Lab Session 07

***(PL/SQL)***

### PL/SQL

PL/SQL stands for procedural language-standard query language. It is a significant member of Oracle programming tool set which is extensively used to code server side programming. Similar to SQL language PL/SQL is also a **case-insensitive** programming language.

It contains the standard programming constructs you would expect from such a language, such as:

* Block Structure
* Variable & types
* Conditional Logic
* Loops
* Cursors, which holds the results returned by a query
* Procedures
* Functions
* Packages, which may be used to group procedures and functions together in one unit

### Block Structures

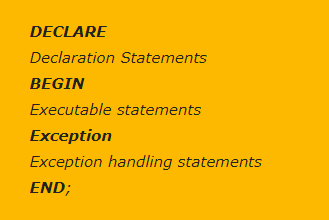
Generally a program written in PL/SQL language is divided into blocks. **We can say blocks are basic programming units in PL/SQL programming language.**

PL/SQL blocks are divided into 3 different sections which are:

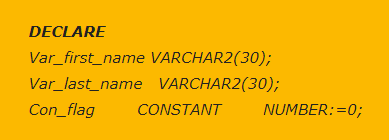
1. The Declaration Section
2. The Execution Section and
3. The Exception-handling Section

The Execution Section is the only mandatory section of block whereas Declaration and Exception Handling sections are optional.

### Basic prototype of PL/SQL Block:



**Declaration Section:**

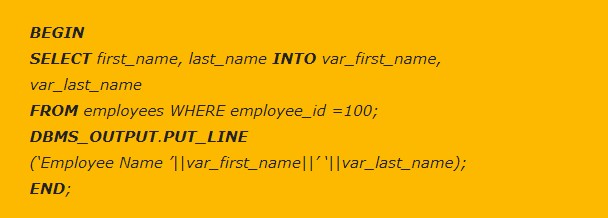
This is the first section of PL/SQL block which contains definition of PL/SQL identifiers such as variables, Constants, cursors and so on. You can say this is the place where all local variables used in the program are defined and documented.

### Execution Section:

This section contains executable statements that allow you to manipulate the variables that have been declared in the declaration section. The content of this section must be complete to allow the block to compile. By complete,mean complete set of instruction for the PL/SQL engine must be between BEGIN and END keyword.

The execution Section of any PL/SQL block always begins with the Keyword BEGIN and ends with the Keyword END.

This is the only mandatory section in PL/SQL block. This section supports all DML commands and SQL\*PLUS built-in functions.



**Exception-Handling Section:**

This is the last section of PL/SQL block which is optional like the declaration block. This section contains statements that are executed when a runtime error occurs within the block.

Runtime error occurs while the program is running and cannot be detected by the PL/SQL compiler. When a runtime error occurs, controlled is pass to the exception handling section of the block the error is evaluated and specific exception is raised.

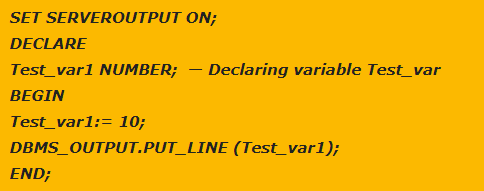
### Variables and Types:

Variable declared within the DECLARE section may only be referenced within that block. A variable declaration has both a name and a type.

#### Syntax:

[Name of the variable] [Type];

#### Example:



* **USING %TYPE keyword:**

It is used to declare a variable of the same type as a specified column in a table: SAL EMP.SAL**%TYPE**;

## Example 1

SET SERVEROUTPUT ON; DECLARE

Width INTEGER;

Height INTEGER:= 2;

Area INTEGER;

BEGIN

Area:= 6;

Width: = area/height; DBMS\_OUTPUT.PUT\_LINE (‘width = ' || width); EXCEPTION

WHEN ZERO\_DIVIDE THEN

DBMS\_OUTPUT.PUT\_LINE ('Division by Zero'); END;

#### /

**Example 2**

DECLARE

ENO EMP.EMPNO%TYPE; NAME EMP.ENAME%TYPE; BEGIN

ENO:= 2;

SELECT ENAME INTO NAME FROM EMP WHERE EMPNO=ENO;

