LAB CYCLE 2

1.Write a PL/SQL code to accept the text and reverse the given text. Check the text is palindrome or not.

PL/SQL CODE

```
DECLARE
  a VARCHAR(15):='MALAYALAM';
  b VARCHAR(15);
  n NUMBER;
 BEGIN
  n:=LENGTH(a);
  FOR I IN REVERSE 1..n
  LOOP
   b:=b || SUBSTR(a,I,1);
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Reversed String: '||b);
  n:=INSTR(a,b);
  IF n!=1 THEN
  DBMS_OUTPUT.PUT_LINE(b ||' is not a paliandrom');
  ELSE
  DBMS_OUTPUT.PUT_LINE(b ||' is  a paliandrom');
  END IF;
  END;
```

OUTPUT

```
DECLARE
 a VARCHAR(15):='MALAYALAM';
 b VARCHAR(15);
 n NUMBER;
BEGIN
 n:=LENGTH(a);
 FOR I IN REVERSE 1..n
 LOOP
  b:=b || SUBSTR(a,I,1);
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Reversed String: '||b);
 n:=INSTR(a,b);
 IF n!=1 THEN
 DBMS_OUTPUT.PUT_LINE(b ||' is not a paliandrom');
 ELSE
 DBMS_OUTPUT.PUT_LINE(b ||' is  a paliandrom');
 END IF;
 END;

Statement processed.
Reversed String: MALAYALAM
MALAYALAM is a paliandrom
```

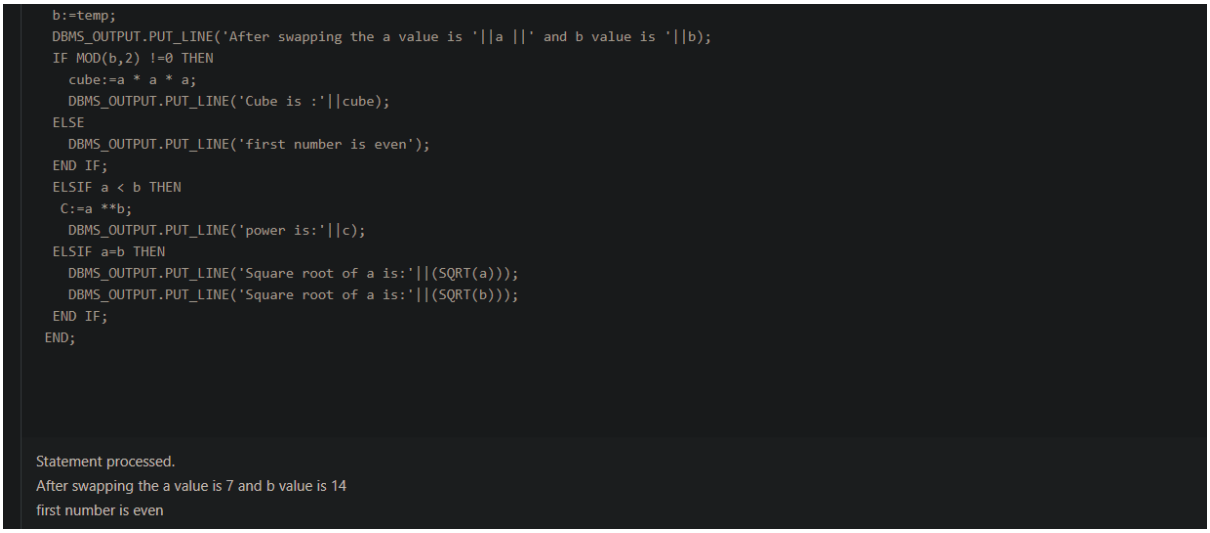
2. Write a program to read two numbers; If the first no > 2nd no, then swap the numbers; if the first number is an odd number, then find its cube; if first no < 2nd no then raise it to its power; if both the numbers are equal, then find its sqrt.

PL/SQL CODE

```
DECLARE
 a INTEGER:=14;
 b INTEGER:=7;
 temp INTEGER:=0;
 c INTEGER;
 cube INTEGER;
BEGIN
 IF a > b THEN
```

```
temp:=a;
a:=b;
b:=temp;
DBMS_OUTPUT.PUT_LINE('After swapping the a value is '||a ||' and b value is '||b);
IF MOD(b,2) !=0 THEN
  cube:=a * a * a;
  DBMS_OUTPUT.PUT_LINE('Cube is :'||cube);
ELSE
  DBMS_OUTPUT.PUT_LINE('first number is even');
END IF;
ELSIF a < b THEN
 C:=a **b;
  DBMS_OUTPUT.PUT_LINE('power is:'||c);
ELSIF a=b THEN
  DBMS_OUTPUT.PUT_LINE('Square root of a is:'||(SQRT(a)));
  DBMS_OUTPUT.PUT_LINE('Square root of a is:'||(SQRT(b)));
END IF;
END;
```

