FUNCTIONS

Functions are a specific piece of code which returns a value.

Points to note:

- Function must return a value.
- It can return data in both IN & IN OUT parameter.
- The return statement in function returns control to calling program & returns the result of the function.
- Functions can be called from SQL.
- Functions are considered Expressions.
- DML statements cannot be used inside functions.

Syntax:

```
Create Or Replace Function function_name (Parameter's)
Return
IS

Begin

End;
```

Executing Function

Method 1:

In SQL Plus

```
SQL> Select  Function_name() from dual;
```

```
If Parameters are passes then:
```

```
SQL> Select Function_name( Parameters) from dual;
```

By Dinesh

Method 2:

In PLSQL

```
Declare
    Variable
Begin
   Variable := function_name();   -- (or)

    Variable := Function_name( parameter);
End;
```

****

System Privileges Related To Functions	`alter any procedure`
	`create any procedure`
	`create procedure`
	`debug any procedure`
	`drop any procedure`
	`execute any procedure`

Object Privileges	`GRANT execute ON <function_name> TO <schema_name>;` Privileges to tables and views granted through roles may not be valid within a function. See the section on AUTHID under PROCEDURES.
Special Restrictions	Functions called from SQL have special restrictions • Stored in database • Must own or have EXECUTE privilege • When used in SELECT statement - cannot contain DML • When used in UPDATE or DELETE - cannot SELECT or perform DML on the same table

By Dinesh

Tables Used

User_1:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

NAME	ID
dinesh	111
anand	112
srikanth	113
sudhir	114
lokesh	115

Utemp:

Data | Explain Plan | Auto Trace | DBMS Output | Code Statistics | Script Output

TNAME	DEPT
vinush	seibe
senthil	apps
sanjay	ibm
srikanth	apps

Calling a Function

Method 1:

```
select function_name() from dual;
```

Method 2:

```
declare
res datatype;  -- data type which is returned by function

begin

Res: = function_name();
dbms_output.put_line ('Enter radius of circle:'||res);

end;
```

By Dinesh

Functions Without Parameters:

-- Simple function

```
Create or replace function hello return VARCHAR2 IS

BEGIN

  return 'hello world';

END hello;
```

-- Function using some calculations

```
Create or replace function f1 return number
As
   r number;
   pi float default 3.14;
   aoc float;
Begin
   dbms_output.put_line('enter radius of circle :');
   r:= &r;
   aoc := (pi * r * r);
      return aoc;
End;
```

Run:

```
SQL> select f1() from dual;
```

-- Compiling the above function inside anonymous block:

```
declare
Res float;
Begin
Res: =f1();
   dbms_output.put_line ('Enter radius of circle :'||res);
End;
```

NOTE:

Function can return only one value. Therefore you cannot use dbms_output.put_line() inside function.

By Dinesh

Using %TYPE

-- Program using function for SELECT CLAUSE

```
Create or replace function u1 return user_1.name%TYPE is
uname user_1.name%TYPE;
Begin
      Select name into uname from user_1 where id = 113;
      Return uname;
End u1;
```

Running:

```
SQL > Select user_1() from dual;
```

Running2 (anonymous block) :

```
Declare
Res char (20);
Begin
Res: =u1();
dbms_output.put_line ('result name : ' || res);
End;
```

-- Program using above Function for INSERT CLAUSE

```
Begin
       insert into utemp values(u1,'apps');
End;
```

Verify output : `select * from utemp`

-- The u1() function is user in WHERE CLAUSE

Query: `select * from utemp where tname = u1`

By Dinesh

-- The function u1 is used in VIEW

VIEW:

```
Create or replace view v_user as
select * from utemp where tname = u1
```

Verify Output: `select * from v_user`

Functions With Parameter

-- The Program is By passing parameter in function

```
create or replace function f1(r float) return float
as
   pi float default 3.14;
   aoc float;
begin
   aoc := (pi * r * r);
   return aoc;
end;
```

RUN:

```
SQL> select f1( 5 ) from dual;
```

Functions using IN, OUT Parameters

By Dinesh

-- Simple program using IN parameter

```
Create or replace function check_1(st IN char) return boolean is
Begin
           if LENGTH(st) = 3 then
                  return TRUE;

               else
                    return FALSE;
               end if;

End;
```

Running :

```
Declare
Res boolean;
Begin

 Res:= check_1('aaaa');

 If res = true then
  dbms_output.put_line( 'String matched ');

 Else
  dbms_output.put_line( 'String not matched ');
 End if;
End;
```

OUTPUT:

```
String not matched
```

By Dinesh

-- Simple program using OUT parameter

```
Create or replace function u2 (msg OUT VARCHAR2) return varchar2 is

Begin
  msg:= 'testing hello world';
   Return msg;
End;
```

Running:

```
Declare
Temp varchar2(20);
Res varchar2(20);
Begin
Res := u2(temp);
dbms_output.put_line( 'Temp variable :' || temp);
dbms_output.put_line( 'Result :' || res);
End;
```

Output:

```
Temp variable :testing hello world
Result :testing hello world
```

-- Simple program to implement IN OUT parameters

```
CREATE OR REPLACE FUNCTION fns(a IN NUMBER, b OUT NUMBER) RETURN NUMBER
    BEGIN
    dbms_output.put_line(a);
    b:= a;
    RETURN a;
END fns;

SQL> variable tmp number;

SQL>  declare
res number;
begin
res:= fns(10,:tmp);
dbms_output.put_line('result =' || res);
end;
```

OUTPUT:

10
Result = 10

By Dinesh