1.Hello World Program in PL/SQL

2. PL/SQL Program To Add Two Numbers

3. PL/SQL Program for Prime Number

4. PL/SQL Program to Find Factorial of a Number

5. PL/SQL Program to Print Table of a Number

6. PL/SQL Program for Reverse of a Number

7. PL/SQL Program for Fibonacci Series

8. PL/SQL Program to Check Number is Odd or Even

9. PL/SQL Program to Reverse a String

10 .Pl/SQL Program for Palindrome Number

11. PL/SQL Program to Swap two Numbers

12. PL/SQL Program for Armstrong Number

PL/SQL FOR LOOP examples

Let’s take some examples of using the FOR LOOP statement to understand how it works.

A) Simple PL/SQL FOR LOOP example

In this example, the loop index is l\_counter, lower\_bound is one, and upper\_bound is five. The loop shows a list of integers from 1 to 5.

BEGIN

FOR l\_counter IN 1..5

LOOP

DBMS\_OUTPUT.PUT\_LINE( l\_counter );

END LOOP;

END;

Variable Scope in PL/SQL

PL/SQL allows the nesting of blocks, i.e., each program block may contain another inner block. If a variable is declared within an inner block, it is not accessible to the outer block. However, if a variable is declared and accessible to an outer block, it is also accessible to all nested inner blocks. There are two types of variable scope −

Local variables − Variables declared in an inner block and not accessible to outer blocks.

Global variables − Variables declared in the outermost block or a package.

Following example shows the usage of Local and Global variables in its simple form −

DECLARE

-- Global variables

num1 number := 95;

num2 number := 85;

BEGIN

dbms\_output.put\_line('Outer Variable num1: ' || num1);

dbms\_output.put\_line('Outer Variable num2: ' || num2);

DECLARE

-- Local variables

num1 number := 195;

num2 number := 185;

BEGIN

dbms\_output.put\_line('Inner Variable num1: ' || num1);

dbms\_output.put\_line('Inner Variable num2: ' || num2);

END;

END;

/

PL/SQL variable assignment

In PL/SQL, to assign a value or a variable to another, you use the assignment operator ( := ) which is a colon( :) followed by the equal sign( = ).

Please see the code listing below to get a better understanding:

DECLARE

v\_first\_name EMPLOYEES.FIRST\_NAME%TYPE;

v\_last\_name EMPLOYEES.LAST\_NAME%TYPE;

n\_employee\_id EMPLOYEES.EMPLOYEE\_ID%TYPE;

d\_hire\_date EMPLOYEES.HIRE\_DATE%TYPE;

BEGIN

v\_first\_name := 'Mary';

v\_last\_name := 'Jane';

d\_hire\_date := to\_date('19700101','YYYYMMDD');

END;

/

Code language: SQL (Structured Query Language) (sql)

In the example above, we assigned Mary to v\_first\_namevariable, Janeto v\_last\_namevariable, and result of the to\_datefunction to d\_hire\_datevariable.

You can use INTOof the SELECT statement to assign a value to a variable. The INTOclause moves the values from the SELECTquery’s column list into corresponding PL/SQL variables.

SET SERVEROUTPUT ON SIZE 1000000;

DECLARE

v\_first\_name EMPLOYEES.FIRST\_NAME%TYPE;

v\_last\_name EMPLOYEES.LAST\_NAME%TYPE;

n\_employee\_id EMPLOYEES.EMPLOYEE\_ID%TYPE;

d\_hire\_date EMPLOYEES.HIRE\_DATE%TYPE;

BEGIN

SELECT employee\_id,

first\_name,

last\_name,

hire\_date

INTO n\_employee\_id,

v\_first\_name,

v\_last\_name,

d\_hire\_date

FROM employees

WHERE employee\_id = 200;

DBMS\_OUTPUT.PUT\_LINE(v\_first\_name);

DBMS\_OUTPUT.PUT\_LINE(v\_last\_name);

DBMS\_OUTPUT.PUT\_LINE(d\_hire\_date);

END;

/

1)write an example using **Anonymous blocks**

2)Write an example using %Type and %rowtype

3)write an example for manipulation in plsql

4) write an examples for statements using for,while,if,switch and continue.