Practical 1
Experiment.

Aim: Writing plsql blocks with basic programing constructs by including followi

A). Sequential statement.
B) Unconstrained loop.

```
declare
v_num number := 1;
begin.
loop.
dbms_output.put_line ('Current value
                        of v_num: ' || v_num)
v_num := v_num + 1;
exit when v_num > 10;
end loop;
end;
/
```

O/P:
```
Current value of v_num: 1
Current value of: v_num: 2
Current value of v_num: 3
Current value of v_num: 4
Current value of v_num: 5
Current value of v_num: 6
Current value of v_num: 7
Current value of v_num: 8
Current value of v_num: 9
Current value of v_num: 10
```

ii) Write a program to find the area circle.

```
declare
r int := &r ;
 a float ;
 begin
 a := 3.14 * r * r ;
 dbms_output.put_line('area is'||a)
 end;
/
```

O/P: Enter the r : 10
 area is : 314.

iii) name & rollno

```
declare
v_name varchar(20);
 v_roll number;
 Begin
 v_name := '&name';
 v_roll := '& roll';
 dbms_output.put_line('Name : '||v_
dbms_output.put_line('rollno:'||v_
end;
/
```

O/P:

```
 Enter value for name : Vipul
 Enter value for rollno : 1
 Name : Vipul
 rollno : 1.
```

Experiment 2.

Aim: A) Creating simple sequence with clauses like start with, increment by, max value, min value cycle | No cycle, cache no cache, order | Noorecer.

```
-> create sequence seq2
   start with 1
   increment by 1
   min value 0
   max value 100.
   no cycle;
```

o/p: Sequence created.

B]. Using sequence for tables : |

```
-> create table emp01 (id number (30),
   name varchar (20), salary number (30),
   job varchar (50), dept id number);
```

o/p: Table created

```
-> insert into emp01 values (seq2.nextval,
      'shubham', 23000, 'manager', 102);

   insert into emp01 values (seq2.nextval,
      'Komal', 20000, 'clerk', 101);
```

```
insert into emp01 values( seq2.nextval,
        'Vipul', 50000, 'Programmer',
                                103);
insert into emp01 values( seq2.nextval,
        'Sumit', 35000, 'HR', 104);
insert into emp01 values( seq2.nextval,
        'Gaurav', 25000, 'clerk', 105);

select * from emp01;
```

O/P:

Job	id	NAME	Salary	dept_id
Manager	1	Shubham	23000	102
clerk	2	Komal	20000	101
Programer	3	Vipul	50000	103
HR	4	Sumit	35000	104
clerk	5	Gaurav	25000	105

Experiment 3.

Aim: Plsql program with basic programing constructs including following

A) IF -- THEN - ELSE

```
declare.
a int := &a;
b int := &b;
begin.
if (a>b) then.
dbms_output.put_line (a is greater');
else.
dbms_output.put_line ('b is greater');
end if
end.
/.
```

O/P:

```
Enter value for a: 45
Enter value for b: 35
a is greater
```

B). IF -- THEN -- ELSE -- ENDIF.

```
-> declare.
str1 varchar (20) := 'str1';
str2 varchar (20) := 'str2';
begin
if str1 = str2 then.
dbms_output.put_line ('str1 and str2 same
else.
dbms_output.put_line('str1 and str2 not sa
end if;
end;
/
```

O/P: Enter value for str1 = Hello
Enter value for str2 = Hello
Str1 and Str2 same

c) Salary Calculation
→ Create employee table
create table employee_desc varchar(20), emp_sal int);
O/P: table created.

→ insert 3 values in the table
insert into employee values(1,'Vipul','HR',2000)
O/P: 1 row created.
insert into employee values(2,'Shubam','HR',1000);
O/P: 1 row created.
insert into employee values(3,'anurag','clerk',1000);
O/P: 1 row created.

