PRACTICAL: 8
PL / SQL PRACTICAL: 3

AIM: FUNCTION, PROCEDURE, TRIGGER BASED PL/SQL PROGRAMS.

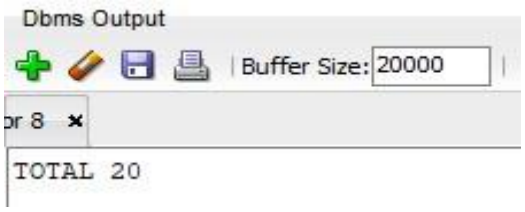
1.Create a Function which will work as addition. e.g select ADDITION (10,10) from dual;

Query:

```
create or replace function addition(a number, b number)
return number
is
ans number(5);
begin ans:=a+b; return
ans;
end;

declare  c
number(2); begin
c:=addition(10,10);
DBMS_OUTPUT.PUT_LINE('TOTAL= '|| c); end;
```

Output:



2. Create a Function that is Work as Calculator. Use operator like '+', '-', '*', '/'.

Query: create or replace function calculator(a number, b number,c varchar2) return
number
is
ans varchar(5);
begin if(c =
'+') then
ans:=a+b;
return ans;

else if(c = '-') then
ans:=a-b;
return ans;

else if(c = '*') then
ans:=a*b;
return ans;

else if(c = '/') then
ans:=a/b;
return ans;

else
DBMS_OUTPUT.PUT_LINE('INVALID OPERATOR. ');
end if; end if; end if; end if; end;

declare c
number(5); O
varchar2(5); begin
c:=calculator(&a,&b,'&O');
DBMS_OUTPUT.PUT_LINE('ANS = '|| c);
end;

Output:

```
c number(5);
a NUMBER(5);
b NUMBER(5);
o varchar2(5);
BEGIN
c:=CALCULATOR(5,3,'*');
DBMS_OUTPUT.PUT_LINE('Ans = '|| c);
END;

PL/SQL procedure successfully completed.
```

```
Ans = 15
```

Here, value of a=5, b=3and operator=*

3. Create a Procedure that update the salary of employee Using following formula: Sal = Sal + (Sal* Percentage/100). Pass two parameter values: Name & Percentage.

Table : Salary

Emp_No	Name	Salary
01	Riya	8000
02	Rima	9000
03	Ajay	10000
04	Raj	11000

```
create table salary(emp_no number(10),name varchar2(10),salary number(10));
insert into salary values('01','Riya',8000);
insert into salary values('02','Rima',9000);
insert into salary values('03','Ajay',10000);
insert into salary values('04','Raj',11000);
select *from salary;
```

SQL | All Rows Fetched:

	EMP_NO	NAME	SALARY
1	1	Riya	8000
2	2	Rima	9000
3	3	Ajay	10000
4	4	Raj	11000

Query:

```
create or replace procedure update_sal(name1 varchar2, per number) is
new_sal number(10);  temp number(10);
begin  select salary into temp from salary where
name=name1; new_sal:=temp+(temp*per/100);
update salary set salary=new_sal where name=name1;
DBMS_OUTPUT.PUT_LINE('SUCCESFULLY UPDATED. ');
end;

execute update_sal('Riya',8);
```

Output:

```
SUCCESFULLY UPDATED!
```

4. When you perform update or delete operation on table company1, it stores the updated or deleted values in table company2. Also show the operation 'update' and 'delete' and username.

Table: Company1 (no, name) Company2 (no, name, operation)

Query:

```
create table company1(no number(5),name varchar2(10));
create table company2(no number(5),name varchar2(10),operation
varchar2(10));  insert into company1 values (1,'DBMS');  update
company1 set name='xyz';  select * from company1;
 select * from company2;
```

```
create  or replace trigger t2
after  update or delete on company1 for each row begin if
updating then  insert into company2
values(:new.no,:new.name,'update');
end if;
 if deleting then  insert into company2
values(:old.no,:old.name,'delete'); end if;
DBMS_OUTPUT.PUT_LINE('Operation Completed. '); end;
```

Output:

```
Table COMPANY1 created.


Table COMPANY2 created.
```

```
insert into company1 values (1,'DBMS'); delete from
company1 where no=1; select * from company2;
```

	NO	NAME	OPERATION
1	2	DBMS	delete

5. Whenever you insert any value into table test1 then table test2 automatically calculate the Total of No 1 and No 2.

Table: Test1 (No1,No2) Test2 (Total).

Query:

```
create table test1(No1 number, No2
number);
create table test2(total number);
insert into test1 values(10,10); select
*from test1;
 select *from test2;

create  or replace trigger t1
after
insert on test1  for each
row  declare  a number;
```

```
begin  a:= :new.no1
+:new.no2;  insert into
test2 values(a);
DBMS_OUTPUT.PUT_LINE('Data Inserted.');
end;
```

Output: