1. Write a PL/SQL block to read a number and check whether it is a palindrome or not.

```
declare
        n number:=&n;
        m number;
        rev number:=0;
begin
        m:=n;
        while m>0 loop
                rev:=rev*10+mod(m,10);
                m:=floor(m/10);
        end loop;
        if rev=n then
                dbms_output.put_line(n||' is a palindrome');
        else
                dbms_output.put_line(n||' is not a palindrome');
        end if;
end;
SQL> @C:\Users\LENOVO\Desktop\palindrome.sql;
 19  /
Enter value for n: 54845
old   2:       n number:=&n;
new   2:       n number:=54845;
54845is a palindrome
PL/SQL procedure successfully completed.
```

2. Write a PL/SQL Program to print the first n fibonacci numbers

```
DECLARE
n number;
a NUMBER:=0;
b NUMBER:=1;
c NUMBER;
```

```
BEGIN
n:=&n;
DBMS_OUTPUT.PUT_LINE(a);
DBMS_OUTPUT.PUT_LINE(b);
for i in 1..n-2 LOOP
 c:=a+b;
 DBMS_OUTPUT.PUT_LINE(c);
 a:=b;
 b:=c;
END LOOP;
END;
SQL> @C:\Users\LENOVO\Desktop\Fibonacci.sql;
 17  /
Enter value for n: 5
old   7: n:=&n;
new   7: n:=5;
0
1
1
2
3
PL/SQL procedure successfully completed.
```

3. Write a PL/SQL block which will accept a number and checks if it is a prime number or not.

```
declare
        n number:=&n;
        c number:=0;
        i number:=0;
begin
        for i in 2..sqrt(n/2) loop
                if mod(n,i)=0 then
```

```
                    c:=1;
                    exit when c=1;
                end if;
        end loop;
        if c=1 then
                dbms_output.put_line(n||' is not a prime number');
        else
                dbms_output.put_line(n||' is a prime number');
        end if;
end;
SQL>  @C:\Users\LENOVO\Desktop\prime.sql;
 18  /
Enter value for n: 27
old   2:       n number:=&n;
new   2:       n number:=27;
27is not a prime number

PL/SQL procedure successfully completed.

SQL> @C:\Users\LENOVO\Desktop\prime.sql;
 18  /
Enter value for n: 3
old   2:       n number:=&n;
new   2:       n number:=3;
3is a prime number

PL/SQL procedure successfully completed.
```

4. Write a PL/SQL block which will accept two numbers and find out their LCM and HCF. The output should be stored in a table called **DEMO_TAB**.

DEMO_TAB

Number1	Number2	LCM	HCF

```
declare
  a number:=&number1;
  b number:=&number2;
  m number;
  n number;
  hcf number;
  lcm number;
begin
  if a>b then
        m:=a;
  else
        m:=b;
  end if;
  loop
        if mod(m,a)=0 and mod(m,b)=0 then
              exit;
        else
              m:=m+1;
        end if;
  end loop;
  lcm:=m;
  hcf:=(a*b)/lcm;
  insert into demo_tab values(a,b,lcm,hcf);
end;
```

```
Enter value for first: 45
old   2:       a number:=&first;
new   2:       a number:=45;
```

```
Enter value for second: 43
old   3:       b number:=&second;
new   3:       b number:=43;

PL/SQL procedure successfully completed.

SQL> select * from demo_tab;

    FIRST     SECOND       LCM       HCF
---------- ---------- ---------- ----------
       45         43       1935         1
```

5. Write a Pl/SQL program using FOR loop to insert ten rows into a database table/

```
SQL> create table demo(n number(2),n_square number(4),n_cube number(6));

declare
 i number;
begin
 for i in 1..10 loop
          insert into demo values(i,i*i,i*i*i);
 end loop;
end;

SQL> select * from demo;

         N   N_SQUARE     N_CUBE
---------- ---------- ----------
         1          1          1
         2          4          8
         3          9         27
```

4	16	64
5	25	125
6	36	216
7	49	343
8	64	512
9	81	729
10	100	1000

10 rows selected.