SQL> ~~insert into employee values~~ Select * from employee;

emp_id	emp_name	emp_disc	emp_sal
1	Vipul	HR	2000
2	Subham	HR	1000
3	Anurag	clerk	1000

D). IF -- ELSE END IF.

```
SQL> Declare
     Job varchar(20) := &job;
     begin
      If job = 'clerk' then
     update employee salary emp_sal = emp_sal(9100*
     where emp_name = "Shubam";                  emp_sal)
     else if  job = 'Manager' then
      update employee set emp_sal = emp_sal + (9100*
      where  emp_name = 'Anurag';                 emp_sal)
      else if  job = 'hr' then
      update employee set emp_sal = emp_sal + (7100*
      where emp_name = 'Vipul';                   emp_sal)
      else
      dbms_output.put_line('Invalid');
      end if;
       end;
O/P:  enter value for job : 'clerk'
      PLSQL procedure successfully completed
```

SQL> Select * from employee;

emp_id	emp_name	emp_desc	emp_sal
1	~~Shubham~~ Vipul	HR	7100
2	~~Anurag~~ Shubam	HR	9100
3	~~Vipul~~ Anurag	clerk	1100

Q1) Greet Good Morning.

```
SQL> declare
     hour varchar(10) := to_char(sysdate, 'hh24');
     begin
```

```
if hour >= 16 then
dbms_output.put_line ('Good Evening');
else if hour >= 12 then
dbms_output.put_line ('Good Afternoon');
else
dbms_output.put_line ('Good Morning');
end if;
end;
```

O/P: Good Afternoon

2) Case statement.

SQL>
```
declare
grade char := '&grade';
begin
case grade
when 'O' then dbms_output.put_line ('Outstanding');
when 'A' then dbms_output.put_line ('excellent');
when 'B' then dbms_output.put_line ('good');
when 'C' then dbms_output.put_line ('Satisfy');
when 'D' then dbms_output.put_line ('Bad');
else
dbms_output.put_line ('Invalid');
end case;
end;
/
```

O/P:
```
Enter value for grade: A.
    excellent.
```

Experiment 4.

<u>Aim</u>: Write plsql blocks with basic programing constructs for following iteratiue.

4a). While loop

i).
```
declare
 i int :=1 ;
 total int := 0;
 begin
  while i<10 loop
  i= i+2;
  total : total+i;
   end loop;
   dbms_output.put_line ('total: '||total).
   end;
   /
```

O/P: total : 25

ii). Table of 11.
```
Declare
     i number := 1;
 begin.
     while i<=10 loop.
        dbms_output.put_line('11 *'||i||' = '
                                      ||11*i);
        i:= i+1;
  end loop;
  end;
  /
```

o/p:
```
11 * 1 = 11
11 * 2 = 22
11 * 3 = 33
11 * 4 = 44
11 * 5 = 55
11 * 6 = 66
11 * 7 = 77
11 * 8 = 88
11 * 9 = 99
11 * 10 = 110
```

4b) FOR LOOP.

i) Unreversed.
```
declare
n int := 1;
begin
for n in 1..10 loop
dbms_output.put_line('num: '||n);
end loop;
end;
/
```
o/p:
```
num : 1
num : 2
num : 3
num : 4
num : 5
num : 6
num : 7
num : 8
num : 9
num : 10
```

4b(i). Reversed for loop.

```
declare
 n int := 10;
 begin
for n in reverse 1..10 loop
 dbms_output.put_line('num: '||n);
 end loop;
 end;
 /
```

O/P:

```
num: 10
num: 9
num: 8
num: 7
num: 6
num: 5
num: 4
num: 3
num: 2
num: 1
```

4b iii) goto in for loop

```
declare
 i number := 1;
 begin
for i in 1..10 loop
if i = 5 then
 goto loop_end;
 end if;
dbms_output.put_line('current value of i:'
                                      ||i);
 end loop;
```

```
<< loop_end >>.
dbms_output.put_line("loop ended");
end;
/.
```

O/P:
```
current value of i : 1
current value of i : 2
current value of i : 3
current value of i : 4
loop ended.
```

Experiments No 5

Aim: Writing plsql blocks with basic programming constructs by including a Goto Jump out of loop and NULL as statement inside if.