OUTPUT



3.Write a program to generate first 10 terms of the Fibonacci series

PL/SQL CODE

```
DECLARE
 a NUMBER:=0;
 b NUMBER:=1;
 fibo Number;
BEGIN
 DBMS_OUTPUT.PUT_LINE(a);
 DBMS_OUTPUT.PUT_LINE(b);
 fibo:=a+b;
 DBMS_OUTPUT.PUT_LINE(fibo);
 FOR i IN 4.. 10
 LOOP
  a:=b;
  b:=fibo;
  fibo:=a+b;
  DBMS_OUTPUT.PUT_LINE(fibo);
 END LOOP;
END;
```

OUTPUT

```
DECLARE
 a NUMBER:=0;
 b NUMBER:=1;
 fibo Number;
BEGIN
 DBMS_OUTPUT.PUT_LINE(a);
 DBMS_OUTPUT.PUT_LINE(b);
 fibo:=a+b;
 DBMS_OUTPUT.PUT_LINE(fibo);
 FOR i IN 4.. 10
 LOOP
  a:=b;
  b:=fibo;
  fibo:=a+b;
  DBMS_OUTPUT.PUT_LINE(fibo);
 END LOOP;
END;

Statement processed.
0
1
1
2
3
5
8
13
21
34
```

4).Write a PL/SQL program to find the salary of an employee in the EMP table (Get the empno from the user). Find the employee drawing minimum salary. If the minimum salary is less than 7500, then give an increment of 15%. Also create an emp %rowtype record. Accept the empno from the user, and display all the information about the employee.

PL/SQL CODE

```
1  create table employee1(emp_no int,emp_name varchar(20),emp_post varchar(20),emp_salary decimal(20,4));

Table created.
```

```
1  insert into employee1 values(221,'Anna','MD',25000);
2  insert into employee1 values(231,'Anju','HR',20000);
3  insert into employee1 values(241,'Krishna','Accountant',20000);
4  insert into employee1 values(211,'Mohan','Clerk',15000);
5  insert into employee1 values(251,'Mani','Peon',10000);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
Declare
 emno employee1.emp_no%type;
 salary employee1.emp_salary%type;
 emp_rec employee1%rowtype;
begin
 emno:=251;
 select emp_salary into salary from employee1 where emp_no=emno;
 if salary<7500 then
  update employee1 set emp_salary=emp_salary * 15/100 where
  emp_no=emno;
 else
 dbms_output.put_line('No more increment');
 end if;
 select * into emp_rec from employee1 where emp_no=emno;
 dbms_output.put_line('Employee num: '||emp_rec.emp_no);
 dbms_output.put_line('Employee name: '||emp_rec.emp_name);
 dbms_output.put_line('Employee post: '||emp_rec.emp_post);
```

dbms_output.put_line('Employee salary: '||emp_rec.emp_salary);

end;

OUTPUT

```
Declare
 emno employee1.emp_no%type;
 salary employee1.emp_salary%type;
 emp_rec employee1%rowtype;
begin
 emno:=251;
 select emp_salary into salary from employee1 where emp_no=emno;
 if salary<7500 then
  update employee1 set emp_salary=emp_salary * 15/100 where
  emp_no=emno;
 else
 dbms_output.put_line('No more increment');
 end if;
 select * into emp_rec from employee1 where emp_no=emno;
 dbms_output.put_line('Employee num: '||emp_rec.emp_no);
 dbms_output.put_line('Employee name: '||emp_rec.emp_name);
 dbms_output.put_line('Employee post: '||emp_rec.emp_post);
 dbms_output.put_line('Employee salary: '||emp_rec.emp_salary);
end;
```

```
Statement processed.
No more increment
Employee num: 251
Employee name: Mani
Employee post: Peon
Employee salary: 10000
```

5) Write a PL/SQL function to find the total strength of students present in different classes of the MCA department using the table Class(ClassId, ClassName, Strength);

PL/SQL CODE & OUTPUT

Table creation

```
create table class2(cls_id int,cls_name varchar(20),cls_std int);
```

```
Table created.
```

Insertion command

```
insert into class2 values(333,'mca',60);
insert into class2 values(332,'mca',60);
insert into class2 values(334,'mba',50);
insert into class2 values(335,'msc',55);
insert into class2 values(336,'msw',70);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Create function code

```
CREATE OR REPLACE  FUNCTION total_std
 RETURN NUMBER IS
 total NUMBER(5):=0;
 BEGIN
 SELECT sum(cls_std) INTO total FROM class2 WHERE cls_name='mca';
 RETURN total;
 END;
```

```
Function created.
```

```
DECLARE
 c NUMBER(5);
BEGIN
 c:=total_std();
 DBMS_OUTPUT.PUT_LINE('Total students in MCA department is:'||c);
END;
```

```
Statement processed.
Total students in MCA department is:120
```

6) Write a PL/SQL **procedure** to increase the salary for the specified employee. Using empno in the employee table based on the following criteria: increase the salary by 5% for clerks, 7% for salesman, 10% for analyst and 20 % for manager. Activate using PL/SQL block.

PL/SQL CODE

Table creation

```
1  create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20));
```

```
Table created.
```

Insertion command

```
1  insert into emp values(1001,'Keerthi',20000,'Manager');
2  insert into emp values(1002,'Mia',6500,'Salesman');
3  insert into emp values(1003,'Athira',7500,'Clerk');
4  insert into emp values(1004,'Ankitha',7400,'Analyst');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Procedure code

```
CREATE OR REPLACE PROCEDURE increSalary
IS
emp1 emp%rowtype;
sal emp.salary%type;
dpt emp.emp_dpt%type;
BEGIN
SELECT salary,emp_dpt INTO sal,dpt FROM emp WHERE emp_no = 1004;
   IF dpt ='clerk' THEN
    UPDATE emp SET salary = salary+salary* 5/100 ;
   ELSIF dpt = 'salesman' THEN
    UPDATE emp SET salary = salary+salary* 7/100  ;
   ELSIF dpt = 'analyst' THEN
    UPDATE emp SET salary = salary+salary* 10/100  ;
  ELSIF dpt = 'manager' THEN
    UPDATE emp SET salary = salary+salary* 20/100  ;
  ELSE
    DBMS_OUTPUT.PUT_LINE ('NO INCREMENT');
   END IF;
   SELECT * into emp1 FROM emp WHERE emp_no = 1004;
   DBMS_OUTPUT.PUT_LINE ('Name: '||emp1.emp_name);
   DBMS_OUTPUT.PUT_LINE ('employee number: '||emp1.emp_no);
   DBMS_OUTPUT.PUT_LINE ('salary: '|| emp1.salary);
   DBMS_OUTPUT.PUT_LINE ('department: '|| emp1.emp_dpt);
END;
```

```
Procedure created.
```

OUTPUT

```
DECLARE
BEGIN
  increSalary();
END;
```

```
Statement processed.
NO INCREMENT
Name: Ankitha
employee number: 1004
salary: 7400
department: Analyst
```

7)Create a **cursor** to modify the salary of ‘president’ belonging to all departments by 50%

PL/SQL CODE

Table creation

```
1  create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20),dsgt varchar(20));
```

```
Table created.
```

Insertion command

```
1  insert into emp values(2001,'Athira',50000,'sales','president');
2  insert into emp values(2002,'Aparna',6500,'Ac','president');
3  insert into emp values(2003,'Thara',7500,'HR','manager');
4  insert into emp values(2004,'Neethu',7500,'Ac','senior grade');
5  insert into emp values(2005,'Nithya',7500,'HR','president');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Trigger code

SQL Worksheet

```
1  DECLARE
2      total_rows number(2);
3      emp1 EMP%rowtype;
4  BEGIN
5
6   UPDATE emp SET salary = salary + salary * 50/100 where dsgt = 'president';
7   IF sql%notfound THEN
8       dbms_output.put_line('no employee salary updated');
9   ELSIF sql%found THEN
10     total_rows := sql%rowcount;
11     dbms_output.put_line( total_rows || ' employee salary details updated');
12  end if;
13 end;
14
```

```
Statement processed.
3 employee salary details updated
```

```
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' employee salary details updated');
 end if;
end;*/
select * from emp;
```

EMP_NO	EMP_NAME	SALARY	EMP_DPT	DSGT
2001	Athira	75000	sales	president
2002	Aparna	9750	Ac	president
2003	Thara	7500	HR	manager
2004	Neethu	7500	Ac	senior grade
2005	Nithya	11250	HR	president

Download CSV

5 rows selected.

8)Write a **cursor** to display list of Male and Female employees whose name starts with S.

PL/SQL CODE

Table creation

```
create table employ1(emp_no varchar(20),emp_name varchar(20),emp_post varchar(20),emp_salary int);
```

Table created.

Insertion command

