Expt - 25

Date -

Aim: Write a pl/sql for a function to calculate the total development cost for a given programmer using cursors

PL/SQL

```
create or replace function gettot(p in software.pname%type) return number a
    s number;
    cursor s is select * from software where pname = p;
    t s%rowtype;
begin
    s:= 0;
    open s;
    loop
        fetch s into t;
        exit when s%notfound;
        s:= s+ t.dcost;
    end loop;
    close s;
    return(s);
end;

declare
    res number;
    p software.pname%type := 'Tulasi';
begin
    res := gettot(p);
    dbms_output.put_line('Total development cost is' ||res);
end;
```

Expt no: 26

Date :

Aim : Write a pl/sql a package with one procedure and one function. procedure displays mathematical product table for a given number. function return product of two number.

pl/sql:

```
create or replace package my_pack1 is
    procedure product_table(a in number);
    function product(a in number, b in number)return number;
end;

create or replace package body my_pack1 as
    procedure product_table(a in number)as
        i number;
        begin
            i:=1;
            while(i<=10)
            loop
                dbms_output.put_line(a||'*'||i||'='||a*i);
                i:=i+1;
            end loop;
        end product_table;
        function product(a in number, b in number) return
        number as c number;
        begin
            c:=a*b;
            return c;
end; end product;
```

```
declare
    x number := 12;
    y number := 2;
    z number;
begin
    my_packl.product_table(x);
    z := my_packl.product(x,y);
    dbms_output.put_line('product of'||x||'and'||y||'is:'||
                         z);

end;
```

Expt - 27

Date -

Aim: Write a PL/SQL for a package with one procedure and one function. procedure displays mathematical product table for a given number. Function returns product of two numbers) salary of given programmer. Function returns the name of programmer for a given project.

PL/SQL:

```
create or replace package pack1 is
    procedure P1(p in programmer.PNAME%type);
    function f1(t in software.TITLE%type) return
        software.PNAME%type;

end;
create or replace package body pack1 procedur
    procedure P1(p in programmer.PNAME%type)as
        s programmer.SALARY%type;

    begin
        select SALARY into s from programmer where
            PNAME=p;
        dbms_output.put_line('salary is '||s);
    end P1;
    function f1(t in software.TITLE%type) return
        software.PNAME%type as
        x software.PNAME%type;
        begin
            select PNAME into x from software where
            TITLE=t;
            return x;
        end f1;
end;
```

```
declare
    a programmer.PNAME%type := 'Anand';
    b software.TITLE%type := 'Read me';
    s software.PNAME%type;
begin
    pack1.p1(a);
    s := pack1.f1(b);
    dbms_output.put_line('PNAME of given project :'||s);
end;
```

Expt - 28

Date -

Aim: write a pl/sql for a package with one procedure and one function. procedure displays the titles of projects done in given language. Function returns the name of institute in which the programmer studied.

pl/sql:

```
create or replace package pack2 is
    procedure p2(x in software.DEV-D%type);
    function f2(y in studies.PNAME%type)return
        studies.SPLACE%type;
end;
create or replace package body pack2 as
    procedure p2(x in software.DEV-D%type) as
    cursor s is select * from software where DEV-D=x;
    t s%rowtype;
    begin
        open s;
        loop
            fetch s in t;
            exit when s%notfound;
            dbms_output.put_line(t.TITLE);
        end loop;
        close s;
    end p2;
    function f2(y in studies.PNAME%type)return
        studies.SPLACE%type as s studies.SPLACE%type;
```

```
    begin
        select SPLACE into s from Studies where PNAME = P;
        return s;
    end f2;

end;

declare
    a software.DEV-D%type := 'Basic';
    b studies.PNAME%type := 'Anand';
    s studies.SPLACE%type;

begin
    packs.p2(a);
    s := packs.f2(b);
    dbms_output.put_line('SPLACE is :' ||s);

end;
```

Expt - 29

Date -

Aim: write a pl/sql for a programmer to demonstrate user defined exceptions.

pl/sql:

```
declare
    a programmer.pname%type := 'Anand';
    datbirth programmer.doj%type;
    invalid_age_exception exception;
    begin
        select doj into datbirth from programmer where
        pname = a;
        if (sysdate - datbirth) /365 > 18 then
            dbms_output.put_line (a || 'is eligible to vote');
        else
            RAISE invalid_age_exception;
        end if;
        Exception
            when invalid_age_exception then
            dbms_output.put_line('Sorry' ||a|| 'is not
            eligible to vote');
        en
    e   end;
```

Expt - 30

Date -

Aim: write a PL/SQL for a trigger to calculate the total of a student before insert operation.

PL/SQL:

```
create table student1 (rollno number(3) primary key,
                       sname varchar2(15), marks1 number(3),
                       marks2 number(3), total number(4));

create or replace trigger auto_cal before insert on student1 for each row
declare
begin
    :new.total:=:new.marks1+:new.marks2;
end;
insert into student1 (1,'Raamesh',44,67,null);
Select * from student1;
```

Exp: - 31

Date -

Aim: write a pl/SQL for a trigger to store the details of updated salary of a programmer into another table.

pl/SQL:

```
Create table prog as select pname, salary from programmer;
Create table update_prog (pname varchar2(20), old_salary number(7,2),
                          new_salary number(7,2), dt date,
                          time varchar2(10));

Create or replace trigger update_status after update on prog for each row
begin
  insert into update_prog values (:old.pname, :old.salary, :new.
  salary, sysdate, substr(current_timestamp, 11, 8));
end;
update prog set salary = salary + 200 where pname = 'anand';
select * from update_prog;
```

Expt - 32

Date -

Aim: Write a pl/sql for a trigger to know latest and oldest tuples in a table.

pl/sql:

```
create table student (rollnumber(3) primary key, sname
varchar2(15), age number(3));

create or replace trigger age before insert on student
declare
begin
    update student set age = age + 1;
end;
insert into student values(1,'siva',0);
select * from student;
```