DBMS\_OUTPUT.PUT\_LINE ('NAME OF THE EMPLOYEE IS '||NAME); END;

/

## Example 3

DECLARE

ENO EMP.EMPNO%TYPE; NAME EMP.ENAME%TYPE; BEGIN

ENO:= 2;

SELECT ENAME INTO NAME FROM EMP WHERE EMPNO=ENO;

DBMS\_OUTPUT.PUT\_LINE ('NAME OF THE EMPLOYEE IS '||NAME); EXCEPTION

WHEN NO\_DATA\_FOUND

THEN DBMS\_OUTPUT.PUT\_LINE ('NO DATA FOUND'); END;

## Example 4(Taking Input)

SET SERVEROUTPUT ON DECLARE

ENO EMP.EMPNO%TYPE; NAME EMP.ENAME%TYPE; BEGIN

ENO: = &ENO;

SELECT ENAME INTO NAME FROM EMP WHERE EMPNO=ENO;

DBMS\_OUTPUT.PUT\_LINE ('NAME OF THE EMPLOYEE IS '||NAME); EXCEPTION

WHEN NO\_DATA\_FOUND

THEN DBMS\_OUTPUT.PUT\_LINE ('NO DATA FOUND'); END;

/

### Types Of Conditional Control Statement in PL/SQL:

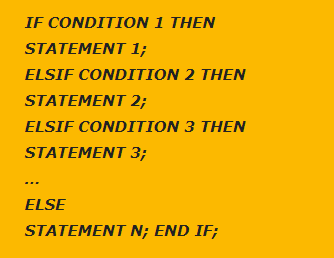
In Oracle PL/SQL we have two types of conditional control statements which are

1. IF statements and
2. CASE statements

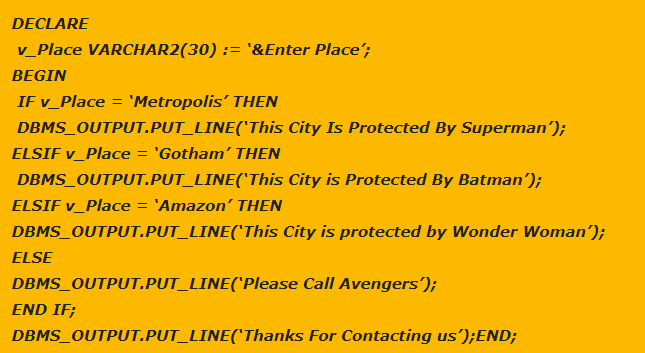
Both these statements can be further divided into different forms. For example IF statements has 3 different forms

1. IF THEN
2. [IF THEN ELSE](http://www.rebellionrider.com/if-then-else-conditional-control-statement-in-pl-sql/)
3. [IF THEN ELSEIF](http://www.rebellionrider.com/if-then-elsif-conditional-control-statement-in-pl-sql/)

#### Syntax:



**Example:**



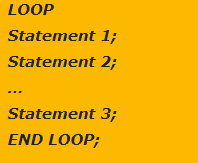
**Types of Loops in Oracle PL/SQL:**

There are 4 types of Loops in Oracle PL/SQL

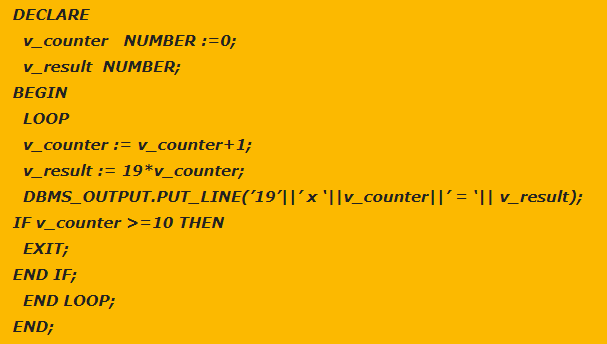
1. Simple Loop
2. While Loop
3. Numeric For Loop and
4. Cursor For loop

