NAME: ROHAN KAMBLE
CLASS: SYBSC-CS
PRN: 202042002

DBMS

PRACTICAL NO 1

Aim: To write Anonymous PL/SQL Block with basic programming construct by including following:

a. Sequential Statements
b. unconstrained loop

Theory :

1a) Sequential Statements: In PL/SQL EXIT, CONTINUE, GOTO Statements (Sequential Control Statements) are used to control your iteration loop.

- EXIT Statement: This statement is used to exit the loop.
- EXIT WHEN Statement: This statement is used to exit, when WHEN clauses condition true.
- CONTINUE Statement: to skip the current iteration with in loop. ·

CONTINUE WHEN Statement: to skip the current iteration with in loop when WHEN clauses condition true.

· GOTO Statement: Transfers the program execution flow unconditionally.

EXIT Statement

EXIT statement unconditionally exits the current loop iteration and transfer control to end of current loop. EXIT statement writing syntax,

Syntax

```
[ label_name ] LOOP
 statement(s);
 EXIT;
END LOOP [ label_name ];
```

Example Code No. 1.1

```
DECLARE
 no NUMBER := 5;
BEGIN
        LOOP
                DBMS_OUTPUT.PUT_LINE ('Inside value: no = ' || no);
 no := no -1;
 IF no = 0 THEN
 EXIT;
 END IF;
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Outside loop end'); -- After EXIT control transfer this statement
END;
/
```

EXIT WHEN Statement

EXIT WHEN statement unconditionally exit the current loop iteration when WHEN clause condition true. EXIT WHEN statement writing syntax,

Syntax

```
[ label_name ] LOOP
 statement(s);
 EXIT WHEN condition;
END LOOP [ label_name ];
```

Example Code No. 1.2

```
SQL>DECLARE
```

```
        i number:=0;
BEGIN
        LOOP
                dbms_output.put_line('Hello');
                i:=i+1;
                EXIT WHEN i>5;
        END LOOP;
END;
/
```

GOTO Statement

GOTO statement unconditionally transfer program control. GOTO statement writing syntax,

Syntax

```
GOTO code_name
        ----------
        ----------
<<code_name>>
----------
```

Example Code No. 1.3

```
SQL>BEGIN
FOR i IN 1..5 LOOP
        dbms_output.put_line(i);
        IF i=4 THEN
                GOTO label1;
        END IF;
END LOOP;
<<label1>>
DBMS_OUTPUT.PUT_LINE('Row Filled');
END;
/
```

1b) Unconstrained loop

LOOP Statements

Loop statements run the same statements with a series of different values. The loop statements are:

- Basic LOOP
- FOR LOOP
- Cursor FOR LOOP
- WHILE LOOP

The statements that exit a loop are:

· EXIT

· EXIT WHEN

The statements that exit the current iteration of a loop are:

· CONTINUE ·

CONTINUE WHEN

· EXIT, EXIT WHEN, CONTINUE, and CONTINUE WHEN and can appear anywhere inside a loop, but not outside a loop.

Basic LOOP Statement : The basic LOOP statement has this structure:

```
[ label ] LOOP
 statements
END LOOP [ label ];
```

With each iteration of the loop, the statements run and control returns to the top of the loop. To prevent an infinite loop, a statement or raised exception must exit the loop.

EXIT Statement

The EXIT statement exits the current iteration of a loop unconditionally and transfers control to the end of either the current loop or an enclosing labeled loop.

Example Code No. 1.4

```
DECLARE
 x NUMBER := 0;
BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
x := x + 1;
 IF x > 3 THEN
 EXIT;
 END IF;
 END LOOP;
 -- After EXIT, control resumes here
 DBMS_OUTPUT.PUT_LINE(' After loop: x = ' || TO_CHAR(x));
END;
/
```

EXIT WHEN Statement

The EXIT WHEN statement exits the current iteration of a loop when the condition in its WHEN clause is true, and transfers control to the end of either the current loop or an enclosing labeled loop.

