

# Database Programming with PL/SQL 9-1: Creating Functions Practice Activities **Vocabulary**

Identify the vocabulary word for each definition below:

|  |  |
| --- | --- |
| **Functie** | A named PL/SQL block that can accept optional IN parameters and must return a single output. |

**Try It / Solve It**

1. Name the characteristics of a stored function.

1. Create a function called full\_name. Pass two parameters to the function, an employee’s last name and first name. The function should return the full name in the format, last name, comma, space, first name (for example: Smith, Joe). Save your code.

CREATE OR REPLACE FUNCTION full\_name

(p\_first\_name IN employees.first\_name%TYPE,

p\_last\_name IN employees.last\_name%TYPE)

RETURN VARCHAR2 IS

full\_name employees.first\_name%TYPE;

BEGIN

RETURN p\_first\_name ||' , ' || p\_last\_name;

END full\_name;

* 1. Test your function from an anonymous block which uses a local variable to store and display the returned value.

DECLARE

name VARCHAR2(30);

BEGIN

name := full\_name('Smith', 'John');

DBMS\_OUTPUT.PUT\_LINE(name);

END;

* 1. Modify your anonymous block from the previous step to remove the local variable declaration and call the function directly from within the DBMS\_OUTPUT.PUT\_LINE call. Test the block again.

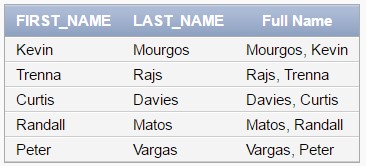
DECLARE

BEGIN

DBMS\_OUTPUT.PUT\_LINE(full\_name('Smith', 'John'));

END;

* 1. Now call the function from within a SELECT statement, not a PL/SQL block. Your SELECT statement should display the first\_name, last\_name, and full name (using the function) of all employees in department 50. Your output should look like this:



SELECT first\_name, last\_name, full\_name(last\_name, first\_name)

FROM employees

WHERE department\_id = 50;

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

1. Create a function called divide that accepts two numbers as input and returns the result of dividing the first number by the second number, rounded to two decimal places. Save your code.

CREATE OR REPLACE FUNCTION divide

(nr1 NUMBER,nr2 NUMBER)

RETURN NUMBER IS

rez NUMBER;

BEGIN

rez := ROUND(nr1 / nr2, 2) ;

RETURN rez;

END;

* 1. Test your function twice from an anonymous block using input values (50, 2) and (25, 3).

BEGIN

DBMS\_OUTPUT.PUT\_LINE(divide(50, 2));

END;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(divide(25, 3));

END;

* 1. Test your function a third time using input values (16, 0). What happens?

**ORA-01476: divisor is equal to zero**

* 1. Modify the function code to trap the ZERO\_DIVIDE exception. The exception handler should return a value of zero from the function if ZERO\_DIVIDE is raised.

CREATE OR REPLACE FUNCTION divide

(nr1 NUMBER,nr2 NUMBER)

RETURN NUMBER IS

rez NUMBER;

BEGIN

rez := ROUND(nr1 / nr2, 2) ;

RETURN rez;

EXCEPTION

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE('Zero\_Divide');

rez := 0;

RETURN rez;

END;

* 1. Test your function again using input values (16,0) as before. Now what happens?

BEGIN

DBMS\_OUTPUT.PUT\_LINE(divide(16, 0));

END;

In acest caz exceptia este interceptata si apare rezultatul 0.

1. List four major differences between a procedure and a function.

-Functia returneaza un rezultat, in timp ce subprogramul poate sa returneze sau NU una sau mai multe valori

-Functia are pentru parametru doar tipul IN

- Clauza RETURN este folosita in schimb la modul OUT

-Procedura este invocata dintr-o instructiune PL/SQL, in timp ce functia este invocata ca parte a unei expresii.

1. Look at the following two subprograms. The first is a procedure; the second is a function. Answer the following questions.

CREATE OR REPLACE PROCEDURE get\_country\_name\_proc

(p\_country\_id IN countries.country\_id%TYPE,

p\_country\_name OUT countries.country\_name%TYPE)

IS

BEGIN

SELECT country\_name INTO p\_country\_name

FROM countries

WHERE country\_id = p\_country\_id;

END;

CREATE OR REPLACE FUNCTION get\_country\_name\_func

(p\_country\_id IN countries.country\_id%TYPE)

RETURN VARCHAR2

IS

v\_country\_name countries.country\_name%TYPE;

BEGIN

SELECT country\_name INTO v\_country\_name

FROM countries

WHERE country\_id = p\_country\_id;

RETURN v\_country\_name; END;

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

2

1. For a given country id, will both of these subprograms return the same results?

Vor returna acelasi rezultat.

1. What is the advantage of creating this subprogram as a function rather than as a procedure?

Functia este mai flexibila si poate fi apelata mai usor.

1. Which of the following procedure and function calls are valid and which are not? Explain why the invalid ones will fail.

DECLARE

v\_country\_id countries.country\_id%TYPE := 2; v\_country\_name countries.country\_name%TYPE; BEGIN

get\_country\_name\_proc(v\_country\_id, v\_country\_name); -- Call 1 v\_country\_name := get\_country\_name\_func(v\_country\_id); -- Call 2 v\_country\_name := get\_country\_name\_proc(v\_country\_id); -- Call 3

END;

Eroare la apelul 3 **PLS-00306: wrong number or types of arguments in call to 'GET\_COUNTRY\_NAME\_PROC'**

SELECT get\_country\_name\_proc(country\_id) -- Call 4

FROM countries;

Eroare la apelul 4 fiindca avem prea putine argumente   
ORA-00904: "GET\_COUNTRY\_NAME\_PROC": invalid identifier

SELECT get\_country\_name\_func(country\_id) -- Call 5

FROM countries;

1. List the ways you can invoke (i.e., call) a function.

-As part ol PL/SQl expressions

-As a parameter to another subprogram

-As an expression in a SQL statment

1. Create a function which accepts a character string as input and returns the same character string but with the order of the letters reversed. For example, "Smith" would be returned as "htimS." Save your code. Hint: you will need to declare a local variable to store the reversed string, and build its contents by reading the input one character at a time (using SUBSTR) in a loop structure, starting from the last character. Each execution of the loop reads the preceding character and concatenates it to the reversed string.

CREATE OR REPLACE FUNCTION reverse\_string(p\_sir VARCHAR2)

RETURN VARCHAR2

IS

i integer;

v\_character CHAR;

result VARCHAR2(50):= ' ';

BEGIN

result:=NULL;

i:=LENGTH(p\_sir);

LOOP

result:=result || SUBSTR(p\_sir,i,1);

i:=i-1;

EXIT WHEN i=0;

END LOOP;

return result;

END;

1. Test your function using the following SQL statements:

SELECT last\_name, reverse\_string(last\_name)

FROM employees;

SELECT country\_name, reverse\_string(country\_name) FROM countries;