## Variable Declaration in PL/SQL

PL/SQL variables must be declared in the declaration section or in a package as a global variable. When you declare a variable, PL/SQL allocates memory for the variable's value and the storage location is identified by the variable name.

The syntax for declaring a variable is −

variable\_name [CONSTANT] datatype [NOT NULL] [:= | DEFAULT initial\_value]

Where, *variable\_name* is a valid identifier in PL/SQL, *datatype* must be a valid PL/SQL data

sales number(10, 2);

name varchar2(25);

address varchar2(100);

## Initializing Variables in PL/SQL

Whenever you declare a variable, PL/SQL assigns it a default value of NULL. If you want to initialize a variable with a value other than the NULL value, you can do so during the declaration, using either of the following −

* The **DEFAULT** keyword
* The **assignment** operator

For example −

counter binary\_integer := 0;

greetings varchar2(20) DEFAULT 'Have a Good Day';

Example

DECLARE

a integer := 10;

b integer := 20;

c integer;

f real;

BEGIN

c := a + b;

dbms\_output.put\_line('Value of c: ' || c);

f := 70.0/3.0;

dbms\_output.put\_line('Value of f: ' || f);

END;

/

## Assigning SQL Query Results to PL/SQL Variables

You can use the **SELECT INTO** statement of SQL to assign values to PL/SQL variables. For each item in the **SELECT list**, there must be a corresponding, type-compatible variable in the **INTO list**. The following example illustrates the concept. Let us create a table named CUSTOMERS −

The following program assigns values from the above table to PL/SQL variables using the **SELECT INTO clause** of SQL −

DECLARE

c\_id customers.id%type := 1;

c\_name customers.name%type;

c\_addr customers.address%type;

c\_sal customers.salary%type;

BEGIN

SELECT name, address, salary INTO c\_name, c\_addr, c\_sal

FROM customers

WHERE id = c\_id;

dbms\_output.put\_line

('Customer ' ||c\_name || ' from ' || c\_addr || ' earns ' || c\_sal);

END;

/

## Declaring a Constant

A constant is declared using the **CONSTANT** keyword. It requires an initial value and does not allow that value to be changed. For example −

PI CONSTANT NUMBER := 3.141592654;

DECLARE

-- constant declaration

pi constant number := 3.141592654;

## The PL/SQL Literals

A literal is an explicit numeric, character, string, or Boolean value not represented by an identifier. For example, TRUE, 786, NULL, 'tutorialspoint' are all literals of type Boolean, number, or string. PL/SQL, literals are case-sensitive. PL/SQL supports the following kinds of literals −

* Numeric Literals
* Character Literals
* String Literals
* BOOLEAN Literals
* Date and Time Literals
* To embed single quotes within a string literal, place two single quotes next to each other as shown in the following program −
* DECLARE
* message varchar2(30):= 'That''s tutorialspoint.com!';
* BEGIN
* dbms\_output.put\_line(message);
* END;
* /