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.

PROGRAM 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) := 'MADAM';
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
11     IF t = s THEN
12         dbms_output.Put_line(t ||''||' is palindrome');
13     ELSE
14         dbms_output.Put_line(t ||''||' is not palindrome');
15     END IF;
16 END;
17
```

```
Statement processed.
MADAM 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.

PROGRAM CODE

```
DECLARE
    a INTEGER :=6;
    b INTEGER :=5;
    temp INTEGER :=0;
    c INTEGER;
    d INTEGER :=2;
    cube INTEGER;
BEGIN
  IF  a > b THEN
    temp :=a;
    a :=b;
    b := temp;
    DBMS_OUTPUT.PUT_LINE('After the 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 of a  is '||cube);
    ELSE
    DBMS_OUTPUT.PUT_LINE(' first number is odd '||a);
    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 :=6;
      b INTEGER :=5;
      temp INTEGER :=0;
      c INTEGER;
      d INTEGER :=2;
      cube INTEGER;
BEGIN
    IF  a > b THEN
       temp :=a;
       a :=b;
       b := temp;
       DBMS_OUTPUT.PUT_LINE('After the 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 of a  is '||cube);
       ELSE
       DBMS_OUTPUT.PUT_LINE(' first number is odd '||a);
       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;
```

```
Statement processed.
After the swapping the a value is 5 and b value is 6
first number is odd 5
```

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

PROGRAM CODE

```
DECLARE
   t1 NUMBER :=0;
   T2 NUMBER :=1;
   T3 NUMBER ;
BEGIN
   DBMS_OUTPUT.PUT_LINE(t1);
   DBMS_OUTPUT.PUT_LINE(t2);
   for i in 3 ..10 loop
   t3 :=t1 + t2;
   dbms_output.put_line(t3);
   t1 := t2;
   t2 := t3;
   END LOOP;
END;
```

OUTPUT

```
DECLARE
    t1 NUMBER :=0;
    T2 NUMBER :=1;
    T3 NUMBER ;
BEGIN
    DBMS_OUTPUT.PUT_LINE(t1);
    DBMS_OUTPUT.PUT_LINE(t2);
    for i in 3 ..10 loop
    t3 :=t1 + t2;
    dbms_output.put_line(t3);
    t1 := t2;
    t2 := t3;
    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.

PROGRAM CODE

```
create table EMPL(emp_no int primary key, emp_name varchar(30), salary int);

insert into EMPL values(101,'Anu',50000);

insert into EMPL values(102,'Dev',6500);

insert into EMPL values(103,'Ami',7500);


DECLARE
   emp1 EMPL%rowtype;
   sal EMPL.salary%type;
BEGIN
   SELECT salary INTO sal FROM EMPL WHERE emp_no = 102;
   IF sal <= 7500 THEN
      UPDATE EMPL SET salary = salary+salary* 15/100 WHERE emp_no = 102;
   ELSE
      DBMS_OUTPUT.PUT_LINE('No increment');
   END IF;
   SELECT * into emp1 FROM EMPL WHERE emp_no = 102;
   DBMS_OUTPUT.PUT_LINE('Employee number : '||emp1.emp_no);
   DBMS_OUTPUT.PUT_LINE('Name : '||emp1.emp_name);
   DBMS_OUTPUT.PUT_LINE('Salary : '||emp1.salary);
END;
```

OUTPUT

```
create table employee(emp_no int,emp_name varchar(20),emp_post
varchar(20),emp_salary decimal(10,2))
```

Table created.

```
1  insert into employee values(101,'Devu','MD',25000);
```

1 row(s) inserted.

```
1  insert into employee values(102,'Anu','HR',20000);
```

1 row(s) inserted.

```
1  insert into employee values(103,'Rina','Accountant',15000);
```

1 row(s) inserted.

```
1  insert into employee values(104,'Teena','Clerk',10000);
```

1 row(s) inserted.

```
1  insert into employee values(105,'Appu','Peon',5000);
```

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;
```

```
Statement processed.
No more increment
Employee num: 104
Employee name: Teena
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);

PROGRAM CODE

```
create table class(cls_id int,cls_name varchar(20),cls_std int);
insert into class values(106,'mca',120);
insert into class values(107,'mca',60);
insert into class values(108,'bca',59);
insert into class values(109,'bca',62);
insert into class values(110,'msc',62);

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

OUTPUT

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

```
Table created.
```

```
insert into class values(106,'mca',60);
```

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

```
insert into class values(107,'mca',60);
```

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

```
insert into class values(108,'bca',57);
```

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

```
insert into class values(109,'bca',59);
```

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

```
insert into class values(110,'msc',62);
```

```
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:180
```

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.