6. Consider the following relation schemas

Inventory

Product_ID	Product_name	Quantity

Purchase_Record

Product_ID	Status	Date

Write a PL/SQL block to read the quantity of a product from inventory and if it is > 0 reduce the quantity by 1 and record the status of purchase of that product as 'PURCHASED'. Otherwise record the status of purchase of that product as 'OUT OF STOCK'. While recording the status of a purchase, record the date of transaction.

```
SQL> create table inventory(product_id number(5),product_name varchar(20),quantity number(4));
```

Table created.

```
create table purchase_record(product_id number(5),status varchar(30),dates varchar(10))
```

Table created.

```
SQL> select * from inventory;
```

```
PRODUCT_ID PRODUCT_NAME          QUANTITY
---------- -------------------- ----------
      1001 soap                          3
      1002 brush                         3
      1003 plate                         0
      1004 dress                         6

declare
 q inventory.quantity%type:=&quantity;
 p_id inventory.product_id%type;
 q1 inventory.quantity%type;
begin
 select product_id into p_id from inventory where quantity=q;
 if q>0 then
            insert into purchase_record values(p_id,'PURCHASED',sysdate);
            update inventory set quantity=quantity-1 where product_id=p_id;
 else
            insert into purchase_record values(p_id,'OUT OF STOCK',sysdate);
 end if;
end;
SQL> @Z:\plsql\record.sql;
 14  /
Enter value for quantity: 6
old   2:  q inventory.quantity%type:=&quantity;
new   2:  q inventory.quantity%type:=6;
PL/SQL procedure successfully completed.

SQL> select * from purchase_record;
PRODUCT_ID STATUS                        DATES
---------- ------------------------------ ----------
      1004 PURCHASED                      02-SEP-22
```

```
SQL> select * from inventory;

PRODUCT_ID PRODUCT_NAME         QUANTITY
---------- -------------------- ----------
      1001 soap                          3
      1002 brush                         3
      1003 plate                         0
      1004 dress                         5
SQL> @Z:\plsql\record.sql;
  14  /
Enter value for quantity: 0
old   2:  q inventory.quantity%type:=&quantity;
new   2:  q inventory.quantity%type:=0;
PL/SQL procedure successfully completed.

SQL> select * from purchase_record;

PRODUCT_ID STATUS                        DATES
---------- ------------------------------ ----------
      1004 PURCHASED                      02-SEP-22
      1003 OUT OF STOCK                   02-SEP-22
```

7. Create a table employee with eno, ename, and basic_pay attributes, insert 3 to 4 records and write a PL/SQL block to calculate the Gross salary & Net salary for an employee for the following conditions:
   - HRA is 15% of basic.
   - DA is 62% of basic.
   - PF is 780/- if gross salary exceeds 8000, otherwise 600/-.
   - Professional tax is 2% of basic.

   and then print the employee no, name, hra, da, pf, ptax, gross salary & net salary for that employee.

```
SQL> create table employee(eno number(5),ename varchar(30),basic_pay number(10));
```

```
Table created.
SQL> insert into employeee values(&eno,&ename,&basic_pay);
Enter value for eno: 1001
Enter value for ename: 'manya'
Enter value for basic_pay: 12328
old   1: insert into employeee values(&eno,&ename,&basic_pay)
new   1: insert into employeee values(1001,'manya',12328)

1 row created.

SQL> /
Enter value for eno: 1002
Enter value for ename: 'kiran'
Enter value for basic_pay: 45312
old   1: insert into employeee values(&eno,&ename,&basic_pay)
new   1: insert into employeee values(1002,'kiran',45312)

1 row created.

SQL> /
Enter value for eno: 1003
Enter value for ename: 'janvi'
Enter value for basic_pay: 23134
old   1: insert into employeee values(&eno,&ename,&basic_pay)
new   1: insert into employeee values(1003,'janvi',23134)

1 row created.

SQL> /
Enter value for eno: 1004
```

```
Enter value for ename: 'ammu'
Enter value for basic_pay: 54211
old   1: insert into employeee values(&eno,&ename,&basic_pay)
new   1: insert into employeee values(1004,'ammu',54211)

1 row created.

declare
        hra number;
        da number;
        pf number;
        pt number;
        gross_salary number;
        net_salary number;
        b_p employee.basic_pay%type;
        emp employee.ename%type;
        eno employee.eno%type;
begin
        emp:=&Employee;
        select basic_pay into b_p from employee where ename=emp;
        select eno into eno from employee where ename=emp;
        hra:=b_p*0.15;
        da:=b_p*0.62;
        gross_salary:=b_p+hra+da;
        if gross_salary>8000 then
                pf:=780;
        else
                pf:=600;
        end if;
        pt:=0.02*b_p;
        net_salary:=gross_salary-pf-pt;
```

```
        dbms_output.put_line('Employee number = '||eno||', Employee name = '||emp||', HRA =
'||hra||', DA = '||da||', PF = '||pf||', PTax = '||pt||', Gross salary = '||gross_salary||', Net Salary =
'||net_salary);

end;

SQL>  @C:\Users\LENOVO\Desktop\employeee.sql;

 31  /

Enter value for employeename: 'manya'

old  15: emp:=&employeename;

new  15: emp:='manya';

Employee number = 1001, Employee name = manya, HRA = 1849.2, DA = 7643.36, PF =

780, PTax = 246.56, Gross salary = 21820.56, Net Salary = 20794


PL/SQL procedure successfully completed.
```