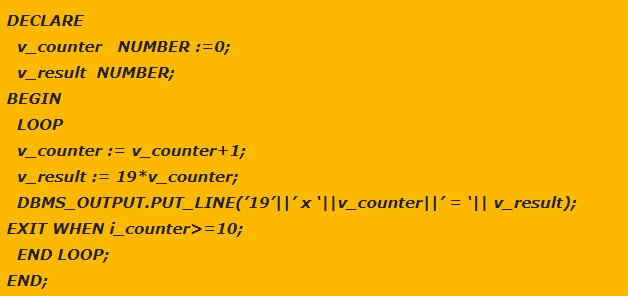
#### SIMPLE LOOP:

**Syntax:**



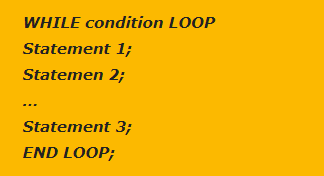
#### Example:



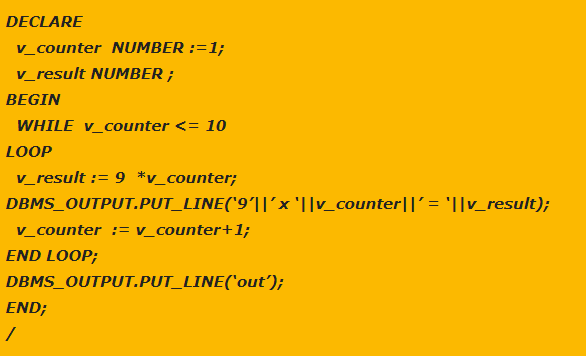


**WHILE LOOP:**

**Syntax:**

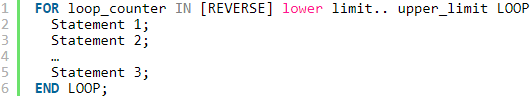


**Example:**

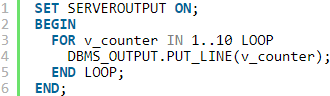


**FOR LOOP:**

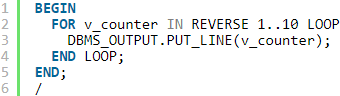
**Syntax:**



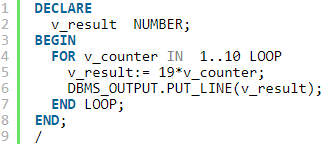
**Example 1:**



**Example 2:**



**Example 3:**

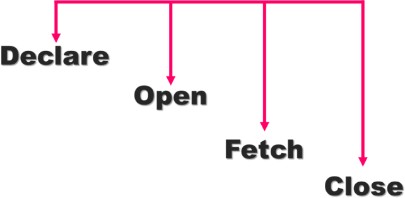


**Cursors**

Cursor is a pointer to a memory area called context area. This context area is a memory region inside the Process Global Area or PGA assigned to hold the information about the processing of a SELECT statement or DML Statement such as INSERT, DELETE, UPDATE or MERGE.

* You use cursor when you have a SELECT statement that returns more than one row from the database. A cursor is basically a set of rows that you can access one at a time.

Steps for creating Cursor:



#### STEP 1 : Declare variables to store column values:

These variables must be compatible with the column types.

#### DECLARE

v\_Empno emp.empno%TYPE; v\_Ename emp.ename%TYPE; v\_Job emp.job%TYPE;

#### STEP 2: Declare the cursor:



**STEP 3: Open the Cursor**

This step runs the SELECT statement. It must be placed in the executable section of the block (BEGIN).



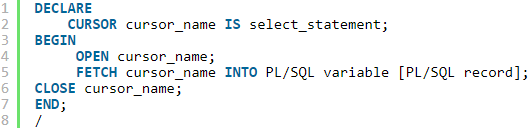
#### STEP 4: Fetch the rows from the cursor Syntax:

**STEP 5: Close the Cursor**

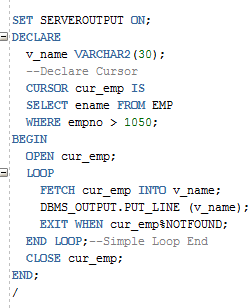
Closing your cursors frees up system resources.