PROGRAM CODE

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

insert into emp values(101,'Sree',50000,'salesman');

insert into emp values(103,'Ami',70000,'manager');

insert into emp values(102,'Anu',64000,'clerk');

insert into emp values(104,'Dev',68000,'analyst');


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;

DECLARE
BEGIN
  increSalary();
END;
```

OUTPUT

SQL Worksheet

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

Table created.
```

SQL Worksheet

```
1  insert into emp values(101,'Sree',50000,'Salesman');

1 row(s) inserted.
```

SQL Worksheet

```
1  insert into emp values(102,'Ami',70000,'manager');

1 row(s) inserted.
```

SQL Worksheet

```
1  insert into emp values(103,'Anu',64000,'clerk')

1 row(s) inserted.
```

SQL Worksheet

```
1  insert into emp values(104,'Dev',68000,'analyst');

1 row(s) inserted.
```

SQL Worksheet

```
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.
```

SQL Worksheet

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

```
Statement processed.
Name: Dev
employee number: 104
salary: 74800
department: analyst
```

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

PROGRAM CODE

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

insert into empy values(201,'Jack',50000,'sales','president');

insert into empy values(202,'Eva',6500,'Ac','president');

insert into empy values(203,'Oliva',7500,'HR','manager');

insert into empy values(204,'Giri',7500,'Ac','snr grade');

insert into empy values(205,'Rena',7500,'HR','president');


DECLARE
   total_rows number(2);
   emp1 EMPY%rowtype;
BEGIN
 UPDATE empy 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;
```

OUTPUT

SQL Worksheet

```
1  DECLARE
2      total_rows number(2);
3      emp1 EMPY%rowtype;
4  BEGIN
5
6   UPDATE empy 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;
```

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

SQL Worksheet

```
1  select * from empy;
```

EMP_NO	EMP_NAME	SALARY	EMP_DPT	DSGT
202	Eva	9750	Ac	president
201	Jack	75000	sales	president
203	Oliva	7500	HR	manager
204	Giri	7500	HR	manager
205	Rena	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.

PROGRAM CODE

```
create table emplo(emp_no varchar(20),emp_name varchar(20),salary int,emp_dpt
varchar(20),gender varchar(10));
insert into emplo values('301','Sree',50000,'sales','male');
insert into emplo values('302','Achu',6500,'Ac','female');
insert into emplo values('303','Tiffi',7500,'HR','female');
insert into emplo values('304','Seena',7500,'Ac','male');
insert into emplo values('305','Aliya',7500,'HR','female');

DECLARE
 CURSOR emp1 is SELECT * FROM emplo 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 information: '||' '||emp2.emp_no || ' ' || emp2.emp_name || ' ' ||
emp2.salary|| ' '||emp2.emp_dpt||' '||emp2.gender);
 end loop;
 dbms_output.put_line('Totel number of rows :'||emp1%rowcount);
close emp1;
end;
```

OUTPUT

```
Statement processed.
employee information:  304 Seena 7500 Ac male
employee information:  301 Sree 50000 sales male
Totel number of rows :2
```

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.

PROGRAM CODE

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

insert into book values( 1100,'spiderman','kobo','21-jan-2009','Dean','issued');
insert into book values( 2200,'harrypotter','tolino','30-mar-2010','John','present in the library');
insert into book values( 3300,'junglebook','scribd','21-june-2011','Charles','sent for binding');
insert into book values( 4400,'cyndrella','blio','01-sep-2016','Oliva','issued');
insert into book values( 5500,'the story','hive','21-jan-2004','Kate','can not be issued');
insert into book values( 6600,magicone','fnac','21-jan-2006','Harry',' issued');

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;


select * from book;
```

OUTPUT

SQL Worksheet

```
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;
```

```
Trigger created.
```

SQL Worksheet

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

ACCESSION_NO	TITLE	PUBLISHER	PUBLISHEDDATE	AUTHOR	STATUS
1100	spiderman	kobo	21-JAN-09	Dean	issued
2200	harrypotter	tolino	30-MAR-10	John	present in the library
3300	junglebook	scribd	21-JUN-11	Charles	sent for binding
4400	cyndrella	blio	01-SEP-16	Oliva	issued
5500	the story	hive	21-JAN-04	Kate	can not 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.

PROGRAM CODE

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

insert into inventory values(100,'pen',100,150);

insert into inventory values(102,'pencil',50,100);

insert into inventory values(103,'eraser',200,150);

insert into inventory values(104,'paper',500,250);


CREATE OR REPLACE TRIGGER checking

 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;


select * from inventory;
```

OUTPUT

SQL Worksheet

```
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;
```

```
Trigger created.
```

SQL Worksheet

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

PDTID	PDTNAME	QTY	REORDER_LEVEL
104	paper	500	250
103	eraser	200	150
100	pen	100	150
102	pencil	50	100

Download CSV

4 rows selected.