8. Consider the following relation schemas

Emp

empid	name	salary	dno

Del_History

dno	Rows_deleted	date

Write a PL/SQL block to delete records of all employees who belong to a particular department and then record the dno, no of rows deleted and date on which deletion occurred in the Del_History table using implicit cursors.

```
SQL> create table emp(empid number(5),name varchar(30),salary number(8),dno number(2));


Table created.


SQL> create table del_history(dno number(2),rows_deleted number(5),date1 varchar(10));


Table created.


SQL> select * from emp;
```

```
    EMPID NAME                 SALARY       DNO
---------- -------------------- ---------- ----------
     1001 manya                  42312         2
     1002 kiran                 32134         1
     1003 janvi                 34213         1
     1004 ammu                   65231         2
     1005 satwi                  23134         1

declare
        d emp.dno%type:=&department;
        row_delete number;
begin
        delete from emp where dno=d;
        row_deleted:=sql%rowcount;
        insert into del_history values(d,row_deleted,sysdate);
end;
SQL>  @C:\Users\LENOVO\Desktop\emp2.sql;
 9  /
Enter value for department: 2
old   2: d emp2.dno%type:=&department;
new   2: d emp2.dno%type:=2;

PL/SQL procedure successfully completed.

SQL> select * from del_history;

      DNO ROWS_DELETED DATES
---------- ------------ ----------
        2            2 20-SEP-22
SQL> select * from emp;
```

EMPID	NAME	SALARY	DNO
1002	kiran	32134	1
1003	janvi	34213	1
1005	satwi	23134	1

9. Given the table EMPLOYEE (EmpNo, Name, Salary, Designation, DeptID) write a cursor to select the five highest and lowest paid employees from the table.

```
create table employee(empno number(5),name varchar(30),salary number(8),designation
varchar(10),deptID varchar(10));
```

Table created.

```
SQL> select empno,salary from employee;
```

EMPNO	SALARY
10001	24335
10002	4211
10003	1234
10004	12313
10005	1331
10006	43213
10007	230
10008	2213

8 rows selected.

```
declare
        cursor c is select empno,salary from employee order by salary;
        cursor c1 is select empno,salary from employee order by salary desc;
        e_no employee.empno%type;
```

```
        sal employee.salary%type;
        i number:=0;
begin
        open c;
        open c1;
        dbms_output.put_line('First Highest paid employee');
        dbms_output.put_line('Emp_no Salary');
        for i in 1..5 loop
                exit when c1%notfound;
                fetch c1 into e_no,sal;
                dbms_output.put_line(e_no||' '||sal);
        end loop;
        dbms_output.put_line('First Lowest paid employee');
        dbms_output.put_line('Emp_no Salary');
        for i in 1..5 loop
                exit when c%notfound;
                fetch c into e_no,sal;
                dbms_output.put_line(e_no||' '||sal);
        end loop;
        close c;
        close c1;
end;
```

```
First Highest paid employee
Emp_no Salary
10006 43213
10001 24335
10004 12313
10002 4211
10008 2213
First Lowest paid employee
```

Emp_no Salary

10007 230

10003 1234

10005 1331

10008 2213

10002 4211

PL/SQL procedure successfully completed.