Example Code No. 1.5

```
DECLARE
 x NUMBER := 0;
BEGIN
 LOOP
 DBMS_OUTPUT.PUT_LINE('Inside loop: x = ' || TO_CHAR(x));
x := x + 1; -- prevents infinite loop
 EXIT WHEN x > 3;
 END LOOP;
 -- After EXIT statement, control resumes here
 DBMS_OUTPUT.PUT_LINE('After loop: x = ' || TO_CHAR(x));
END;
/
```

Nested, Labeled Basic LOOP Statements with EXIT WHEN Statements Example Code No. 1.6

```
DECLARE
 s PLS_INTEGER := 0;
 i PLS_INTEGER := 0;
 j PLS_INTEGER;
BEGIN
 <<outer_loop>>
 LOOP
 i := i + 1;
 j := 0;
 <<inner_loop>>
 LOOP
 j := j + 1; s := s + i * j; -- Sum
 several products
 EXIT inner_loop WHEN (j > 5);
 EXIT outer_loop WHEN ((i * j) > 15);
 END LOOP inner_loop;
 END LOOP outer_loop;
 DBMS_OUTPUT.PUT_LINE
 ('The sum of products equals: ' || TO_CHAR(s));
END;
/
```

Nested, Unabeled Basic LOOP Statements with EXIT WHEN Statements Example Code No. 1.7

```
DECLARE
 i PLS_INTEGER := 0;
 j PLS_INTEGER := 0;

BEGIN
 LOOP
```

```
 i := i + 1;
 DBMS_OUTPUT.PUT_LINE ('i = ' || i);

 LOOP
 j := j + 1;
 DBMS_OUTPUT.PUT_LINE ('j = ' || j);
 EXIT WHEN (j > 3);
 END LOOP;

 DBMS_OUTPUT.PUT_LINE ('Exited inner loop');

 EXIT WHEN (i > 2);
 END LOOP;

 DBMS_OUTPUT.PUT_LINE ('Exited outer loop');
END;
/
```

~ * ~ * ~ * ~ * ~ * ~

PRACTICAL NO 2

1)IF-THEN

```
SQL> create table emp (
        empno integer, sal
        number(8,2)
       )

SQL> insert into emp values(1,10000);
SQL> insert into emp values(2,20000);
SQL> insert into emp values(3,5000);

declare e_id
  emp.empno%type:=&e_id;
  e_sal emp.sal%type; begin
  select sal into e_sal from emp
  where empno=e_id;
  if(e_sal<=10000) then update
  emp
```

```
 set sal=sal+2000 where empno=e_id;

dbms_output.put_line('salary updated');

end if; end;
```

OUTPUT:- Enter value for e_id: 7902

salary updated

PL/SQL procedure successfully completed

---

2)IF-THEN-ELSE

```
declare a

number(10):=&a;

begin if a<20 then

 DBMS_OUTPUT.PUT_LINE('a is less than 20');

else
DBMS_OUTPUT.PUT_LINE('a is not less than 20');

end if;

end;

/
```

OUTPUT:- Enter value for a: 15

a is less than 20

PL/SQL procedure successfully completed.

SQL> /

Enter value for a: 21

a is not less than 20

PL/SQL procedure successfully completed.

---

3)IF-THEN-ELSIF

```
declare paytype
varchar2(10):='&paytype'; begin

IF(paytype='chb')THEN
dbms_output.put_line('clock hourly basis');
ELSIF(paytype='s')THEN
```

```
 dbms_output.put_line('Full              Time');

 ELSIF(paytype='conb')THEN

 dbms_output.put_line('contract basis');

ELSE

dbms_output.put_line('Invalid type');

END IF;

EXCEPTION

WHEN OTHERS THEN

dbms_output.put_line('Error occured');

END;

/
```

o/p:- Enter value for paytype: s old 2:

```
paytype varchar2(10):='&paytype'; new

2: paytype varchar2(10):='s';

Full Time
PL/SQL procedure successfully completed.
```

---

4) SIMPLE CASE STATEMENT

```
DECLARE
  grade CHAR(1);
BEGIN
  grade := 'B';

  CASE grade
    WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
    WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
    WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
    WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
    WHEN 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
    ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
  END CASE;
END;
/
```

OUTPUT:-Very Good

PL/SQL procedure successfully completed.

---

5)SEARCHED CASE EXPRESSION

```
DECLARE
  percent number(2);
BEGIN
  percent:=&percent;

  CASE
    WHEN percent >= 70 THEN DBMS_OUTPUT.PUT_LINE('A+ grade');
    WHEN percent between 60 and 70 THEN DBMS_OUTPUT.PUT_LINE('A
grade');
    WHEN percent between 55 and 60 THEN DBMS_OUTPUT.PUT_LINE('B
grade');
    WHEN percent between 50 and 55 THEN DBMS_OUTPUT.PUT_LINE('C
    grade');
    WHEN percent between 45 and 50 THEN DBMS_OUTPUT.PUT_LINE('D
grade');
    WHEN percent between 40 and 45 THEN DBMS_OUTPUT.PUT_LINE('E
grade');

ELSE
 DBMS_OUTPUT.PUT_LINE('No such percentage');
  END CASE;
```

```
END;
/
```

OUTPUT:- Enter value for percent: 70

A+ grade

PL/SQL procedure successfully completed.