```
insert into employ1 values(2002,'Surya','MD',30000);
insert into employ1 values(2003,'Sruthi','HR',25000);
insert into employ1 values(2004,'Punnya','ACC',20000);
insert into employ1 values(2005,'Arun','Clerk',15000);
insert into employ1 values(2006,'John','Peon',10000);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Cursor code

```
DECLARE
    CURSOR emp1 IS
      SELECT emp_no,emp_name,emp_post,emp_salary  FROM employ1 where emp_name like ('S%') ;
    emp2 emp1%ROWTYPE;
BEGIN
    OPEN emp1;

    LOOP
        FETCH emp1 INTO emp2;

        EXIT WHEN emp1%NOTFOUND;

        dbms_output.Put_line('Employee_ID: ' ||emp2.emp_no);
        dbms_output.Put_line('Employee_Name: ' ||emp2.emp_name);
        dbms_output.Put_line('Employee_post: ' ||emp2.emp_post);
        dbms_output.Put_line('Employee_salary: '||emp2.emp_salary);
    END LOOP;
    CLOSE emp1;
END;
```

```
Statement processed.
Employee_ID: 2002
Employee_Name: Surya
Employee_post: MD
Employee_salary: 30000
Employee_ID: 2003
Employee_Name: Sruthi
Employee_post: HR
Employee_salary: 25000
```

OUTPUT

```
Statement processed.
Employee_ID: 2002
Employee_Name: Surya
Employee_post: MD
Employee_salary: 30000
Employee_ID: 2003
Employee_Name: Sruthi
Employee_post: HR
Employee_salary: 25000
```

9)Create the following tables for Library Information System: Book : (accession-no, title, publisher, publishedDate, author, status). Status could be issued, present in the library, sent for binding, and cannot be issued. Write a **trigger** which sets the status of a book to "cannot be issued", if it is published 15 years back.

PL/SQL CODE

Table creation

```
1  create table book(accession_no int , title varchar(20), publisher varchar(20), publishedDate date, author varchar(20), status varchar(30));
```

```
Table created.
```

Insertion command

```
1  insert into book values( 2011,'wings of fire','dc','21-jan-2009','APJ','issued');
2  insert into book values( 2012,'Freedom','dc','30-mar-2010','john mathew','present in the library');
3  insert into book values( 2013,'wings','dc','21-june-2011','Adithya','sent for binding');
4  insert into book values( 2014,'Home','dc','01-sep-2016','johnswook','issued');
5  insert into book values( 2015,'The sea','dc','21-jan-2004','sachi','can not be issued');
6  insert into book values( 2016,'memories','dc','21-jan-2006','jk roll',' issued');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Trigger code

```
CREATE OR REPLACE TRIGGER search1
 before insert  ON book
 FOR EACH ROW
 declare
  temp date;
BEGIN
 select sysdate into temp from dual;
 if inserting  then
  if :new.publishedDate < add_months(temp, -180) then
     :new.status:='cannot be issued' ;
  end if;
 end if;
end;
```

```
Trigger created.
```

```
select * from book;
```

ACCESSION_NO	TITLE	PUBLISHER	PUBLISHEDDATE	AUTHOR	STATUS
2011	wings of fire	dc	21-JAN-09	APJ	issued
2012	Freedom	dc	30-MAR-10	john mathew	present in the library
2013	wings	dc	21-JUN-11	Adithya	sent for binding
2014	Home	dc	01-SEP-16	johnswook	issued
2015	The sea	dc	21-JAN-04	sachi	can not be issued
2016	memories	dc	21-JAN-06	jk roll	issued

Download CSV

6 rows selected.

10) Create a table Inventory with fields pdtid, pdtname, qty and reorder_level. Create a **trigger** control on the table for checking whether qty<reorder_level while inserting values.

PL/SQL CODE & OUTPUT

Table creation

```
1   create table inventory(pdtid number primary key, pdtname varchar(10), qty int,reorder_level number);
```

```
Table created.
```

Insertion command

```
1  insert into inventory values(101,'pencil',100,150);
2  insert into inventory values(112,'tap',50,100);
3  insert into inventory values(121,'marker',200,150);
4  insert into inventory values(151,'notbook',500,250);
5
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Trigger code

```
1  CREATE OR REPLACE TRIGGER checking
2  before insert  ON inventory
3   FOR EACH ROW
4  declare
5  BEGIN
6   if inserting  then
7    if :new.qty > :new.reorder_level then
8        :new.reorder_level:=0;
9    end if;
10  end if;
11 end;
12
```

```
Trigger created.
```

OUTPUT

```
1  select * from inventory;
2
```

PDTID	PDTNAME	QTY	REORDER_LEVEL
101	pencil	100	150
112	tap	50	100
121	marker	200	150
151	notbook	500	250

Download CSV

4 rows selected.