EXPERIMENT: 7

Aim: Program development using WHILE LOOPS, numeric FOR LOOPS, nested loops using ERROR Handling, BUILT –IN Exceptions, USE defined Exceptions, RAISEAPPLICATION ERROR

WHILE LOOP: A WHILE LOOP statement repeatedly executes a target statement as long as a given condition is true.

Syntax:

```
WHILE condition LOOP
        sequence_of_statements
END LOOP;
```

A PL/SQL Program to find sum of ODD number upto given number using While loop

```
SQL> set serveroutput on;
SQL> declare
  2              inval number;
  3              endval number;
  4              s number default 0;
  5    begin
  6              inval:=1;
  7              endval:=&endval;
  8              while inval<endval loop
  9                      s:=s+inval;
 10                      inval:=inval+2;
 11             end loop;
 12     dbms_output.put_line('sum of odd numbers between 1 and '||endval||' is '|| s);
 13 end;
 14 /
Enter value for endval: 50
old   7: endval:=&endval;
new   7: endval:=50;
sum of odd numbers between 1 and 50 is 625

PL/SQL procedure successfully completed.
```

```
Enter value for endval: 100
old   7: endval:=&endval;
new   7: endval:=100;
sum of odd numbers between 1 and 100 is 2500

PL/SQL procedure successfully completed.
```

FOR Loop: A FOR LOOP is a repetition control structure that allows us to efficiently write a loopthat needs to execute a specific number of times.

Syntax

```
FOR counter IN initial_value ..final_value LOOP
        sequence_of_statements;
END LOOP;
```

A PL/SQL code to print multiplication table using for loop

```
SQL> set serveroutput on;
SQL> declare
  2              var1 number;
  3               var2 number;
  4    begin
  5              dbms_output.put_line('Enter number to print multiplication table');
  6              var1:=&var1;
  7              for var2 in 1..10 loop
  8                      dbms_output.put_line(var1||'x'||var2||'='||var1*var2);
  9              end loop;
 10  end;
 11  /
Enter value for var1: 3
old   6: VAR1:=&VAR1;
new   6: VAR1:=3;
Enter number to print multiplication table
3X1=3
3X2=6
```

3X3=9

3X4=12

3X5=15

3X6=18

3X7=21

3X8=24

3X9=27

3X10=30

PL/SQL procedure successfully completed.

NESTED LOOP: Nested loop means one loop inside another loop. It may be either basic, while or for loop.

Syntax:

```
WHILE condition1 LOOP
        sequence_of_statements1
        WHILE condition2 LOOP
                sequence_of_statements2
         END LOOP;
END LOOP;
```

PL/SQL program to print n prime number using nested loop

```
SQL> declare
 2              i number(3);
 3              j number(3);
 4    begin
 5              i := 2;
 6              LOOP
 7                      j:= 2;
 8                      LOOP
 9                              exit WHEN ((mod(i, j) = 0) or (j = i));
10                       j := j +1;
11              END LOOP;
12              IF (j = i ) THEN
13                      dbms_output.put_line(i || ' is prime');
```

```
 14              END IF;
 15              i := i + 1;
 16                      exit WHEN i = 50;
 17              END LOOP;
 18  END;
 19  /
2 is prime
3 is prime
5 is prime
7 is prime
11 is prime
13 is prime
17 is prime
19 is prime
23 is prime
29 is prime
31 is prime
37 is prime
41 is prime
43 is prime
47 is prime

PL/SQL procedure successfully completed.
```

Syntax for Exception Handling

The General Syntax for exception handling is as follows. The default exception will be handled using WHEN others THEN:

```
DECLARE
        <declarations section>
BEGIN
        <executable command(s)>
EXCEPTION
        <exception handling goes here >
WHEN exception1 THEN
```

```
        exception1-handling- statements
WHEN exception2 THEN
        exception2-handling-statements
WHEN exception3 THEN
        exception3-handling-statements
.....
WHEN others THEN
        exception3-handling-statements
END;
```

```
SQL> set serveroutput on;
SQL> declare
  2    s_sid student.sid%type:= 506;
  3    s_name student.sname%type;
  4    s_rank student.rank%type;
  5  begin
  6    select sname,rank into s_name,s_rank from student where sid=s_sid;
  7   dbms_output.put_line('Name:' || s_name);
  8   dbms_output.put_line('Rank:' || s_rank);
  9  exception
 10     when no_data_found THEN
 11              dbms_output.put_line('No such student!');
 12   WHEN others THEN
 13             dbms_output.put_line('Error!');
 14  end;
 15  /

No such student!

PL/SQL procedure successfully completed.
```

Raising Exceptions

Exceptions are raised by the database server automatically whenever there is any internal database error, but exceptions can be raised explicitly by the programmer by using the command RAISE.

Syntax:

```
DECLARE
        exception_name EXCEPTION;
BEGIN
        IF condition THEN
                RAISE exception_name;
        END IF;
EXCEPTION
        WHEN exception_name THEN
                        statement;
 END;
```

User-defined Exceptions

PL/SQL allows you to define your own exceptions according to the need of your program. A user-defined exception must be declared and then raised explicitly, using either a RAISE statement or the procedure DBMS_STANDARD.RAISE_APPLICATION_ERROR.

The syntax for declaring an exception is:

```
DECLARE
        my-exception EXCEPTION;
SQL> set serveroutput on;
SQL> declare
                s_sid student.sid%type:= &ss_id;
                s_name student.sname%type;
                s_rank student.rank%type;
        ex_invalid_id EXCEPTION;
   begin
     if s_sid<=0 then
            raise ex_invalid_id;
      else
          select sname,rank into s_name,s_rank from student where sid=s_sid;
            dbms_output.put_line('Name:' || s_name);
```

```
            dbms_output.put_line('Rank:' || s_rank);
    end if;
  exception
    when ex_invalid_id then
          dbms_output.put_line('ID must be greater than zero');
   when no_data_found then
          dbms_output.put_line('No such student!');
    WHEN others THEN
          dbms_output.put_line('Error!');
  end;
  /
Enter value for ss_id: 510
old   2:              s_sid student.sid%type:= &ss_id;
new   2:               s_sid student.sid%type:= 510;
No such student!

PL/SQL procedure successfully completed.

SQL> /
Enter value for ss_id: 0
old   2:              s_sid student.sid%type:= &ss_id;
new   2:               s_sid student.sid%type:= 0;
ID must be greater than zero

PL/SQL procedure successfully completed.
```