Basic Programming Structure of the Cursor:



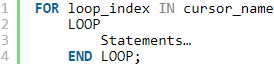
Example:



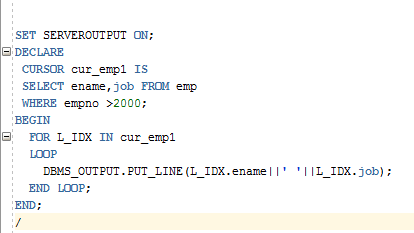
### Cursors and FOR Loops

You can use the power of FOR loop to access the rows in a cursor. When you use a FOR loop, you don’t have to explicitly open and close the cursor the FOR loop does this automatically.

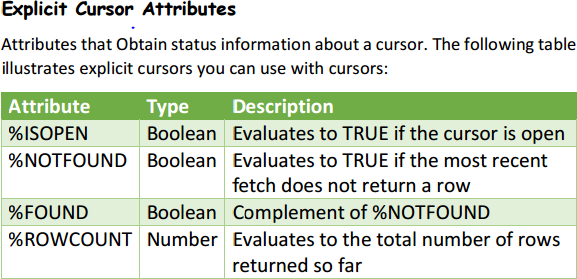
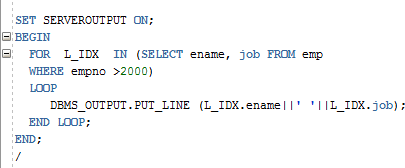
#### Syntax:



Example 1:



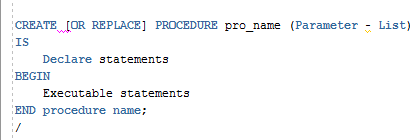
Example 2:



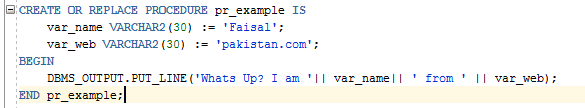
# Procedures

A **procedure** is a group of PL/SQL statements that you can call by name.

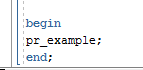
#### Syntax:



**Example:**



**Invoking the Procedure**



OR



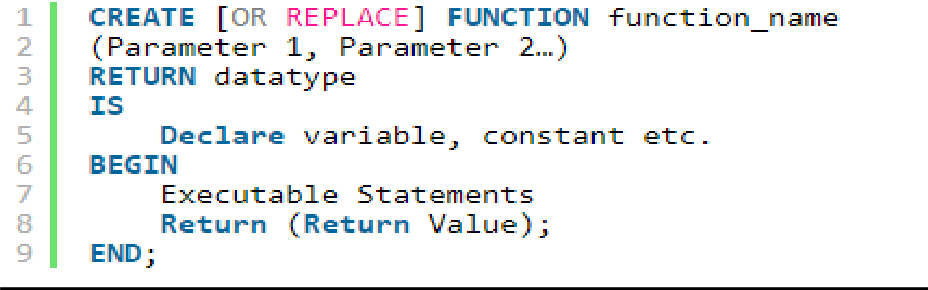
OR



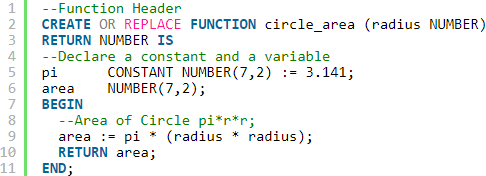
**Functions**

In Oracle Database we can define a PL/SQL function as a self-contained sub-program that is meant to do some specific well defined task. Functions are named PL/SQL block which means they can be stored into the database as a database object and can be reused.

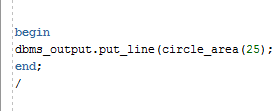
**Syntax:**



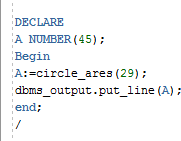
#### Example:



