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:-

SQL Worksheet

```
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:=10;
   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 b is :'||(SQRT(b)));
   END IF;
END;
```

OUTPUT:-

SQL Worksheet

```
1  DECLARE
2      a INTEGER:=10;
3      b INTEGER:=7;
4      temp INTEGER:=0;
5      c INTEGER;
6      cube INTEGER;
7  BEGIN
8      IF a > b THEN
9          temp:=a;
10         a:=b;
11         b:=temp;
12         DBMS_OUTPUT.PUT_LINE('After swapping the a value is '||a ||' and b value is '||b);
13         IF MOD(b,2) !=0 THEN
14             cube:=a * a * a;
15             DBMS_OUTPUT.PUT_LINE('Cube is :'||cube);
16         ELSE
17             DBMS_OUTPUT.PUT_LINE('first number is even');
```

```
Statement processed.
After swapping the a value is 7 and b value is 10
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:-

SQL Worksheet

```
1  DECLARE
2      a NUMBER:=0;
3      b NUMBER:=1;
4      c NUMBER;
5  BEGIN
6      DBMS_OUTPUT.PUT(a||'   '||B||'   ');
7      FOR I IN 3..10 LOOP
8      c:=a+b;
9      DBMS_OUTPUT.PUT(c||'   ');
10     a:=b;
11     b:=c;
12     END LOOP;
13 DBMS_OUTPUT.PUT_LINE('   ');
14 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:-

```
create table employee(emp_no int,emp_name varchar(30),emp_post
varchar(30),emp_salary decimal(20,4));
```

```
Table created.
```

```
1  insert into employee values(101,'Sanjay','MD',25000);
2  insert into employee values(102,'Dhyan','HR',20000);
3  insert into employee values(103,'Sangeetha','Accountant',15000);
4  insert into employee values(104,'Anoop','Clerk',10000);
5  insert into employee values(105,'Sarah','Peon',5000);
6
```

```
1 row(s) inserted.

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:-

```
Statement processed.
No more increment
Employee num: 104
Employee name: Anoop
Employee post: Clerk
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:-

```
create table class(cls_id int,cls_name varchar(30),cls_std int);
```

```
Table created.
```

```
insert into class values(301,'mca',50);
insert into class values(302,'mca',60);
insert into class values(303,'bca',50);
insert into class values(304,'bca',69);
insert into class values(305,'msc',52);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
CREATE OR REPLACE FUNCTION total_std
    RETURN NUMBER IS
    total NUMBER(5):=0;
    BEGIN
        SELECT sum(cls_std) INTO total FROM class 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:110
```

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.

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

```
Table created.
```

```
insert into emp values(101,'arun',50000,'salesman');
insert into emp values(102,'appu',6500,'manager');
insert into emp values(103,'ammu',7500,'clerk');
insert into emp values(104,'anitha',7500,'analyst');
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

Procedure:

```
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;
```

```
Procedure created.
```

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

```
Statement processed.
Name: anitha
employee number: 104
salary: 8250
department: analyst
```

7. Create a **cursor** to modify the salary of 'president' belonging to all departments by 50%

PL/SQL Code:-

```
create table Employee(emp_id int,emp_name varchar(30),emp_post varchar(20),emp_salary int,emp_dept varchar(20))
```

```
Table created.
```

```
insert into Employee values(100,'Joseph','manager',56000,'sales')
```
1 row(s) inserted.

```
insert into Employee values(101,'Ravi','clerk',23000,'sales')
```
1 row(s) inserted.

```
insert into Employee values(102,'Paul','execute',48000,'HR')
```
1 row(s) inserted.

```
insert into Employee values(103,'Rani','president',50000,'HR')
```
1 row(s) inserted.

```
insert into Employee values(104,'Antony','president',48000,'marketing')
```
1 row(s) inserted.

```
insert into Employee values(105,'Rose','accountant',45000,'marketing')
```
1 row(s) inserted.

```
insert into Employee values(106,'Lovely','president',49000,'purchase')
```
1 row(s) inserted.

```
insert into Employee values(107,'Babu','supervisor',32000,'purchase')
```
1 row(s) inserted.

