EXPERIMENT: 6

Aim: Develop a program that includes the features NESTED IF, CASE and CASE expression. The program can be extended using the NULLIF and COALESCE functions

NESTED IF:

Nested if-then statements mean an if statement inside another if statement

Syntax:-

```
if (condition1) then
        -- Executes when condition1 is true
        if (condition2) then
                -- Executes when condition2 is true
        end if;
end if;
```

PL/SQL Program to find biggest of three number using nested if

```
SQL> set serveroutput on
SQL> declare
  2    a number:=10;
  3    b number:=12;
  4    c number:=5;
  5  begin
  6    dbms_output.put_line('a='||a||' b='||b||' c='||c);
  7    if a>b AND a>c then
  8          dbms_output.put_line('a is greatest');
  9    else
 10          if b>a AND b>c then
 11                dbms_output.put_line('b is greatest');
 12          else
 13                dbms_output.put_line('c is greatest');
 14          end if;
 15    end if;
```

```
16  end;
17  /
```

Output:

a=10 b=12 c=5

b is greatest

PL/SQL procedure successfully completed.

CASE and CASE Expression

- CASE statement selects one sequence of statements to execute.
- CASE Statement is used to handle flow control within procedural code and it determines which code block to execute based on specified conditions.
- The CASE statement uses a selector rather than multiple Boolean expressions. A selector is an expression, the value of which is used to select one of several alternatives.
- CASE Expression in SQL is used for transforming or selecting values within a query and returning different results based on conditions.

Syntax

```
CASE selector
        WHEN 'value1' THEN S1;
        WHEN 'value2' THEN S2;
        WHEN 'value3' THEN S3;
        ...
        ELSE Sn; -- default case
END CASE;
```

```
SQL> set serveroutput on;
SQL> declare
  2  grade char(1);
  3  begin
  4  grade:='&grade';
```

```
 5  case
 6  when grade='a' then
 7  dbms_output.put_line('Excellent');
 8  when grade='b' then
 9  dbms_output.put_line('very good');
10  when grade='c' then
11  dbms_output.put_line('good');
12  when grade='d' then
13  dbms_output.put_line('fair');
14  when grade='f' then
15  dbms_output.put_line('poor');
16  else
17  dbms_output.put_line('No such grade');
18  end case;
19  end;
20  /
Enter value for grade: a
old   4: grade:='&grade';
new   4: grade:='a';
Excellent

PL/SQL procedure successfully completed.

SQL> /
Enter value for grade: d
old   4: grade:='&grade';
new   4: grade:='d';
fair

PL/SQL procedure successfully completed.
```

SQL> /

Enter value for grade: e

old 4: grade:='&grade';

new 4: grade:='e';

No such grade

PL/SQL procedure successfully completed.

NULLIF:

Takes two arguments. If the two arguments are equal, then NULL is returned. otherwise the first argument is returned.

Syntax:

select column_name, NULLIF(argument1,arguement2) from table_name;

SQL> select *from emp;

ENO	ENAME	LOC	SALARY
101	ali	vja	15000
102	ravi	hyd	25000
103	raju	gnt	35000
103	raju	gnt	35000

SQL> select ename, nullif('ali','ali') from emp;

ENAME	NUL
ali	
ravi	
raju	
raju	

SQL> select ename, nullif('ali','ali1') from emp;

ENAME NUL

---------- ---

ali ali

ravi ali

raju ali

raju ali

COALESCE:

COALESCE () function accepts a list of arguments and returns the first one that evaluates to a non-null value.

Syntax: coalesce("expression1","expression2",...);

SQL> select coalesce(NULL,'CRRCOE','IT') from dual;

COALES

CRRCOE