Functions

Functions are program units that execute one or more statements and return a value through the RETURN clause. A function must have at least one RETURN statement. Functions can be called within any expression (considering data type), initialization, in SQL (some limitation), as arguments.

Parameters

Procedures, functions, and cursors may have parameters. Each parameter is defined by its name, datatype, mode (IN, OUT, IN OUT), and optional default value. Datatype - any PL/SQL datatype or **programm**er-defined datatype, without **constrain of size**. The actual size of parameter is determined from the calling program. Example: NUMBER is ok NUMBER(10) is invalid procedure data type.

Parameters are called: formal and actual. Formal parameters are the names that are declared in the header of the procedure or function. Actual parameters are values or expressions placed in the parameter list when procedure or function is called. Arguments may be passed using: positional or named notation.

CALC_GPA ('S198888', SYSDATE)   CALC_GPA(date=>SYSDATE, student_id => 'S198888')

CREATE [OR REPLACE] FUNCTION *function_name*
[(*param_name* [IN | out |IN OUT] *formal_data_type* [, …]   )]
RETURN *formal_data_type*
(IS | AS)
  Declaration of variables
BEGIN
  function_body;
END

Exercise:

Table E_CLIENT exists in the database . The following are the results from `describe e_client command`:

```
Name      Null     Type
--------- -------- ------------
CLIENT_ID NOT NULL CHAR(10)
C_NAME    NOT NULL VARCHAR2(20)
DOB       NOT NULL DATE
C_CITY             VARCHAR2(20)
```

Write a PL/SQL function to calculate the age of a client. The input parameter is the client_id. The function returns the number of **whole years** (use TRUNC function or, if you prefer, FLOOR function or any other valid method to calculate a whole number; use SYSDATE for the current date).
If the input parameter is invalid, i.e., the client_id does not exist in the database, the function should return -1.

```
CREATE OR REPLACE
  FUNCTION calc_client_age(
      p_cid CHAR)
    RETURN INTEGER
  IS
    V_Age   INTEGER := 0;
    v_count INTEGER := 0;
  BEGIN
    SELECT COUNT(*) INTO V_Count FROM E_Client WHERE Client_Id =
P_Cid;
    IF V_Count = 0 THEN
      V_Age   := -1;
    ELSE
      SELECT TRUNC(Months_Between(Sysdate, Dob)/12, 0)
      INTO V_Age
      FROM E_Client
      WHERE Client_Id = P_Cid;
    END IF;
    RETURN v_age;
  END;
```

Testing:

```
Select 'abc' "Client ID", Calc_Client_Age ('abc') "Age" From Dual;

select 'C000000001' "Client ID", calc_client_age ('C000000001')
"Age" from dual;

Client ID        Age
--------- ----------
abc               -1

Client ID         Age
---------- ----------
C000000001         68
```