```
DECLARE
total_rows number(2);
emp1 Employee%rowtype;
BEGIN
UPDATE Employee SET emp_salary=emp_salary+emp_salary * 50/100 where
emp_post='president';
IF sql%notfound THEN
```

```
dbms_output.put_line(' no employee updated');
ELSIF sql%found THEN
total_rows := sql%rowcount;
dbms_output.put_line( total_rows ||' employee updated');
end if;
SELECT * into emp1 FROM Employee WHERE (emp_id=104 and emp_post='president');
END;
```

Statement processed.

3 employee updated

```
select * from Employee where emp_post='president'
```

EMP_ID	EMP_NAME	EMP_POST	EMP_SALARY	EMP_DEPT
103	Rani	president	75000	HR
104	Antony	president	72000	marketing
106	Lovely	president	73500	purchase

Download CSV

3 rows selected.

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

```
create table employ(emp_id int,emp_name varchar(20),emp_post varchar(20),emp_gender varchar(10),emp_salary int);
```

Table created.

```
insert into employ values('101','Anu','HR','F',10000);
insert into employ values('102','Sanjay','Manager','M',15000);
insert into employ values('103','Sreya','Sales','F',8000);
insert into employ values('104','Rajeev','Peon','M',5000);
```

```
1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

Code:-

```
DECLARE
   CURSOR emp1 IS
     SELECT emp_id,emp_name,emp_post,emp_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_id);
       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: 102
Employee_Name: Sanjay
Employee_post: Manager
Employee_salary: 15000
Employee_ID: 103
Employee_Name: Sreya
Employee_post: Sales
Employee_salary: 8000
```

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:-

```
create table book(acc_no int,title varchar(30),publisher varchar(30),pub_date date,author varchar(30),status varchar(60));
```

```
Table created.
```

```
create or replace trigger checkbook
before insert or update on book
for each row
Declare
    dop book.pub_date%type;
    yrs number(10);

Begin
        dop :=:new.pub_date;
        yrs := (months_between(sysdate,dop))/12;
        if (yrs > 15) then
        :new.status := 'CANNOT BE ISSUED';
        dbms_output.put_line('This Book Is 15 Years Old, Its Status Has Been Changed To "CANNOT BE ISSUED"');
        end if;
end;
```

```
Trigger created.
```

```
insert into book values(201,'The God of Small Things','H&C','19-may-1990','Arundhati Roy','present in library');
insert into book values(202,'Indian Home Rule','Manjusha Publications','23-mar-1987','Mahatma Gandhi','sent to binding');
insert into book values(203,'The Satanic Versus','Dhadha Publications','15-jun-1988','Salman Rushdie','issued');
```

```
1 row(s) inserted.
This Book Is 15 Years Old, Its Status Has Been Changed To "CANNOT BE ISSUED"

1 row(s) inserted.
This Book Is 15 Years Old, Its Status Has Been Changed To "CANNOT BE ISSUED"

1 row(s) inserted.
This Book Is 15 Years Old, Its Status Has Been Changed To "CANNOT BE ISSUED"
```

```
select * from book;
```

ACC_NO	TITLE	PUBLISHER	PUB_DATE	AUTHOR	STATUS
201	The God of Small Things	H&C	19-MAY-90	Arundhati Roy	CANNOT BE ISSUED
202	Indian Home Rule	Manjusha Publications	23-MAR-87	Mahatma Gandhi	CANNOT BE ISSUED
203	The Satanic Versus	Dhadha Publications	15-JUN-88	Salman Rushdie	CANNOT BE ISSUED

Download CSV

3 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:-

```
create table inventory(pdtid int,pdtname varchar(30),qty int,reorder_level int);
```

```
Table created.
```

```
CREATE OR REPLACE TRIGGER inven
 before insert ON inventory
 FOR EACH ROW
declare
BEGIN
 if(inserting)then
  if(:new.qty > :new.reorder_level)then
      :new.reorder_level:=0;
  end if;
 end if;
end;
```

```
Trigger created.
```

```
insert into inventory values(101,'Teddy',150,69);
insert into inventory values(102,'Doll',234,270);
insert into inventory values(103,'Car',234,270);
insert into inventory values(104,'Bike',234,270);
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
1 row(s) inserted.
```

```
select * from inventory;
```

PDTID	PDTNAME	QTY	REORDER_LEVEL
101	Teddy	150	0
102	Doll	234	270
103	Car	234	270
104	Bike	234	270

Download CSV

4 rows selected.