10. Consider the following relation schemas

Bank_Main

Acc_no	Name	Address	Acc-type	Curr_balance

Bank_Trans

Acc_no	Tr_type	Tr_date	Tr_amt	Upd_flag

Where acc_type is S – savings C- current

Tr_type is D – deposit W – withdrawal

Write a PL/SQL block to update master table's (i.e., Bank_Main) curr_balance field with transaction details from transaction file (i.e., Bank_Trans) and also change the status of the Upd_flag field of transaction table to 'Y' once the updation is complete.

SQL> create table bank_main(acc_no number(4),name varchar(20),address varchar(20),acc_type varchar(1),curr_balance number(10));

Table created.

SQL> select * from bank_main;

ACC_NO	NAME	ADDRESS	A	CURR_BALANCE
1001	Ramu	Guntur	S	15000
1002	Bindu	Vizag	S	10000
1003	Abilash	Hyderabad	C	10000
1004	Babu	Vijayawada	C	9000

SQL> create table bank_trans(acc_no number(4),tr_type varchar(1),tr_date varchar(10),tr_amt number(10),upd_flag varchar(1));

Table created.

SQL> select * from bank_trans;

ACC_NO	T	TR_DATE	TR_AMT	U
1001	D	02-09-2002	5000	N
1002	W	02-09-2002	2000	Y
1003	D	02-09-2002	4000	Y
1004	W	02-09-2002	10000	N

```
declare
        negative_amt exception;
        cursor c is select acc_no,tr_type,tr_amt,upd_flag from bank_trans;
        acc_number bank_trans.acc_no%type;
        c_balance bank_main.curr_balance%type;
        trans_type bank_trans.tr_type%type;
        tr_amount bank_trans.tr_amt%type;
        flag bank_trans.upd_flag%type;
begin
        open c;
        loop
                exit when c%notfound;
                fetch c into acc_number,trans_type,tr_amount,flag;
                select curr_balance into c_balance from bank_main where acc_no=acc_number;
                if c_balance-tr_amount<0 then
                        raise negative_amt;
                else
                        if flag='N' then
                                if trans_type='D' then
```

```
                                        update bank_main set curr_balance=c_balance+tr_amount
where acc_no=acc_number;
                                elsif trans_type='W' then
                                        update bank_main set curr_balance=c_balance-tr_amount
where acc_no=acc_number;
                                end if;
                                update bank_trans set upd_flag='Y' where acc_no=acc_number;
                        end if;
                end if;
        end loop;
        close c;
exception
        when negative_amt then
                dbms_output.put_line('Transaction do not takes place for account number =
'||acc_number);
                update bank_trans set upd_flag='Y' where acc_no=acc_number;
end;
```

```
SQL> @C:\Users\kallu\Desktop\dbms_lab\trans.sql;
 35  /
Transaction do not takes place for account number = 1004

PL/SQL procedure successfully completed.

SQL> select * from bank_main;

   ACC_NO NAME                 ADDRESS              A CURR_BALANCE
---------- -------------------- -------------------- - ------------
      1001 Ramu                 Guntur               S        20000
      1002 Bindu                Vizag                S        10000
      1003 Abilash              Hyderabad            C        10000
```

1004 Babu Vijayawada C 9000

SQL> select * from bank_trans;

ACC_NO T TR_DATE TR_AMT U

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

1001 D 02-09-2022 5000 Y

1002 W 02-09-2022 2000 Y

1003 D 02-09-2022 4000 Y

1004 W 02-09-2022 10000 Y

11. Write a PL/SQL block to handle the following built-in exceptions

no_data_found

too_many_rows

zero_divide

SQL> create table stud(id number(3),dept varchar(5));

Table created.