A)

```
• DECLARE.
  I NUMBER ;
  BEGIN
    For I in 1...10 loop
      if I = 5 then
          Goto Loop_end;
      End if ;
      dbms_output. put_line ('Current value
                                of I:' ||I);
    End loop ;
    << Loop_End >>
    dbms _output .put _line ('Loop Ended');
  End ;
```

Output :

```
Current value of I : 1
Current value of I : 2
Current value of I : 3
Current value of I : 4.
Loop Ended
```

8). Create a plsql block that has four sections. Each section should output a statement. Use labels and Goto command to execute in following order.
Section 3
Section 2
Section 1
Section 4

```
begin
 goto S3 ;
 <<S1>>
 dbms_output.put_line('Section 1');
 goto S4 ;
 <<S2>>
 dbms_output.put_line('Section 2');
 goto S1 ;
 <<S3>>
 dbms_output.put_line('Section 3');
 goto S2 ;
 <<S4>>
 dbms_output.put_line('Section 4');
 end;
 /
```

O/P:

```
Section 3
Section 2
Section 1
Section 4
```

Experiments 6.

Aim: * Writing procedures in Plsql block.
i). Create an empty procedure, replace and call procedure.
ii). Create a stored procedure and call it.
iii). Define procedure to insert data.
iv). A forwards declaration of procedure.

```
Create table Employee (Emp-no int primary
              Key, Emp-name varchar (30),
              Sal NUMBER (6), Age INT);
Insert into Employee values (1, 'Ram',
                         25000, 30);
Insert into Employee values (2, 'Shyam'
                         27000, 31);
Insert into Employee values (3, 'Arjun'. ~~26000~~
                         '26000', 29);
```

- Create an empty procedure.

```
Create or replace procedure proc_emp
As
Begin
  Null :- Do Nothing.
End ;
/
```

- Calling an empty procedure.

```
Exec proc_emp.
```

— Replace a procedure, create a stored procedure

```
Create or replace procedure proc_emp
   (P_no IN INT, P_name IN VARCHAR(21),
      P_sal IN NUMBER, P_Age IN INT)
As
Begin
 Insert into Employee (Empno, Emp_name
 sal, age) values (P_no, P_name,
      P_Sal, P_Age);
 Commit;
 dbms_output.put_line('Record inserted
                        successfully');
 End;
 /
```

— Calling a procedure

```
Exec proc_emp(4, 'KRISNA', 30000, 31)
```

— Retreiving data.

```
Select * from employee;
```

Experiment 7

7A) Define and call function
7B) Use function in select clause

-> Create or Replace function calculate_employees_bonus (employee_id in number)

```
return number
as
begin
  declare bonus number;
 begin
   select sal * 0.1 into bonus from employee
where emp_no = employee_id;
   return bonus;
 end;
 end;
 /
```

C] Call function in dbms_output.put_line

```
declare
  employee_id number;
  bonus number;
begin
  bonus := calculate_employees_bonus
                    (employee_id);
  dbms_output.put_line('Bonus of employees'
          || Employee_id || 'is' || bonus);
end;
/
```

PlSQL procedure successfully complete

D] Recursive function.

```
create function calculate_factorial (num1
                            in number)
Return number
As
begin.
   If num1 = 0 then
        Return 1;
   else
        return num 1* calculate_factorial
                          (num1-1);
  end if;
  end;
  /
```

Calling function.

```
declare
   Factorial number;
begin.
   Factorial := calculate_factorial (5);
   dbms_output.put_line('Factorial of 5:'
                         || Factorial);
end;
/
```

O/P: Factorial of 5: 120
     Plsql procedure successfully completed

num : 10

7E] Count employee from function and return value back. First create table

```
create table emp( empno number(4) constrain
  e_pk primary key, ename varchar2(8);
 init varchar2 (5), job varchar2(8), Mgr
 number (4); bdate date, sal number (62),
 comm number (6,2), deptno number(2)
 default 10);

insert into emp values (1, `Tushar', 'N', `Coder',
        13, Date '1965-12-17', 800, Null, 20);
insert into emp values(2, 'Vishal', `J', `Tester', 6,
        Date, '1991-02-20', 1600, 300);
insert into emp values (3, `Samir', `T', `Tester'
       6, Date '1996-02-22', 1250, 500);

-> Create or replace function count_employees
              (CNT in number)
 return number
  as
    employee_count number;
 begin
   select count (*) into employee_count
                          from emp;
   return employee_count;
 end;
 /
```

O/P: Number of employee : 5
 PlSQL procedure complete successfuly.

