ADVANCED DBMS LAB

CYCLE -2

PL/SQL Programs (Trigger, Cursor, Stored Procedures and Functions)

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) :='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
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.
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.

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:

SQL Worksheet

```
1  DECLARE
2        a INTEGER :=6;
3        b INTEGER :=5;
4        temp INTEGER :=0;
5        c INTEGER;
6        d INTEGER :=2;
7        cube INTEGER;
8  BEGIN
9      IF  a > b THEN
10        temp :=a;
11        a :=b;
12        b := temp;
13        DBMS_OUTPUT.PUT_LINE('After the swapping the a value is '||a ||' and b value is '||b);
14        IF MOD(b,2) !=0 THEN
15         cube :=a * a * a;
16        DBMS_OUTPUT.PUT_LINE('cube of a  is '||cube);
17        ELSE
18        DBMS_OUTPUT.PUT_LINE(' first number is odd '||a);
19        END IF;
20
21     ELSIF a < b THEN
```

```
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:

SQL Worksheet

```
1  DECLARE
2      t1 NUMBER :=0;
3      t2 NUMBER :=1;
4      t3 NUMBER ;
5  BEGIN
6        dbms_output.put_line(t1);
7        dbms_output.put_line(t2);
8        for i in 3 ..10 loop
9        t3 :=t1 + t2;
10       dbms_output.put_line(t3);
11       t1 := t2;
12       t2 := t3;
13       END LOOP;
14
15
16 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 EMP(emp_no int primary key,emp_name varchar(20),emp_post varchar(20),emp_salary decimal(10,2));

insert into EMP values(101,'arun','MD',50000);

insert into EMP values(102,'ajay','HR',25000);

insert into EMP values(103,'raman','Accountant',15000);

insert into EMP values(104,'ravi','clerk',7000);

insert into EMP values(105,'shivan','peon',5000);



DECLARE

   emno EMP.emp_no%type;

   salary EMP.emp_salary%type;

   emp_rec EMP%rowtype;

BEGIN

   emno :=102;

   SELECT emp_salary INTO salary FROM EMP WHERE emp_no = emno;

   IF  salary <= 7500 THEN

     UPDATE EMP 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 EMP WHERE emp_no = emno;
    DBMS_OUTPUT.PUT_LINE('NAME: '||emp_rec.emp_name);
    DBMS_OUTPUT.PUT_LINE('Employee Number: '||emp_rec.emp_no);
    DBMS_OUTPUT.PUT_LINE('SALARY: '||emp_rec.emp_salary);
    DBMS_OUTPUT.PUT_LINE('POST: '||emp_rec.emp_post);
END;
```

OUTPUT:

SQL Worksheet

```
1  create table EMP(emp_no int primary key,emp_name varchar(20),emp_post varchar(20),emp_salary decimal(10,2));
2  insert into EMP values(101,'arun','MD',50000);
3  insert into EMP values(102,'ajay','HR',25000);
4  insert into EMP values(103,'raman','Accountant',15000);
5  insert into EMP values(104,'ravi','clerk',7000);
6  insert into EMP values(105,'shivan','peon',5000);
7
8  DECLARE
9      emno EMP.emp_no%type;
10     salary EMP.emp_salary%type;
11     emp_rec EMP%rowtype;
12 BEGIN
13     emno :=102;
14     SELECT emp_salary INTO salary FROM EMP WHERE emp_no = emno;
15     IF  salary <= 7500 THEN
16        UPDATE EMP SET emp_salary=emp_salary * 15/100 WHERE emp_no = emno;
17     ELSE
18        DBMS_OUTPUT.PUT_LINE ('NO MORE INCREMENT');
19     END IF;
20     SELECT * INTO emp_rec FROM EMP WHERE emp_no = emno;
21     DBMS_OUTPUT.PUT_LINE('NAME: '||emp_rec.emp_name);
22     DBMS_OUTPUT.PUT_LINE('Employee Number: '||emp_rec.emp_no);
23     DBMS_OUTPUT.PUT_LINE('SALARY: '||emp_rec.emp_salary);
24     DBMS_OUTPUT.PUT_LINE('POST: '||emp_rec.emp_post);
25 END;
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