SQL> select * from stud;

ID DEPT

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

Y20CS009 CSE

Y20EC019 ECE

Y20CS074 CSE

Y20CS134 CSE

Y20EC193 ECE

```
declare
        a number:=&number1;
        b number:=&number2;
        c number;
        stu_id stud.id%type;
begin
        select id into stu_id from stud where dept='EEE';
        select id into stu_id from stud where dept='CSE';
        c:=a/b;
exception
        when no_data_found then
                dbms_output.put_line('No rows selected');
        when too_many_rows then
                dbms_output.put_line('More than one row is selected');
        when zero_divide then
                dbms_output.put_line('A number can not divide by zero');
end;
```

```
Enter value for number1: 12
old   2:       a number:=&number1;
new   2:       a number:=12;
Enter value for number2: 0
old   3:       b number:=&number2;
new   3:       b number:=0;
No rows selected

PL/SQL procedure successfully completed.
```

12. Write a PL/SQL block to check whether the quantity of any product in Inventory table is < 0. If so, using an exception display relevant message and update quantity to 0.

Inventory

Product_ID	Product_name	Quantity

```
SQL> create table inventory(product_id number(5),product_name varchar(20),quantity number(3));
Table created.
SQL> select * from inventory;
PRODUCT_ID PRODUCT_NAME         QUANTITY
---------- -------------------- ----------
    12312 Add wash                 10
    12313 Gril                     -1
    12314 Power Bot                 3
    12315 Galaxy s7                 5
    12316 Notebook 9                17
    12317 4 Door Flex               3
6 rows selected.
```

```
declare
        cursor c is select quantity from inventory;
        neagtive_quantity EXCEPTION;
        q inventory.quantity%type;
begin
        open c;
        loop
                exit when c%notfound;
                fetch c into q;
                if q<0 then
                        raise neagtive_quantity;
                end if;
        end loop;
exception
        when neagtive_quantity then
```

```
                dbms_output.put_line('The quantity of the product is less than zero');
                update inventory set quantity=0 where quantity=q;
end;

SQL> @C:\Users\kallu\Desktop\dbms_lab\plsql12.sql;
 21  /
The quantity of the product is less than zero
PL/SQL procedure successfully completed.

SQL> select * from inventory;
PRODUCT_ID PRODUCT_NAME          QUANTITY
---------- -------------------- ----------
    12312 Add wash                    10
    12313 Gril                     0
    12314 Power Bot                    3
    12315 Galaxy s7                   5
    12316 Notebook 9                  17
    12317 4 Door Flex                 3
6 rows selected.
```

1) Write a stored procedure, raise_salary which accepts an employee number, increment and modifies salary of that employee in employee table. Modified salary = salary increase amount+ current salary. If employee number is not found or if the current salary is null, it should raise an exception. Otherwise, updates the salary.

```
SQL> select * from employee;
   EMPNO     SALARY
---------- ----------
    1001      74863
    1002       4572
    1003       8572
    1004      12046
    1005      54321
    1006
```

```
create or replace procedure raise_salary(eno in number,inc_sal in number) as
        salary_null exception;
        sal employee.salary%type;
        begin
                select salary into sal from employee where empno=eno;
                if sal is null then
                        raise salary_null;
                end if;
                update employee set salary=sal+inc_sal where empno=eno;
        exception
                when no_data_found then
                        dbms_output.put_line('The Employee is not found');
                when salary_null then
                        dbms_output.put_line('The Salary is null');
end;

declare
        eno employee.empno%type:=&Employee_Number;
        inc_sal employee.salary%type:=&Modified_Salary;
begin
        raise_salary(eno,inc_sal);
end;

SQL> @C:\Users\kallu\Desktop\dbms_lab\proc_main.sql;
 8  /
Enter value for employee_number: 1001
old   2:       eno employee.empno%type:=&Employee_Number;
new   2:       eno employee.empno%type:=1001;
Enter value for modified_salary: 212
old   3:       inc_sal employee.salary%type:=&Modified_Salary;
new   3:       inc_sal employee.salary%type:=212;
```

```
PL/SQL procedure successfully completed.

SQL> /
Enter value for employee_number: 1009
old   2:       eno employee.empno%type:=&Employee_Number;
new   2:       eno employee.empno%type:=1009;
Enter value for modified_salary: 312
old   3:       inc_sal employee.salary%type:=&Modified_Salary;
new   3:       inc_sal employee.salary%type:=312;
The Employee is not found

PL/SQL procedure successfully completed.

SQL> select * from employee;

    EMPNO     SALARY
---------- ----------
     1001      75075
     1002       4572
     1003       8572
     1004      12046
     1005      54321

SQL> @Z:\plsql\proc_call.sql;
  7  /
Enter value for employee_number: 1006
old   2:       eno employee.empno%type:=&Employee_Number;
new   2:       eno employee.empno%type:=1006;
Enter value for modified_salary: 123
old   3:       inc_sal employee.salary%type:=&Modified_Salary;
```

