ADBMS 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):='abccba';
    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;
```

```
Statement processed.

abccba 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:=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;
```

```
Statement processed.

After swapping the a value is 9 and b value is 12

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;
```

```
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(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;
```

```
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);

PL/SQL CODE:

```
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);
END;

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:

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

```
Table created.
```

```
insert into emplo values(101,'Jeevn',50000,'salesman');
```

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

```
insert into emplo values(102,'Rohit',6500,'manager');
```

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

```
insert into emplo values(103,'Jose',7500,'clerk');
```

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

```
insert into emplo values(104,'Arun',7500,'analyst');
```

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

```
CREATE OR REPLACE PROCEDURE increSalary
IS
emp1 emplo%rowtype;
sal emplo.salary%type;
dpt emplo.emp_dpt%type;

BEGIN
SELECT salary,emp_dpt INTO sal,dpt FROM emplo WHERE emp_no=103;
   IF dpt ='clerk' THEN
     UPDATE emplo SET salary = salary+salary* 5/100;
   ELSIF dpt = 'salesman' THEN
     UPDATE emplo SET salary = salary+salary* 7/100;
   ELSIF dpt = 'analyst' THEN
     UPDATE emplo SET salary = salary+salary* 10/100;
  ELSIF dpt = 'manager' THEN
     UPDATE emplo SET salary = salary+salary* 20/100;
  ELSE
     DBMS_OUTPUT.PUT_LINE ('NO INCREMENT');
   END IF;
   SELECT * into emp1 FROM emplo WHERE emp_no = 103;
   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: Jose
employee number: 103
salary: 7875
department: clerk
```

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(20),emp_post
varchar(20),emp_salary int,emp_dept varchar(10))
```
```
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.

PL/SQL CODE:

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

```
Table created.
```

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

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

```
insert into employ values(105,'Sona','HR','F',20000);
```

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

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

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

```
insert into employ values(109,'Simi','Clerk','F',10000);
1 row(s) inserted.
```

```
insert into employ values(201,'Sajeev','Peon','M',5000);
1 row(s) inserted.
```

```
DECLARE
    CURSOR emp1 IS
      SELECT emp_no,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_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.emp_salary);
    END LOOP;
    CLOSE emp1;
END;
```

```
Statement processed.

Employee_ID: 105
Employee_Name: Sona
Employee_post: HR
Employee_salary: 20000

Employee_ID: 109
Employee_Name: Simi
Employee_post: Clerk
Employee_salary: 10000

Employee_ID: 201
Employee_Name: Sajeev
Employee_post: Peon
Employee_salary: 5000
```

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(20),publisher varchar(20),pub_date date,author
varchar(20),status varchar(50));
```

```
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(239,'Dance Dance Dance','Pothi','11-mar-1987','Haruki
Murakami','present in library');
```

```
1 row(s) inserted.

This Book Is 15 Years Old, Its Status Has Been Changed To "CANNOT BE ISSUED"
```

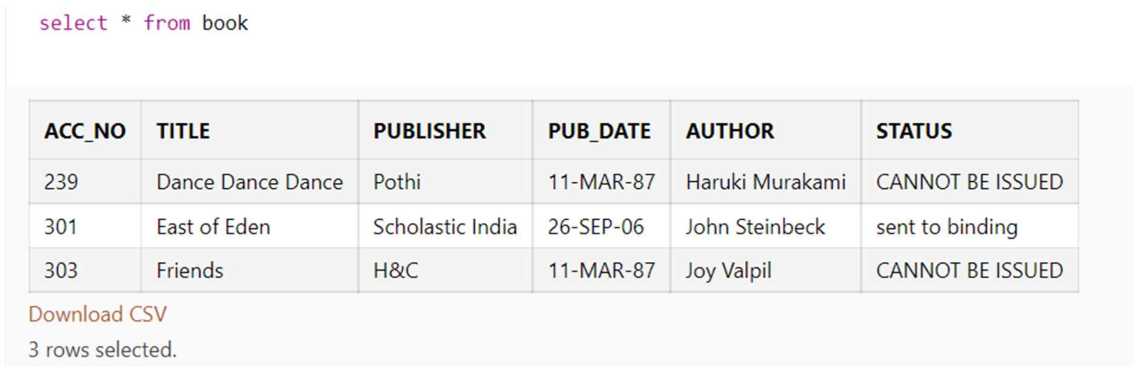
```
insert into book values(301,'East of Eden','Scholastic India','26-SEP-06','John
Steinbeck','sent to binding');
```

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

```
insert into book values(303,'Friends','H&C','11-mar-1987','Joy Valpil','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
239	Dance Dance Dance	Pothi	11-MAR-87	Haruki Murakami	CANNOT BE ISSUED
301	East of Eden	Scholastic India	26-SEP-06	John Steinbeck	sent to binding
303	Friends	H&C	11-MAR-87	Joy Valpil	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(20),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.
```

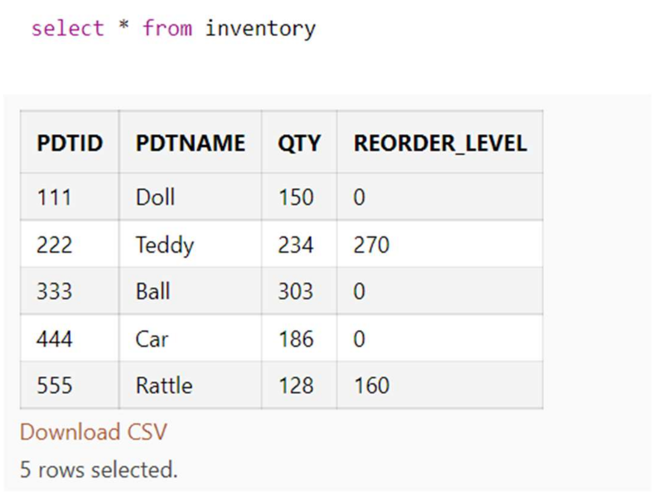
```
insert into inventory values(111,'Doll',150,69);
```
```
1 row(s) inserted.
```

```
insert into inventory values(222,'Teddy',234,270);
```
```
1 row(s) inserted.
```

```
insert into inventory values(333,'Ball',303,160);
```
```
1 row(s) inserted.
```

```
insert into inventory values(444,'Car',186,160);
```
```
1 row(s) inserted.
```

```
insert into inventory values(555,'Rattle',128,160);
```
```
1 row(s) inserted.
```



```sql
select * from inventory
```

PDTID	PDTNAME	QTY	REORDER_LEVEL
111	Doll	150	0
222	Teddy	234	270
333	Ball	303	0
444	Car	186	0
555	Rattle	128	160

Download CSV

5 rows selected.