Function

A named PL/SQL block having set of statement ,on which we perform mathematical calculation to perform specific task and must return a value is called Function

OR

Function is a named pl/sql block which is used to solve some particular task. We are not allowed to use DML statements in functions.

- Function is a PL/SQL object or subprogram.
- Function also like procedure.
- Function is also known as precompiled objects.
- by default functions return single value.
- Functions are executed only implicitly. Don't uses execute command, just call function.
- Function may or may not take parameters.

Every procedure having two parts:

- ***Function Specification:*** It contains the name of the function and type of the parameters.
- ***Function Body:*** It contains a declaration section, execution section and exception
  section similar to a general PL/SQL block. Here we are solving actual task.

Syntax:

```
create or replace function <fun_name> ( argument1, argument2.....)
return <datatype>
is
<declaration statement>
begin
<assigenment stmt>
<data processing stmt>
<output stmt>
return (value/varibale value);
end <fun_name>;
```

How to call function

```
fun_name (argument value...);
```

From where we will Call or execute a function

1) Using select query
2) From a procedure
3) From other function
4) From package
5) From program

Example: A function taking 3 arguments and returning arithmetic operation.

```
create or replace function cal(a in number,b in number,c in char)
return number
is
r number;
begin
if(c='+') then
r:=a+b;
elsif(c='*') then
r:=a*b;
elsif (c='-') then
r:=a-b;
else
r:=a/b;
end if;
return(r);
end cal;
/
```

Example: A function taking 2 arguments and returning multiplication of these value.

```
create or replace function fun_multi(x int, y int)
return int
is
z int;
begin
z:=x*y;
return(z);
end fun_multi;
```

1) Calling a function from select query

```
select fun_multi(12,10) from dual;
```

2) Calling a function from program

```
declare
a int;
b int;
c int;
begin
a:=&a;
b:=&b;
```

```
c:=fun_multi(a,b);
dbms_output.put_line('Result:  ' ||c);
end;
```

3) Calling a function from procedure

```
create or replace procedure proc_fun_multi ( x1 int, x2 int, x3 OUT int)
is
begin
x3:=fun_multi(x1,x2);
end proc_fun_multi;
```

Example: Write a pl/sql function for passing emp no as parameter return gross salary from
emp table bases on following condition.
Gross:= basic salary + hra + da-pf;
hra=10% of salary, da=20% of salary, pf=10% of salary

```
create or replace function fun_gross_sal (veno number)
return number
is
vsal number;
gross number;
hra number;
da number;
pf number;
begin
select sal into vsal from emp where eno=veno;
hra:=vsal*0.1;
da:=vsal*0.2;
pf:=vsal*0.1;
gross:= vsal + hra + da-pf;
return gross;
end;
/
```

Execution: select fun_gross_sal(3) from dual;
Output: 9000

Execution using program

```
Declare
z number(10);
begin
z:=fun_gross_sal(3);
dbms_output.put_line(z);
end;
/
```

Example: Function to calculate bonus

```
create or replace function f_bonus( s emp.salary%type)
return number
is
b number;
begin
if(s>=1000) then
b:=0.05*s;
elsif(s>=2000 and s<3000) then
b:=0.10*s;
elsif (s>=3000 and s<5000) then
b:=0.15*s;
else
b:=0.25*s;
end if;
return(b);
end f_bonus;
/
```

Procedure 1: Display eno,ename,sal,bounus and final salary of given eno?

```
create or replace procedure proc_empfsal(veno emp.eno%type)
is
vsal emp.sal%type;
bonus number;
fsal number;
begin
select sal into vsal from emp where eno=veno;
bonus:= f_bonus(vsal); -- function calling statement
fsal:=bonus+vsal;
dbms_output.put_line('Given Employee No:     '|| veno);
dbms_output.put_line('-----------------------------------------------');
dbms_output.put_line('Given Employee No.s Old Salary:     '|| vsal);
dbms_output.put_line('Bonus Amount:     '|| bonus);
dbms_output.put_line('-----------------------------------------------');
dbms_output.put_line('Given Employee No.s Final Salary:     '|| fsal);
end proc_empfsal;
/
```

- **Similarities between Procedure and Function**
  - Both can be called from other PL/SQL blocks.

- If the exception raised in the subprogram is not handled in the subprogram exception handling section, then it will propagate to the calling block.
- Both can have as many parameters as required.
- Both are treated as database objects in PL/SQL.

- **Difference between Procedure and Function**

parameters	Function	Procedure
Basics	Functions calculate the results of a program on the basis of the given input.	Procedures perform certain tasks in a particular order on the basis of the given inputs.
Try-Catch Blocks	Functions do not provide support for the try-catch Blocks.	Procedures provide support for the try-catch Blocks.
SQL Query	We can call a function in a SQL Query.	We cannot call a procedure in a SQL Query.
SELECT	The SELECT statements can have function calls.	The SELECT statements can never have procedure calls.
Return	A function would return the returning value/control to the code or calling function.	A procedure, on the other hand, would return the control, but would not return any value to the calling function or the code.
DML Statements	We cannot use the DML statements in a function, (functions such as Update, Delete, and Insert).	We can always use the DML statements in the case of a procedure.
Call	A function can be called using a procedure.	A procedure cannot be called using any function.
Compilation	The compilation of a function occurs when we call them in a program.	The compilation of the procedures needs to occur once, and in case it is necessary, these can be called repeatedly, and we don't have to compile them every single time.

Expression	A function must deal with expressions.	A procedure need not deal with expressions.
Explicit Transaction Handling	Functions cannot have explicit transaction handling.	Explicit transaction handling exists in the case of a procedure.

Function Types

Basically function are of two types

1) Predefined function

2) User Defined function

1. Predefined Functions

These are functions that come built-in with Oracle and are available for use without the need for user definition. They can be used directly in SQL statements or PL/SQL blocks.

Examples include:

- SYSDATE: Returns the current date and time.
- USER: Returns the current database username.
- TO_NUMBER(), TO_CHAR(), TO_DATE(): Conversion functions for different data types.

2. User-defined Functions

These are custom functions that you can create to perform specific tasks and return a single value. They are written within PL/SQL blocks or in stored packages.

- Basic Syntax for Defining Functions:

```
CREATE OR REPLACE FUNCTION function_name (para1 datatype, para2 datatype)
        RETURN return_datatype IS
BEGIN
        -- Function body
       RETURN value; -- Must return a single value
END;
```

User-defined functions can be further categorized based on their return types or usage:

- **Scalar Functions**: Return a single value, like numbers or strings.

- **Table Functions**: Return a collection or table-like structure (e.g., TABLE or VARRAY types).

3. Aggregate Functions

These functions perform calculations over a group of rows and return a single result.

Examples:

- SUM(), AVG(), COUNT(), MIN(), MAX()

4. Cursor Functions

These are used to retrieve values from cursors in PL/SQL.

Examples include:

- CURSOR%FOUND, CURSOR%NOTFOUND: Check if a cursor has returned data.
- CURSOR%ROWCOUNT: Returns the number of rows fetched by a cursor.

5. Recursive Functions

These are functions that call themselves in order to solve complex problems, such as calculating factorials or traversing hierarchical data structures.

```
create or replace function fact1(n number)
  return number
as
    b number;
  begin
     if (n=0) then
        return 1;
     else
       b:=n*fact1(n-1);
     end if;
     return b;
end;
```