new 3: inc_sal employee.salary%type:=123;

The Salary is null

PL/SQL procedure successfully completed.

2) Write a PL/SQL function that accepts department number and returns the total salary of the department

```
create table dept(deptno number(2),salary number(10));
```

Table created.

```
SQL> select * from dept;

     DNO    SALARY
---------- ----------
      10     23451
      30     12345
      20      1344
      20     23562
      20     45253
      10      1244
```

6 rows selected.

```
create or replace function dept_sal(dept_no number) return number as
        sal dept.salary%type;
begin
        select sum(salary) into sal from dept where dno=dept_no group by dno;
        return sal;
end;

declare
        dept_no dept.dno%type:=&Dept_no;
        tot_sal dept.salary%type;
begin
```

```
        tot_sal:=dept_sal(dept_no);
        dbms_output.put_line('The total salary for department number '||dept_no||' = '||tot_sal);
end;
```

```
SQL> @Z:\plsql\fun.sql;
  7  /

Function created.

SQL> @Z:\plsql\fun_call.sql;
  8  /
Enter value for dept_no: 20
old   2:       dept_no dept.dno%type:=&Dept_no;
new   2:       dept_no dept.dno%type:=20;
The total salary for department number 20 = 70159

PL/SQL procedure successfully completed.
```

3) Write a PL/SQL block that computes increment of an employee in employee table by using incr function which takes employee number as argument, calculates increment and returns the same based on the following criteria:
   - If salary <= 3000 – increment = 30% of salary
   - If salary > 3000 and <= 6000– increment = 20% of salary
   - Else increment = 10% of salary.

```
SQL> create table employee(eno number(5),salary number(5));
Table created.
SQL> select * from employee;
     ENO    SALARY
---------- ----------
    10001     12000
    10002      3000
```

```
   10003      4000
create or replace function incr(e_no number) return number as
        sal employee.salary%type;
        increment number;
        begin
                select salary into sal from employee where eno=e_no;
                if sal<=3000 then
                        increment:=0.3*sal;
                elsif sal<=6000 then
                increment:=0.2*sal;
                else
                        increment:=0.1*sal;
                end if;
                return increment;
        end;

declare
        e_no employee.eno%type:=&employee_number;
        sal employee.salary%type;
        increment number;
begin
        select salary into sal from employee where eno=e_no;
        increment:=incr(e_no);
        dbms_output.put_line('The increment salary of salary='||sal||' is '||increment);
end;

SQL> @Z:\plsql\fun_emp.sql;
 15  /
Function created.
SQL> @Z:\plsql\fun_emp_call.sql;
 10  /
```

```
Enter value for employee_number: 10001
old   2:       e_no employee.eno%type:=&employee_number;
new   2:       e_no employee.eno%type:=10001;
The increment salary of salary=12000 is 1200

PL/SQL procedure successfully completed.

SQL> /
Enter value for employee_number: 10002
old   2:       e_no employee.eno%type:=&employee_number;
new   2:       e_no employee.eno%type:=10002;
The increment salary of salary=3000 is 900

PL/SQL procedure successfully completed.

SQL> /
Enter value for employee_number: 10003
old   2:       e_no employee.eno%type:=&employee_number;
new   2:       e_no employee.eno%type:=10003;
The increment salary of salary=4000 is 800

PL/SQL procedure successfully completed.
```