7F] Call function and store return value to a variable.

```
declare
 employee_count number;
 cnt number;
begin.
   employee_count := count_employees(CNT);
   dbms_output.put_line ('Number of
              employees:' || employee_count);
end;
/.        O/P: Number of employee: 5
               PL/SQL procedure complete successfully.
```

> Select ~~[illegible]~~ * from employee;

empno	ename	init	job	date	sal	depno
1	Tushar	N	Coder	1961-1-7	800	Null
2	Vishal	J	Tester	1961-2-20	300	30
3	Sumit	T	Tester	1961-2-21	400	30

Experiment 8

Aim: Creating and working with insert / update / delete trigger using before / after clause.

```
create table emp( empno number(4) constrain
e-pk primary key, ename varchar2(8),
init varchar2(5), job varchar2(8), Mgr
number(4), bdate date, sal number(6,2)
comm number(6,2), deptno number(2));

insert into emp values(1,'Tushar','N',
      'Coder', 13, date '1965-12-17', 800, Null);
insert into emp values( 2,'Vishal','J','Tester',
  6, date '1961-02-20', 1600, 300, 30);
insert into emp values( 3, 'Samir','T','Test'
  6, date '1962-02-22', 1250, 500, 30);
```

ii). Creating and working with insert triger using before clause.

```
create or replace trigger emp_insert_trigger
before insert on emp
for each row
begin.
  dbms_output.put_line ('Inserting new
  employee record :' || New. empno);
end;
/

insert into emp value( 4, 'Sujit', 'P', 'Developer'
              6, date '1972-03-22', 1600, 500);
```

O/P:

```
    1 Row(s) inserted.
    inserting new employee record : 4.
```

8ii) Creating and working with update trigger using before clause.

```
Create or replace trigger emp_update_
  trigger
before update on emp
for each row
begin
  dbms_output.put_line ('Updating employee
  record:' || old.empno );
end;
/
update emp set sal = 1500 where empno=1;
```

O/P:

```
    1 Row(s) updated,
    updating employee record : 1
```

8 iii). Create and working with delete trigger using before clause

```
create trigger emp_delete_trigger
before delete on emp
for each row
begin
  dbms_output.put_line('Deleting employee
                    record:'||:old.empno);
end;
delete from emp where empno=5;
```

O/P:

```
   0 Row(s) Deleted.
```

8 iv) Creating and working with insert trigger using after clause.

```
create trigger emp_insert_trigger1
after insert on emp
for each row
 begin
  dbms_output.put_line('New employee record
 inserted:'||:New.empno);
 end;
 /
insert into emp values(6,'KUSH','C','JR.
         MAN',7,Date '1968-1-7',2000,121)
```

O/P:

```
      1 Row(S) inserted
      inserting new employee record: 6.
```

8v). Creating and working with update trigger using after clause

```
create trigger emp_update trigger1
after update on emp
for each row
begin
   dbms_output.put_line('Employee record
                updated: "|| : Old. Empno);
end;
/
Update emp set sal = 1400 where empno=3;
```

```
O/P:    1 Row(s) updated
        updating employee record: 3.
```

8vi). Creating and working with delete trigger using after clause.

```
create trigger emp_delete_trigger1
after delete on emp
for each row
begin
   dbms_output.put_line('Employee record
                deleted: "|| Old. empno);
end;
Delete from emp where empno = 5;
```

```
O/P:     1 Row(s) Deleted
         Deleting employee record: 5
```

Experiment 9.

Aim: Write an implicit and explicit cursor to complete the task.

