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
 s VARCHAR2(10) := 'malayalam';
 l VARCHAR2(20);
 t VARCHAR2(10);
BEGIN
 FOR i IN REVERSE 1..Length(s) LOOP
 l := Substr(s, i, 1);
 t := t||''||l;
 END LOOP;
 IF t = s THEN
 dbms_output.Put_line(t ||''||' is palindrome');
 ELSE
 dbms_output.Put_line(t||''||' is not palindrome');
 END IF;
END;
```

OUTPUT

```
1  DECLARE
2   s VARCHAR2(10) := 'malayalam';
3   l VARCHAR2(20);
4   t VARCHAR2(10);
5  BEGIN
6   FOR i IN REVERSE 1..Length(s) LOOP
7   l := Substr(s, i, 1);
8   t := t||''||l;
9   END LOOP;
10  IF t = s THEN
11  dbms_output.Put_line(t ||''||' is palindrome');
12  ELSE
13  dbms_output.Put_line(t||''||' is not palindrome');
14  END IF;
15 END;
```

```
Statement processed.
malayalam is palindrome
```

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:=8;
 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 b is :'||(SQRT(b)));
END IF;
END;
```

OUTPUT

```
DECLARE
 a INTEGER:=14;
 b INTEGER:=8;
 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);
```

```
Statement processed.
After swapping the a value is 8 and b value is 14
first number is even
```

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

PL/SQL CODE

```
DECLARE
 a NUMBER:=0;
 b NUMBER:=1;
 c NUMBER;
BEGIN
 DBMS_OUTPUT.PUT(a||' '||B||' ');
 FOR I IN 3..10 LOOP
 c:=a+b;
 DBMS_OUTPUT.PUT(c||' ');
 a:=b;
 b:=c;
 END LOOP;
DBMS_OUTPUT.PUT_LINE(' ');
END;
```

OUTPUT

```
DECLARE
 a NUMBER:=0;
 b NUMBER:=1;
 c NUMBER;
BEGIN
 DBMS_OUTPUT.PUT(a||' '||B||' ');
 FOR I IN 3..10 LOOP
 c:=a+b;
 DBMS_OUTPUT.PUT(c||' ');
 a:=b;
 b:=c;
 END LOOP;
DBMS_OUTPUT.PUT_LINE(' ');
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 COD

```
create table employeee(emp_no int,emp_name varchar(20),emp_post
varchar(20),emp_salary decimal(10,2));
Table created.
```

```
Table created.
```

```
1  insert into employeee values(101,'Rahul','MD',25000);
2  insert into employeee values(102,'Kripa','HR',25000);
3  insert into employeee values(103,'Anju','Clerk',25000);
4  insert into employeee values(104,'Arya','Accountant',25000);
5  insert into employeee values(201,'Singhu','Peon',25000);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Declare

emno employee.emp_no%type;

salary employee.emp_salary%type;

emp_rec employee%rowtype;

begin

emno:=104;

select emp_salary into salary from employee where emp_no=emno;

if salary<7500 then

update employee 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 employee 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

```
 emp_rec employee%rowtype;
begin
 emno:=104;
 select emp_salary into salary from employee where emp_no=emno;
 if salary<7500 then
 update employee 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 employee 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: 104
Employee name: Manu
Employee post: Clerk
Employee salary: 20000
```

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

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

```
Table created.
```

```sql
insert into class values(302,'mca',60);
insert into class values(303,'mca',60);
insert into class values(304,'msw',57);
insert into class values(305,'mba',77);
insert into class values(306,'msc',87);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
1  CREATE OR REPLACE FUNCTION total_std
2   RETURN NUMBER IS
3   total NUMBER(5):=0;
4   BEGIN
5   SELECT sum(cls_std) INTO total FROM class WHERE cls_name='mca';
6   RETURN total;
7   END;
8
9
```

```
Function created.
```

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

OUTPUT

```
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

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

```
Table created.
```

```
1  insert into emp values(201,'arya',60000,'salesman');
2  insert into emp values(202,'anju',8500,'manager');
3  insert into emp values(203,'sindhu',6500,'clerk');
4  insert into emp values(204,'Ram',14500,'analyst');
5
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

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 =
104;

  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 = 104;
  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;
```

```
1  CREATE OR REPLACE PROCEDURE increSalary
2  IS
3  emp1 emp%rowtype;
4  sal emp.salary%type;
5  dpt emp.emp_dpt%type;
6  BEGIN
7  SELECT salary,emp_dpt INTO sal,dpt FROM emp WHERE emp_no = 204;
8     IF dpt ='clerk' THEN
9       UPDATE emp SET salary = salary+salary* 5/100 ;
10    ELSIF dpt = 'salesman' THEN
11      UPDATE emp SET salary = salary+salary* 7/100  ;
12    ELSIF dpt = 'analyst' THEN
13      UPDATE emp SET salary = salary+salary* 10/100  ;
14   ELSIF dpt = 'manager' THEN
15      UPDATE emp SET salary = salary+salary* 20/100  ;
```