4) Write a stored procedure that displays the employee names and their total earnings from the Emp Table. *Hint: Total earning of an employee = 12*(gross_salary+commission)*

```
SQL> create table emp(ename varchar(20),gross_sal number(5),commission number(5));
Table created.
SQL> select * from emp;
```

```
ENAME                GROSS_SAL COMMISSION
-------------------- ---------- ----------
Sravani                 2000       1000
Pragathi                2500       5000
Vijaya                  1200       500
Praveena                3000       1000
```

```
create or replace procedure tot_earning as
cursor c is select * from emp;
emp_name emp.ename%type;
e_gsal emp.gross_sal%type;
e_com emp.commission%type;
tot_earn number;
begin
        open c;
        dbms_output.put_line('Emp Name Total Earning');
        loop
                exit when c%notfound;
                fetch c into emp_name,e_gsal,e_com;
                tot_earn:=12*(e_gsal+e_com);
                dbms_output.put_line(emp_name||'   '||tot_earn);
        end loop;
        close c;
end;

begin
tot_earning;
end;
SQL> @Z:\plsql\pro_emp.sql;
 18  /
```

Procedure created.

SQL> @Z:\plsql\pro_emp_call.sql;

4 /

Emp Name Total Earning

Sravani 36000

Pragathi 90000

Vijaya 20400

Praveena 48000

Praveena 48000

PL/SQL procedure successfully completed.

5) Create a database trigger that checks whether the new salary of employee is less than existing salary. If so, raise an appropriate exception and avoid that updation.

SQL> select empno,sal from emp;

EMPNO	SAL
7369	800
7499	1600
7521	1250
7566	2975
7654	1250

```
create or replace trigger emp_newsal before insert or update on emp
for each row
begin
        if (:new.sal<:old.sal) then
                raise_application_error(-20000,'You cannot update salary');
        end if;
```

```
end;

SQL> @C:\Users\kallu\Desktop\dbms_lab\tigger.sql;
 8 /

Trigger created.

SQL> update emp set sal=1000 where empno=7369;

1 row updated.

SQL> update emp set sal=500 where empno=7499;
update emp set sal=500 where empno=7499
       *
ERROR at line 1:
ORA-20000: You cannot update salary
ORA-06512: at "SCOTT.EMP_NEWSAL", line 3
ORA-04088: error during execution of trigger 'SCOTT.EMP_NEWSAL'
```

6) Consider the following tables

PERSINFO

EMPNO	NAME	AGE

AUDITPERSINFO

EMPNO	NAME	AGE	OPERATION	ODATE

PERSINFO is the table for which the auditing must be performed and AUDITPERSINFO is the table which keeps track of the records deleted or modified. Create a database trigger audit_trial. This trigger is forced when an UPDATE or a DELETE is performed on the table PERSINFO. It first checks for the operation being performed on the table. Then depending on the operation, a variable (that corresponds to operation) is assigned the value 'UPDATE' or 'DELETE' and then inserts the updated/deleted record into AUDITPERSINFO.

SQL> create table persinfo(empno number(5),name varchar(20),age number(2));

Table created.

SQL> create table auditpersinfo(empno number(5),name varchar(20),age number(2),operation varchar(10),odate date);

Table created.

SQL> select * from persinfo;

EMPNO	NAME	AGE
1001	Sravani	25
1002	Vindhya	30
1003	Pragathi	24
1004	Vijaya	31
1005	Praveena	27

```
create or replace trigger audit_trial after update or delete on persinfo
for each row
declare
        op auditpersinfo.operation%type;
begin
        if updating then
                op:='UPDATE';
        end if;
        if deleting then
                op:='DELETE';
        end if;
        insert into auditpersinfo values(:old.empno,:old.name,:old.age,op,sysdate);
end;
```

```
SQL> update persinfo set age=40 where empno=1003;

1 row updated.

SQL> delete from persinfo where empno=1004;

1 row deleted.

SQL> select * from persinfo;

    EMPNO NAME                 AGE
---------- -------------------- ----------
     1001 Sravani               25
     1002 Vindhya               30
     1003 Pragathi              40
     1005 Praveena              27

 SQL> select * from auditpersinfo;

    EMPNO NAME                 AGE OPERATION  ODATE
---------- -------------------- ---------- ---------- ---------
     1003 Pragathi              24 UPDATE     16-SEP-22
     1004 Vijaya                31 DELETE     16-SEP-22
```