Statement processed.
NO MORE INCREMENT
NAME: ajay
Employee Number: 102
SALARY: 25000
POST: HR
```

---

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(201,'mca',120);

insert into class values(202,'mca',60);

insert into class values(203,'bca',59);

insert into class values(204,'bca',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;
insert into class values(205,'msc',62);


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

OUTPUT:

SQL Worksheet

```
1  create table class(cls_id int,cls_name varchar(20),cls_std int);
2  insert into class values(201,'mca',60);
3  insert into class values(202,'mca',80);
4  insert into class values(203,'bca',38);
5  insert into class values(202,'bca',50);
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

SQL Worksheet

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

```
Function created.
```

SQL Worksheet

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

```
Statement processed.
Total students in MCA department is:140
```

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,'Kiran',50000,'salesman');

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

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

insert into emp values(104,'arun',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));
2  insert into emp values(201,'Reshana',50000,'salesman');
3  insert into emp values(202,'Ramshi',70000,'manager');
4  insert into emp values(203,'Zoya',64000,'clerk');
5  insert into emp values(204,'Zidhan',68000,'analyst');
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

SQL Worksheet

```
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  ;
16   ELSE
17      DBMS_OUTPUT.PUT_LINE ('NO INCREMENT');
18    END IF;
19    SELECT * into emp1 FROM emp WHERE emp_no = 204;
20    DBMS_OUTPUT.PUT_LINE ('Name: '||emp1.emp_name);
21    DBMS_OUTPUT.PUT_LINE ('employee number: '||emp1.emp_no);
22    DBMS_OUTPUT.PUT_LINE ('salary: '|| emp1.salary);
23    DBMS_OUTPUT.PUT_LINE ('department: '|| emp1.emp_dpt);
24 END;
```

```
Procedure created.
```

SQL Worksheet

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

```
Statement processed.
Name: Zidhan
employee number: 204
salary: 74800
department: analyst
```

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

PROGRAM CODE:

```
CREATE TABLE employee(designation varchar(20),department varchar(20),salary number(7));

INSERT INTO employee values('president','Administration',50000);

INSERT INTO employee values('president','Marketing',45000);

INSERT INTO employee values('Manager','Administration',35000);

INSERT INTO employee values('Manager','Marketing',48000);

DECLARE

  total_rows number(2);
```

```
  totalnorows number;
BEGIN
  SELECT count(*) INTO totalnorows FROM employee;
  dbms_output.put_line('total no of rows:'||totalnorows);
  UPDATE employee SET salary = salary+ salary*0.5 WHERE designation='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 salaries updated');
  END IF;
  END;
```

OUTPUT

SQL Worksheet

```
1  CREATE TABLE employee(designation varchar(20),department varchar(20),salary number(7));
2  INSERT INTO employee values('president','Administration',50000);
3  INSERT INTO employee values('president','Marketing',45000);
4  INSERT INTO employee values('Manager','Administration',35000);
5  INSERT INTO employee values('Manager','Marketing',48000);
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

SQL Worksheet

```
1  DECLARE
2     total_rows number(2);
3     totalnorows number;
4  BEGIN
5     SELECT count(*) INTO totalnorows FROM employee;
6     dbms_output.put_line('total no of rows:'||totalnorows);
7     UPDATE employee SET salary = salary+ salary*0.5 WHERE designation='president';
8     IF sql%notfound THEN
9     dbms_output.put_line('no employee salary updated');
10    ELSIF sql%found THEN total_rows := sql%rowcount;
11    dbms_output.put_line(total_rows || 'employee salaries updated');
12    END IF;
13    END;
14
```

```
Statement processed.
total no of rows:4
2employee salaries updated
```

SQL Worksheet

```
1  select * from employee;
2
```

DESIGNATION	DEPARTMENT	SALARY
president	Administration	75000
president	Marketing	67500
Manager	Administration	35000
Manager	Marketing	48000

Download CSV

4 rows selected.

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

PROGRAM CODE:

```
create table emp(emp_no varchar(20),emp_name varchar(20),salary int,emp_dpt varchar(20),gender varchar(10));

insert into emp values('101','arun',50000,'sales','male');

insert into emp values('102','sandeep',6500,'Ac','male');

insert into emp values('103','ammu',7500,'HR','female');

insert into emp values('104','snitha',7500,'Ac','female');

insert into emp values('105','anitha.c',7500,'HR','female');

DECLARE

 CURSOR emp1 is SELECT * FROM emp 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