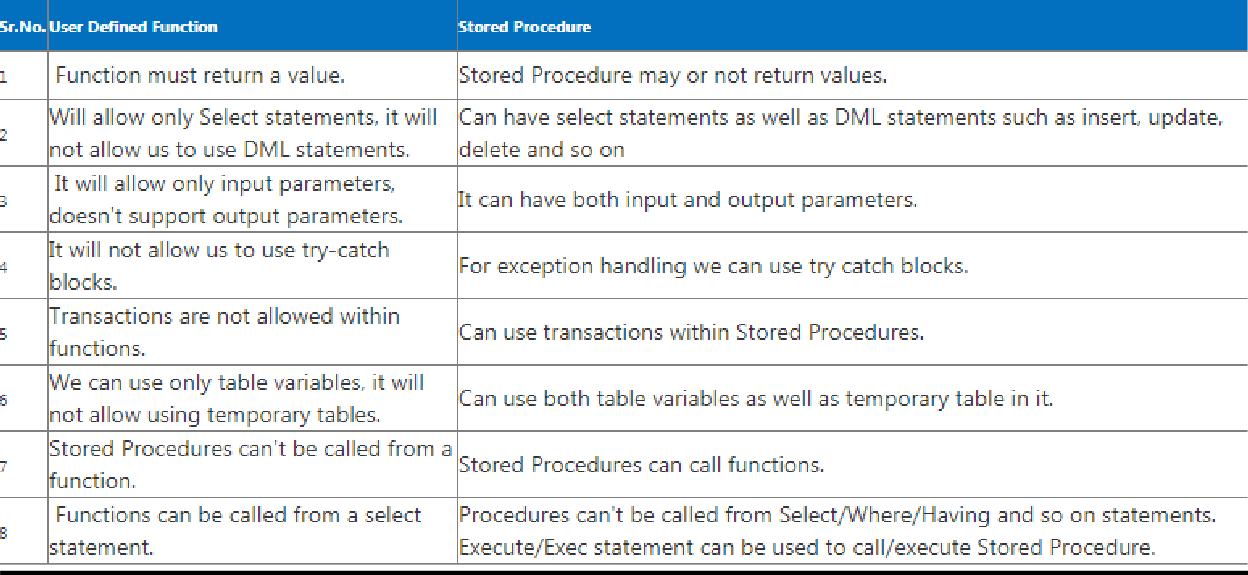
**CALLING FUNCTION:**



OR



# Difference between stored procedures and functions:



**Views**

A view is a virtual table.A view contains rows and columns, just like a real table. The fields in a view are fields from one or more real tables in the database.You can add SQL functions, WHERE, and JOIN statements to a view and present the data as if the data were coming from one single table.

#### Syntax

CREATE VIEW view\_name AS SELECT column\_name(s) FROM table\_name

WHERE condition

#### Example:

The view "Current Product List" lists all active products (products that are not discontinued) from the "Products" table.

The view is created with the following SQL:

CREATE VIEW Current\_Product\_List AS SELECT Product\_ID,Product\_Name FROM Products

WHERE Discontinued=‘NO’; SELECT \* FROM Current\_Product \_List;

#### Example:

CREATE VIEW Products\_Above\_Average\_Price AS SELECT Product\_Name,Unit\_Price

FROM Products

WHERE Unit\_Price>(SELECT AVG(Unit\_Price) FROM Products)

**View from View**

CREATE VIEW Category\_Sales\_For\_1997 AS

SELECT DISTINCT CategoryName,Sum(ProductSales) AS Category\_Sales FROM Product\_Sales \_for \_1997

GROUP BY Category\_Name;

**Exercise**

1. Write a PL/SQL code that takes two inputs from user, add them and store the sum in new variable and show the output.
2. Write a PL/SQL code that takes two inputs, lower boundary and upper boundary, then print the sum of all the numbers between the boundaries INCLUSIVE.
3. Write a PL/SQL code to retrieve the employee name, hire date, and the department name in which he works, whose number is input by the user.
4. Write a PL/SQL code to check whether the given number is palindrome or not.
5. Write a PL/SQL code that takes all the required inputs from the user for the Employee table and then insert it into the Employee and Department table in the database.
6. Write a PL/SQL code to find the first employee who has a salary over $2500 and is higher in the chain of command than employee 7499. Note: For chain, use of LOOP is necessary.
7. Write a PL/SQL code to print the sum of first 100 numbers.

#### Views:

1. Create a view, that stores information of only those employees who belongs to Accounts department.

#### Cursors:

1. Write a PL/SQL code to print out the employee information who earns more than 2000 salary.
2. Write a PL/SQL program displaying top 10 employee details

#### Procedures:

1. To Write a pl/sql program for creating a procedure for calculating sum of two numbers. Execute it as well.