```
Procedure created.
```

CODE& OUTPUT

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

```
1  DECLARE
2  BEGIN
3    increSalary();
4  END;
```

```
Statement processed.
Name: Ram
employee number: 204
salary: 15950
department: analyst
```

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

CODE

Createtable

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

```
Table created.
```

Insert

```
1  insert into emp values(111,'arun',50000,'sales','president');
2  insert into emp values(112,'appu',6500,'Ac','president');
3  insert into emp values(113,'ammu',7500,'HR','manager');
4  insert into emp values(114,'arrya',7500,'Ac','snr grade');
5  insert into emp values(115,'manu.c',7500,'HR','president');
6
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
DECLARE
    total_rows number(2);
    emp1 EMP%rowtype;
BEGIN

 UPDATE emp SET salary = salary + salary * 50/100 where dsgt = 'president';
 IF sql%notfound THEN
     dbms_output.put_line('no employee salary updated');
 ELSIF sql%found THEN
    total_rows := sql%rowcount;
    dbms_output.put_line( total_rows || ' employee salary details  updated');
 end if;
end;
```

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

OUPUT

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

```
1  select * from emp;
2
```

EMP_NO	EMP_NAME	SALARY	EMP_DPT	DSGT
111	arun	75000	sales	president
112	appu	9750	Ac	president
113	ammu	7500	HR	manager
114	arrya	7500	Ac	snr grade
115	manu.c	11250	HR	president

Download CSV

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

```
1  create table employ(emp_no varchar(20),emp_name varchar(20),emp_post varchar(20),Salary int);
```

```
Table created.
```

```
1  insert into employ values(101,'Sandra','Sby',35000);
2  insert into employ values(102,'Manu','Kalpetta',65000);
3  insert into employ values(103,'Mahima','Calicut',15000);
4  insert into employ values(104,'Sinoj','Kcr',25000);
5  insert into employ values(105,'Sinto','Mnc',425000);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
DECLARE
    CURSOR emp1 IS
      SELECT emp_no,emp_name,emp_post,Salary  FROM employ 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.Salary);
    END LOOP;
    CLOSE emp1;
END;
```

```
Statement processed.
Employee_Name: Sandra
Employee_post: Sby
Employee_salary: 35000
Employee_ID: 104
Employee_Name: Sinoj
Employee_post: Kcr
Employee_salary: 25000
Employee_ID: 105
Employee_Name: Sinto
Employee_post: Mnc
Employee_salary: 425000
```

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.

CODE

CREATE TABLE

```
1  create table book(accession_no int,tittle varchar(20),publisher varchar(20),publishedDate date,author varchar(20),status varchar(30));
2
```

```
Table created.
```

INSERT

```
1  insert into book values(2231,'hapy','cd','21-mar-2009','Maria','issued');
2  insert into book values(2242,'sad','cp','19-jun-2010','Meenu','issued');
3  insert into book values(2253,'star','mv','11-mar-2016','Meera','issued');
4  insert into book values(2264,'lion','ww','21-mar-2004','Madhu','issued be issued');
5  insert into book values(2275,'miracle','qw','23-mar-2006','Maneesh','cannot be issued');
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Trigger Code

```
1  CREATE OR REPLACE TRIGGER search1
2  before insert ON book
3  FOR EACH ROW
4  declare
5  temp date;
6  BEGIN
7  select sysdate into temp from dual;
8  IF inserting THEN
9    IF :new.publishedDate < add_months(temp,-180) THEN
10     :new.status:='cannot be issued';
11   end IF;
12 end IF;
13 end;
14
```

```
Trigger created.
```

OUTPUT

```
1  select * from book;
2
```

ACCESSION_NO	TITTLE	PUBLISHER	PUBLISHEDDATE	AUTHOR	STATUS
2231	hapy	cd	21-MAR-09	Maria	issued
2242	sad	cp	19-JUN-10	Meenu	issued
2253	star	mv	11-MAR-16	Meera	issued
2264	lion	ww	21-MAR-04	Madhu	cannot be issued
2275	miracle	qw	23-MAR-06	Maneesh	cannot be issued

Download CSV

5 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

CODE

Create table

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

```
Table created.
```

Trigger

```
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.
```

```
1  insert into inventory values(111,'pencil',110,150);
2  insert into inventory values(112,'pen',70,100);
3  insert into inventory values(113,'marker',200,170);
4  insert into inventory values(114,'notbook',500,300);
5
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

OUTPUT

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

PDTID	PDTNAME	QTY	REORDER_LEVEL
111	pencil	110	150
112	pen	70	100
113	marker	200	0
114	notbook	500	0

Download CSV

4 rows selected.