STORED FUNCTIONS (OR) USER DEFINED FUNCTIONS:

* these functions are created by user explicitely.It is a named block. It will accept value   from user, perform some task and it must return a value.

* The main difference between a function & procedure is the function must return a value and the procedure may or may not return a value.

syntax:

```
 Create or replace function <function_name>[(arugment mode datatype,
                                  argument mode datatype,....)]
                  Return <Datatype>
                     is
<Declare Variables>
 Begin
  <exec-statements>;
  Return (value);
 Exception
  <exec-statements>;
  Return (value);
 End <function_name>;
```

EX: create a sf to input department name and return sum of salary of department?

```
FUNCTION SF1(p_DNAME VARCHAR2)
RETURN NUMBER
AS
v_TOTSAL NUMBER(10);
BEGIN
SELECT SUM(SAL) INTO v_TOTSAL FROM EMP E,DEPT D
WHERE E.DEPTNO=D.DEPTNO AND DNAME=p_DNAME;
RETURN v_TOTSAL;
END;
SQL> SELECT SF1('SALES') FROM DUAL;
```

EX: create a sf to return no.of employee in between given dates?

```
FUNCTION SF2(SD DATE,ED DATE)
RETURN NUMBER
AS
v_COUNT NUMBER(10);
BEGIN
SELECT COUNT(*) INTO v_COUNT FROM EMP
WHERE HIREDATE BETWEEN SD AND ED;
RETURN v_COUNT;
END;
SQL> SELECT SF2('01-JAN-81','31-DEC-81') FROM DUAL;
```

EX: create a sf to input employee number and return that employee gross salary as per given conditions are

- **i) hra -------- 10%**
- **ii) da -------- 20%**
- **iii) pf --------10%.**

```
FUNCTION SF3(p_EMPNO NUMBER)
RETURN NUMBER
AS
v_BSAL NUMBER(10);
v_HRA NUMBER(10);
v_DA NUMBER(10);
v_PF NUMBER(10);
v_GROSS NUMBER(10);
BEGIN
SELECT SAL INTO v_BSAL FROM EMP WHERE EMPNO=p_EMPNO;
v_HRA:=v_BSAL*0.1;
v_DA:=v_BSAL*0.2;
v_PF:=v_BSAL*0.1;
```

```
v_GROSS:=v_BSAL+v_HRA+v_DA+v_PF;
RETURN v_GROSS;
END;
SQL> SELECT SF3(7788) FROM DUAL;
```

Ex: Write a function to find simple interest.

```
Create or Replace Function SI(P Number, T Number, R Number)
                    Return Number
                         is
Simple_Int Number;
Begin
m Simple_Int:=(P*T*R)/100;
 Return (Simple_Int);
End Si;
```

- Generally Functions are Executed by using 'SELECT' statement.

```
Select SI(1000,2,10) from dual;
```

Ex: create a sf to find experience of given employee ?

```
 Create or replace function Emp_Exp(Tempno Emp.Empno%type)
                    Return Varchar2
                         is
 Tdate Emp.Hiredate%type;
 Texp Number;
 Begin
  select Hiredate into Tdate from Emp
                                      where Empno=Tempno;
  Texp:=round((sysdate-tdate)/365);
  Return(Tempno||' Employee Experience is '||Texp||' Years.');
 Exception
  when No_data_found then
  Return('Given Employee Record Not Found.');
 End Emp_Exp;
SQL> SELECT EMP_EXP(7788) FROM DUAL;
SQL> SELECT EMP_EXP(EMPNO) FROM EMP;
```

==

Ex: Write a function to increment Employee Salaries based on their Experiences.

```
        if exp>=30 -> 50%
        exp>=25 -> 30%
        exp>=20 -> 20%
        exp<20 -> 10%
```

audit_Emp table: empno,ename,job,hiredate,sal,expe,increment_sal,Netsal?

==

Function for to calculate employee Experience:

```
 Create or Replace Function Emp_Expe(Tempno Emp.Empno%type)
                  return Number
                     is
 Texp number;
 Begin
  select round((sysdate-hiredate)/365) into Texp from Emp
                                                  where empno=Tempno;
  Return(Texp);
 End Emp_Expe;
```

Create a table to store incremented Employee Details.

```
   Create table Audit_Emp(Empno Number(4),Ename Varchar2(10),
                  Job Varchar2(10), Hiredate Date,
                  sal Number(7,2), Expe Number(3),
                  incr_sal Number(7,2),Netsal Number(8,2),
                  Deptno Number(2));
```

Procedure for to calculate Increment Salary:

```
Create or Replace Procedure Emp_Proc
            is
Cursor Emp_cur is Select Empno,Ename,Job,Hiredate,Sal,Deptno From Emp;
e emp_cur%rowtype;
texp number;
incr_sal Number;
Begin
 Open Emp_cur;
 Loop
  Fetch Emp_cur into e;
  Exit when Emp_cur%notfound;
  Texp:=Emp_Expe(e.empno);
  if Texp>=30 then
    incr_sal:=E.sal*50/100;
  elsif Texp>=25 then
    incr_sal:=E.sal*30/100;
  elsif Texp>=20 then
    incr_sal:=E.sal*20/100;
  elsif Texp<20 then
    incr_sal:=E.sal*10/100;
  end if;
  insert into Audit_Emp values (e.empno,e.ename,e.job,e.hiredate,e.sal,Texp,
              incr_sal,(e.sal+Incr_sal),e.deptno);
 End Loop;
 dbms_output.put_line(emp_cur%rowcount||' Employees Salaries Incremented.');
End Emp_Proc;
```

Note:

All Functions are stored in user_objects.

All Functions Bodies are stored in 'user_source' system table.

To see the function body

Ex:

```
sql> select text from user_source where name='EMP_EXPE';
```

Dropping Functions

syntax:

```
sql> Drop Function <function_name>;
```

Ex:

```
sql> Drop Function Emp_Expe;
```