SQL Worksheet

```
1  create table emp(emp_no varchar(20),emp_name varchar(20),salary int,emp_dpt varchar(20),gender varchar(10));
2  insert into emp values('101','arun',50000,'sales','male');
3  insert into emp values('102','sandeep',6500,'Ac','male');
4  insert into emp values('103','ammu',7500,'HR','female');
5  insert into emp values('104','snitha',7500,'Ac','female');
6  insert into emp values('105','anitha.c',7500,'HR','female');
7
```

```
Table created.
```

SQL Worksheet

```
1  DECLARE
2   CURSOR emp1 is SELECT * FROM emp WHERE emp_name like ('s%');
3   emp2 emp1%rowtype;
4  BEGIN
5   open emp1;
6   loop
7    fetch emp1 into emp2;
8    exit when emp1%notfound;
9    dbms_output.put_line('employee information: '||' '||emp2.emp_no || ' ' || emp2.emp_name || ' ' || emp2.salary|| ' '||emp2.emp_dpt||' '||emp2.gender);
10  end loop;
11  dbms_output.put_line('Totel number of rows :'||emp1%rowcount);
12 close emp1;
13 end;
```

```
Statement processed.
employee information:  102 sandeep 6500 Ac male
employee information:  104 snitha 7500 Ac female
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 beissued.Write a **trigger** which sets the status of a book to "cannot be issued", if it ispublished 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));

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;

insert into book values( 2511,'abcd','cp','21-jan-2009','john','issued');

insert into book values( 2512,'efhj','cp','30-mar-2010','malik','present in the library');

insert into book values( 2513,'hijk','cp','21-june-2011','sonu','sent for binding');

insert into book values( 2514,'lmno','cp','01-sep-2016','johns','issued');

insert into book values( 2515,'pqrst','cp','21-jan-2004','joppy','can not be issued');

insert into book values( 2516,'uvwx','cp','21-jan-2006','juosoop',' issued');

SELECT * FROM book;
```

OUTPUT

SQL Worksheet

Clear

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

Table created.

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;
14
```

Trigger created.

SQL Worksheet

```
1  insert into book values( 2511,'abcd','cp','21-jan-2009','john','issued');
2  insert into book values( 2512,'efhj','cp','30-mar-2010','malik','present in the library');
3  insert into book values( 2513,'hijk','cp','21-june-2011','sonu','sent for binding');
4  insert into book values( 2514,'lmno','cp','01-sep-2016','johns','issued');
5  insert into book values( 2515,'pqrst','cp','21-jan-2004','joppy','can not be issued');
6  insert into book values( 2516,'uvwx','cp','21-jan-2006','juosoop',' 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.
```

SQL Worksheet

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

ACCESSION_NO	TITLE	PUBLISHER	PUBLISHEDDATE	AUTHOR	STATUS
2511	abcd	cp	21-JAN-09	john	issued
2512	efhj	cp	30-MAR-10	malik	present in the library
2513	hijk	cp	21-JUN-11	sonu	sent for binding
2514	lmno	cp	01-SEP-16	johns	issued
2515	pqrst	cp	21-JAN-04	joppy	cannot be issued
2516	uvwx	cp	21-JAN-06	juosoop	cannot be issued
2511	abcd	cp	21-JAN-09	john	issued
2512	efhj	cp	30-MAR-10	malik	present in the library
2513	hijk	cp	21-JUN-11	sonu	sent for binding
2514	lmno	cp	01-SEP-16	johns	issued
2515	pqrst	cp	21-JAN-04	joppy	cannot be issued

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);
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;
insert into inventory values(101,'pencil',100,150);
insert into inventory values(112,'tap',50,100);
insert into inventory values(121,'marker',200,150);
insert into inventory values(151,'notbook',500,250);
select * from inventory;
```

OUTPUT

SQL Worksheet

```
1  create table inventory(pdtid number primary key, pdtname varchar(10), qty int,reorder_level number);
2
```

```
Table created.
```

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;
12
```

```
Trigger created.
```

SQL Worksheet

```
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.
```

SQL Worksheet

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

PDTID	PDTNAME	QTY	REORDER_LEVEL
101	pencil	100	150
112	tap	50	100
121	marker	200	0
151	notbook	500	0

Download CSV

4 rows selected.