PRACTICAL NO 3

AIM: WRITING PL/SQL BLOCKS FOR ITERATIVE STRUCTURES

FOR LOOP:-

1} WAP to accept a string and count how many vowels present in the string

```
DECLARE
V VARCHAR2(300):='&V';
CNT NUMBER(5):=0;
C CHAR;
BEGIN
FOR i IN 1..LENGTH(V)
LOOP
C:=SUBSTR(V,i,1);
```

```
IF C IN ('A','E','I','O','U') THEN

CNT:=CNT+1;

END IF;

END LOOP;

DBMS_OUTPUT.PUT_LINE('NO OF VOWELS PRESENT = '||CNT);

END;

/
```

OUTPUT:- Enter value for v: a

NO OF VOWELS PRESENT = 0

PL/SQL procedure successfully completed.

SQL> /

Enter value for v: A

NO OF VOWELS PRESENT = 1

PL/SQL procedure successfully completed.

SQL> /

Enter value for v: ABHI

NO OF VOWELS PRESENT = 2

PL/SQL procedure successfully completed. **2}** DECLARE

```
  step  PLS_INTEGER := 5;
BEGIN
  FOR i IN 1..3 LOOP
    DBMS_OUTPUT.PUT_LINE (i*step);
  END LOOP;
END;
/
```

OUTPUT:- 5

10

15

PL/SQL procedure successfully completed.

3} declare

```
    n number; fac
    number:=1; i
    number;

begin
    n:=7;

    for i in 1..n
    loop
    fac:=fac*i; end
    loop;

    dbms_output.put_line('factorial='||fac);
end;
/
```

OUTPUT :- Enter value for n: 7

factorial=5040

PL/SQL procedure successfully completed.

4} begin

<<outer_loop>>

for i IN 1..3 loop

<<inner_loop>>

for j IN 1..3 loop

dbms_output.put_line('i is= '||i||' j is= '||j);

end loop inner_loop; end loop

outer_loop; end;

OUTPUT :-

i is= 1 j is= 1

i is= 1 j is= 2

i is= 1 j is= 3

i is= 2 j is= 1

i is= 2 j is= 2

i is= 2 j is= 3

i is= 3 j is= 1

i is= 3 j is= 2

i is= 3 j is= 3

PL/SQL

procedure

successfully

completed.

WHILE LOOP

1} create table area$

(radius number(2),

area number(10,2)

);

OUTPUT:- Table Created

declare

rad area$.radius%TYPE:=3;

area area$.area%TYPE;

Begin

while(rad<8)

loop

```
area:=3.142*rad*rad;

insert into area$ values(rad,area);

rad:=rad+1;

end loop;

end;

/
```

OUTPUT :- PL/SQL procedure successfully completed.

```
select * from area$;
```

OUTPUT:-

```
    RADIUS       AREA

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

         3      28.28

         4      50.27

         5      78.55

         6      113.11

         7      153.96
```

2} Declare

a number(2):=10; begin while(a<=20)

loop dbms_output.put_line('value of a

is:= '||a); a:=a+1; end loop; end;

OUTPUT :-

value of a is:= 10

value of a is:= 11

value of a is:= 12

value of a is:= 13

value of a is:= 14

value of a is:= 15

value of a is:= 16

value of a is:= 17

value of a is:= 18

value of a is:= 19

value of a is:= 20

PL/SQL procedure successfully completed.

PRACTICAL NO 4

AIM:WRITING PL/SQL BLOCKS WITH NULL INSIDE IF

```
create table mytable

( total_col

number(2), num_col

number(2), char_col

number(2), date_col

number(2)

);
```

OUTPUT:- Table Created

```
declare

tc number(2):='&tc';

nc number(2):='&nc';

cc number(2):='&cc';

dc number(2):='&dc';

begin

if(tc is null) then tc:=1; end if; if(nc is

null) then nc:=0; end if; if(cc is null)
```

then cc:=0; end if; if(dc is null) then

dc:=0; end if; insert into mytable

values(tc,nc,cc,dc); end;

/

OUTPUT:-

Enter value for tc: 2

Enter value for nc:

Enter value for cc:

Enter value for dc:

PL/SQL procedure successfully completed.

select * from mytable;

OUTPUT:-

TOTAL_COL	NUM_COL	CHAR_COL	DATE_COL
2	0	0	0

PRACTICAL NO 5

AIM- Create Empty Procedure and call the procedure

Create or replace procedure display11

as begin dbms_output.put_line(' ');

end;

