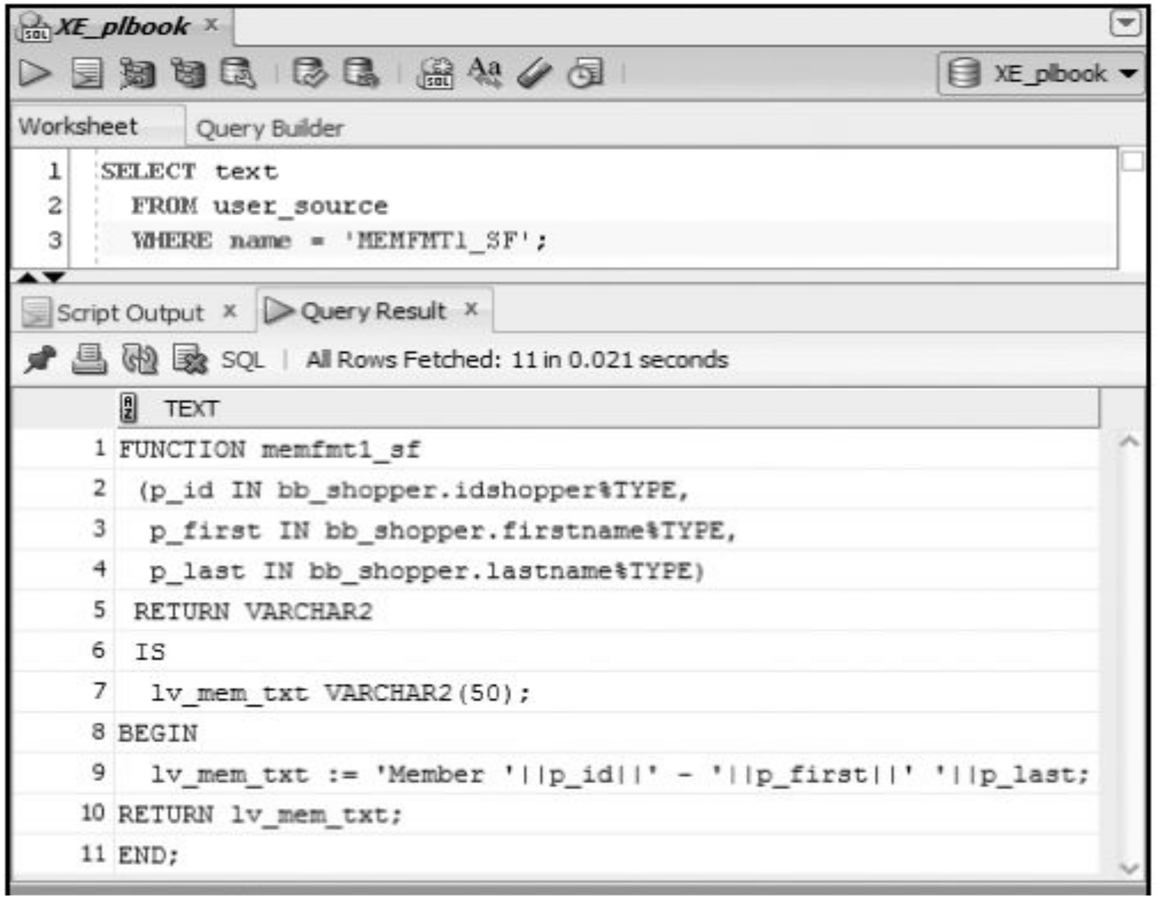


```
DESC memfmtl_sf
```

```
DESC memfmtl_sf
Argument Name  Type          In/Out Default
-------------- ------------- ------ -------
<return value> VARCHAR2      OUT
P_ID           NUMBER        IN
P_FIRST        VARCHAR2 (15) IN
P_LAST         VARCHAR2 (20) IN
```

Data Dictionary Information (continued)

- View source code using USER_SOURCE



```
SELECT text
  FROM user_source
  WHERE name = 'MEMFMT1_SF';
```

All Rows Fetched: 11 in 0.021 seconds

```
TEXT
FUNCTION memfmt1_sf
 (p_id IN bb_shopper.idshopper%TYPE,
  p_first IN bb_shopper.firstname%TYPE,
  p_last IN bb_shopper.lastname%TYPE)
 RETURN VARCHAR2
 IS
  lv_mem_txt VARCHAR2(50);
BEGIN
  lv_mem_txt := 'Member '||p_id||' - '||p_first||' '||p_last;
RETURN lv_mem_txt;
END;
```

Delete Functions

DROP FUNCTION *function_name*;

P
L
/
S
Q
L

Summary

- Functions can be used in PL/SQL and SQL statements
- A function is part of an expression
- Functions include parameters and must return a value
- OUT parameter rarely used
- Pass parameter values by value or reference
- Multiple RETURN statements can be included and only one is executed

Summary (continued)

- Actual versus formal parameters
  - Formal parameters – included in a program unit
  - Actual parameters – arguments used in a program unit call
- Purity levels refer to rules for functions to be used in SQL statements
- Options are available for improving performance such as PARALLEL_ENABLE
- DESCRIBE and USER_SOURCE view
- DROP command removed a function