LAB CYCLE-2

KIRAN KARTHIKEYAN

09 MCA B

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

```
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

```
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.

```
DECLARE
 a INTEGER:=12;
 b INTEGER:=9;
 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;
```

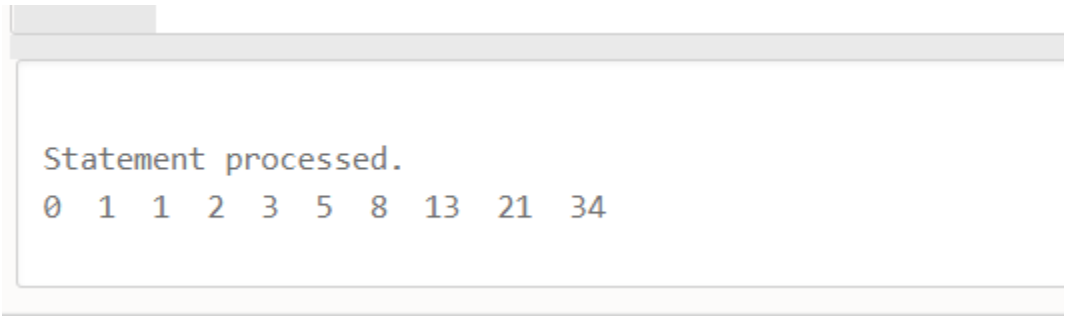
OUTPUT

```
Statement processed.
After swapping the a value is 27 and b value is 77
Cube is :19683
```

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

```
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



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.

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

Table created.

```
insert into employee values(103,'Rahul','MD',25000);
```

1 row(s) inserted.

```
insert into employee values(105,'Ravi','HR',20000);
```

1 row(s) inserted.

```
insert into employee values(107,'Rani','Accountant',15000);
```

1 row(s) inserted.

```
insert into employee values(109,'Rema','Clerk',10000);
```

1 row(s) inserted.

```
insert into employee values(201,'Ramu','Peon',5000);
```

1 row(s) inserted.

```
Declare
 emno employee.emp_no%type;
 salary employee.emp_salary%type;
 emp_rec employee%rowtype;
begin
 emno:=109;
 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

```
No more increment
Employee num: 109
Employee name: Rema
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);

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

Table created.

```
insert into class values(201,'mca',60);
```

1 row(s) inserted.

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

1 row(s) inserted.

```
insert into class values(203,'bca',57);
```

1 row(s) inserted.

```
insert into class values(204,'bca',59);
```

1 row(s) inserted.

```
insert into class values(205,'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);
```

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.

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

```
Table created.
```

```
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');
```

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

```
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;
```

OUTPUT

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

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

```
create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20),dsgt varchar(20));
insert into emp values(101,'Kiran',60000,'sales','president');
insert into emp values(102,'Karthik',7500,'Ac','president');
insert into emp values(103,'Appu',8500,'HR','manager');
insert into emp values(104,'Sarath',9500,'Ac','snr grade');
insert into emp values(105,'Aleena.c',8500,'HR','president');

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;
```

OUTPUT

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

EMP_NO	EMP_NAME	SALARY	EMP_DPT	DSGT
101	Kiran	90000	sales	president
102	Karthik	11250	Ac	president
103	Appu	8500	HR	manager
104	Sarath	9500	Ac	snr grade
105	Aleena.c	12750	HR	president

Download CSV

5 rows selected.

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

```
create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20))
insert into emp values(101,'Arun',75000,'Manager')
insert into emp values(102,'Sarath',65000,'Sales')
insert into emp values(103,'Kiran',55000,'Clerk')
insert into emp values(104,'Joel',68000,'Analyst')

DECLARE
   CURSOR emp1 IS
    SELECT emp_no,emp_name,emp_dpt,salary  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_ID: ' ||emp2.emp_no);
      dbms_output.Put_line('Employee_Name: ' ||emp2.emp_name);
      dbms_output.Put_line('Employee_salary: ' ||emp2.salary);
      dbms_output.Put_line('Employee_post: '||emp2.emp_dpt);
```

```
    END LOOP;

    CLOSE emp1;

END;
```

OUTPUT

```
Statement processed.
Employee_ID: 102
Employee_Name: Sarath
Employee_salary: 65000
Employee_post: Sales
```

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.

```
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;
```

```
Trigger created.
```

```
insert into book values( 2000,'Great','ACD','21-jan-2009','John','issued');
insert into book values( 2001,'Kingdom','CP','30-mar-2010','Joel','present in the library');
insert into book values( 2002,'Gods on','CP','21-june-2011','Kishor','sent for binding');
insert into book values( 2003,'lOve life','HP','01-sep-2016','Arya','issued');
insert into book values( 2004,'Life','CP','21-jan-2004','Steller','can not be issued');
insert into book values( 2005,'Self ','HB','21-jan-2006','Ram',' issued');
select * from book;
```

OUTPUT

ACCESSION_NO	TITLE	PUBLISHER	PUBLISHEDDATE	AUTHOR	STATUS
2000	Great	ACD	21-JAN-09	John	issued
2001	Kingdom	CP	30-MAR-10	Joel	present in the library
2002	Gods on	CP	21-JUN-11	Kishor	sent for binding
2003	lOve life	HP	01-SEP-16	Arya	issued
2004	Life	CP	21-JAN-04	Steller	cannot be issued
2005	Self	HB	21-JAN-06	Ram	cannot be issued

Download CSV

6 rows selected.

9) 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.

```
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

PDTID	PDTNAME	QTY	REORDER_LEVEL
101	PEN	100	150
112	PENCIL	50	100
121	FILE	200	0
151	NOTEBOOK	500	0

Download CSV

4 rows selected.