A-call the procedure

execute display11;

B1-Create stored procedure Create or

replace procedure display01 as

begin dbms_output.put_line('hello

world'); end;

B2-call the procedure

execute display01;

****
****

C-Write a pl/sql block to define procedure to insert data in a table

```
create table employee0s1
(
empid varchar2(6),
ename varchar2(15),
doj date, salary
number(10,2) );
```

Q>inserting a value

```
insert into employee0s1 values('A10','abc','12-jun-17',16000);
```

Q>Create procedure

```
declare eid varchar2(6);
ename varchar2(15); dt
```

date:='15-jul-17'; sal

number(10,2):=18000;

procedure insert0s1(eid varchar2,ename varchar2,dt date,sal

number) as begin insert into employee0s1

values(eid,ename,dt,sal);

end; begin

insert0s1(eid,ename,dt,sal);

end;

SQL> select * from employee0s1;

****

D1-write a procedure to calculate square of a given number

declare a number(5); b number(5);

procedure square(a in number,b out number)

```
is
begin b:=a*a; end; begin a:=5;

square(a,b);

dbms_output.put_line('square of a:'||b);

end;

/
```

D2-this program finds the avg of two values,here procedure takes two numbers using IN mode return their minimum using OUT parameters

```
declare
a number(5); b number(5); c number (5); procedure

average(a in number,b in number,c out number)

is

begin c:=(a+b)/2; end; begin a:=6; b:=8;

average(a,b,c);
```

```
dbms_output.put_line('average of a,b:'||c);

end;

/
```

D3-Find out minimum of two numbers

```
declare a number(5); b number(5); c number(5); procedure

minimum(a in number,b in number,c out number)

is

begin if(a<b) then c:=a; else c:=b; end if;

end; begin a:=6; b:=8; minimum(a,b,c);

dbms_output.put_line('minimum of a,b:'||c);

end;

/
```

PRACTICAL NO 6

AIM :WRITING FUNCTIONS IN PL/SQL BLOCK

A}create or replace function find_area_rect

(height in number, Width in number)

return number

as

varea number;

begin

varea := height * Width;

return varea;

end;

****

select find_area_rect(10,20)area from dual;

B}CREATE OR REPLACE FUNCTION ss_thresh

RETURN NUMBER AS

x NUMBER(9,2);

BEGIN

x := 65400;

RETURN x;

END;

/

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


SS_THRESH()
-----------
     65400
```

C}Call function in dbms_output.put_line

```
SQL>

SQL>  create or replace function ite(
     p_expression boolean,
     p_true varchar2,
     p_false varchar2 ) return varchar2 as
   begin
     if p_expression then
        return p_true;
     end if;
     return p_false;
  end ite;
  /
```

Function created.

SQL> execute dbms_output.put_line(ite(1=2, 'Equal', 'Not Equal'));

Not Equal

PL/SQL procedure successfully completed.

SQL> execute dbms_output.put_line(ite(2>3, 'True', 'False'));

False

PL/SQL procedure successfully completed.

D}DECLARE

  num number;

  factorial number;

FUNCTION fact(x number)

RETURN number

IS

  f number;

BEGIN

```
   IF x=0 THEN
      f := 1;
   ELSE
      f := x * fact(x-1);
   END IF;
RETURN f;
END;
BEGIN
   num:= 6;
   factorial := fact(num);
   dbms_output.put_line(' Factorial '|| num || ' is ' || factorial);
END;
```

E}Note: create table emp with some columns and insert few records

```
create or replace function totalemp

return number as
vcount number;
```

```
begin

select count(*) into vcount from emp;

return vcount;

end totalemp;
```

****

```
select totalemp from emp;
```

F}

```
create or replace function first_function return varchar2 as
  begin
    return 'Hello World';
  end first_function;
  /
```

Function created.

```
declare
```

```
l_str varchar2(100) := null;

begin

l_str := first_function;

dbms_output.put_line( l_str );

end;

/
```

Hello World

PRACTICAL NO 7

AIM: WRITING RECURSIVE FUNCTIONS IN PL/SQL BLOCKS

```
create function fibonacci1(b

 number) return number as ft

 number:=0; st number:=1; nt

 number; begin

 dbms_output.put_line(ft);
```

```
dbms_output.put_line(st); for i in

1..(b-2)

loop nt:=st+ft; ft:=st;

st:=nt;

dbms_output.put_line(nt)

; end loop; return null;

end fibonacci1;

/
```

Output:

```
execute dbms_output.put_line(fibonacci1(10));
```