1)
```
declare
name varchar2(20);
cursor cur_emp
is
select ename from emp where empno=5;
begin.
open cur_emp;
loop
fetch cur_emp into name;
dbms_output.put_line(name);
exit when cur_emp % Not found;
end loop;
dbms_output.put_line ('Record not found');
close cur_emp;
end;
/.
```

O/P:
```
Statement processed.
Record not found.
Plsql procedure successfully completed
```

11) Implicit cursor.

```
declare
v_no number (8);
begin.
update emp set sal = sal + 1000
where empno = 1;
v_no := sql % Rowcount ;
if sql % found then
dbms_output.put_line('Salary of'
            || v_no||'employee updated');
else
dbms_output.put_line('employee not found)
end if;
end;
/
```

```
Y_no ||' Employee Updated');
```

~~O/P:~~

~~Salary of 1 employee updated~~.

```
else
dbms_output.put_line ('Employee No');
End if;
/
```

O/P:

```
Updating employee record : 104.
Salary of 1 employee updated.
Pl/sql procedure completed successfully.
```

Experiment 10

Aim: Create packages and use it in sql block to complete the task.

~~Create packages and use it~~

```
Create or replace package employee_
management as
procedure update_salaries(percentage
in number);
end;

Create or replace package body
employee_management as
procedure update_salaries(percentag
    in number)
as
begin
update emp
set sal = sal * (1+ percentage/100);
end;
end;
/

declare
 salary_increase_percentage number:=5;
begin
employee_management.update_salaries
    (salary_increase_percentage);
end;
```

O/P:
```
Updating employee record : 101
Updating employee record : 102
Updating employee record : 103
Updating employee record : 104
Updating employee record : 105
```

Experiment 11.

Aim: Write sql block to handle exception by writing

A). Predefined exceptions

```
declare
    e_id emp.empno%type := 8;
    e_name emp.ename%type;
    e_job emp.job%type;
begin
    select ename.job into e_name, e_job
    from emp
    where empno = e_id;
    dbms_output.put_line("Name:"||e_
                                 name);
    dbms_output.put_line("job:" ||e_job);

Exception
  when no_data_found then
     dbms_output.put_line("No such
                            employee");
  when others then
     dbms_output.put_line("Error");
end;
/
```

O/P: No such employee;

B). User - defined exceptions

```
declare
    e_id emp.empno%.type := 0;
    e_name emp.ename%.type;
    e_job emp.job%.type;

    ex_invalid exception;
begin.
  if e_id <= 0 then.
      raise ex_invalid_id;
  else
      select e_name, job into e_name,
                    e_job
  from emp.
  where empno = e_id;
  dbms_output.put_line("Name:" ||
                        e_name);
  dbms_output.put_line('job:', || e_job);
  end if;

Exception.
  when ex-invalid_id then
  dbms_output.put_line("Id must
                be greater then zero');
when no_data_found then.
   dbms_output.put_line('No such
                 employee");
when others then
   dbms_output.put_line ('Error');
end;
/
```

O/P: No Such employee;

11c) Reclared pre-defined exceptions

```
declare
    e_id emp.empno %type := 8;
    e_name emp.ename % type;
    e_job emp.job % type;

  No_data_found exception;
  begin
  select ename, job into e_name, e_job
  from emp
  where empno = e_id;
  dbms_output.put_line ('Name: '|| e_name);
  dbms_output.put_line ('Job: '|| e_job);

  exception
      when standard.no_data_found then
      dbms_output.put_line ('No Such
                                  employee');
   when others then
      dbms_output.put_line ('Error');
   End;
   /
```

O/P: No Such employee;

Experiment 12.

Aim: Create nested tables and work with nested tables.

Create or replace type add_type as table of varchar(50);

create table edb (empno number(4) primary key, e_name varchar2(8), dept_no numb (2), Default 10, addresses add_type)
Nested table addresses store as nesta tab~~le~~_add;

insert into edb (empno, e_name, dept_no, address) values (1, 'Ram', 10, Add_type ('103. Navghar gaon', 'Bhayander'));

insert into edb (emp_no, e_name, dept_no, address) values (4, 'Jatin', 20, Add_type ('123 Main st',));

O/P:

Emp-no	Ename	deptno	Address
1	Ram	10	Bhayander ~~w~~ ;
4	Jatin	20	Main St.