Dbms -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) := '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;
```

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:=12;
b INTEGER:=9;
temp INTEGER:=0;
c INTEGER;
cube INTEGER;
BEGIN
IF a > b THEN
temp:=a;
a:=b;
b:=temp;
DBMS_OUTPUT_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_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_LINE('Square root of a is :'||(SQRT(a)));
DBMS_OUTPUT_LINE('Square root of b is :'||(SQRT(b)));
END IF:
END;
```



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

PROGRAM 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;
```

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 employee(emp_no int,emp_name varchar(20),emp_post
varchar(20),emp_salary decimal(10,2));
insert into employee values(103, 'Rahul', 'MD', 25000);
insert into employee values(105, 'Ravi', 'HR', 20000);
insert into employee values(107, 'Rani', 'Accountant', 15000);
insert into employee values(109, 'Rema', 'Clerk', 10000);
insert into employee values(201, 'Ramu', 'Peon', 5000);
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

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', 60);
insert into class values(202, 'mca', 60);
insert into class values(203,'bca',57);
insert into class values(204, 'bca', 59);
insert into class values(205, '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_LINE('Total students in MCA department is:'||c);
END;
Output
```

```
create table class(cls_id int,cls_name varchar(20),cls_std int);
insert into class values(201, mca_i,0);
insert into class values(202, mca_i,0);
insert into class values(202, mca_i,0);
insert into class values(204, bca_i,77);
insert into class values(204, bca_i,79);
insert into class values(204, bca_i,79);
insert into class values(205, msc_i,20);
cmeare on reprace Function total_std
arturn number is
total number(s):-0;
end
insert into class values(205, msc_i,20);
cmeare on reprace Function total_std
arturn number is
total number(s):-0;
end
insert into class values(205, msc_i,20);
cmeare on reprace Function total_std
arturn number is
total number(s):-0;
end
insert into class values(205, msc_i,20);
cmeare on reprace Function total_std
insert into class values(201, msc_i,20);
insert into class values(202, msc_i,20);
insert into class values(20
```

```
Table created.

1 row(s) inserted.

Function created.
```

```
Function created.

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.

PROGRAM CODE

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

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_LINE ('employee number: '||emp1.emp_no);
 DBMS_OUTPUT_LINE ('salary: '|| emp1.salary);
 DBMS_OUTPUT_LINE ('department: '|| emp1.emp_dpt);
END;
DECLARE
BEGIN
 increSalary();
END;
Output
```

```
1 CREATE OR REPLACE PROCEDURE increSalary
                BEOIN

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 dpt = 'manager' THEN

UPDATE emp SET salary = salary*salary* 20/100;
                     UPDATE emp SET salary = salary+salary*
ELSE
DBMS_OUTPUT.PUT_LINE ('NO INCREMENT');
END IF;
                        END IF;
SELECT * into emp1 FROM emp laHERE emp_no = 104;
DEMS_OUTPUT.PUT_LINE ('Name: '||emp1.emp_name);
DEMS_OUTPUT.PUT_LINE ('employee number: '||emp1.emp_no);
DEMS_OUTPUT.PUT_LINE ('alary: '||emp1.salary);
DEMS_OUTPUT.PUT_LINE ('department: '|| emp1.emp_dpt);
DEMS_OUTPUT.PUT_LINE ('department: '|| emp1.emp_dpt);
                create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20)); 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',7590, 'analyst');
    27 insert into emp v
28 insert into emp v
29 insert into emp v
30 insert into emp v
31
32 DECLARE
33 BEGIN
34 increSalary();
35 END;
SQL Worksheet
                                                                                                                                                                                                                                                                                                                                                                                         1 row(s) inserted.
1 row(s) inserted.
```

```
Procedure created.

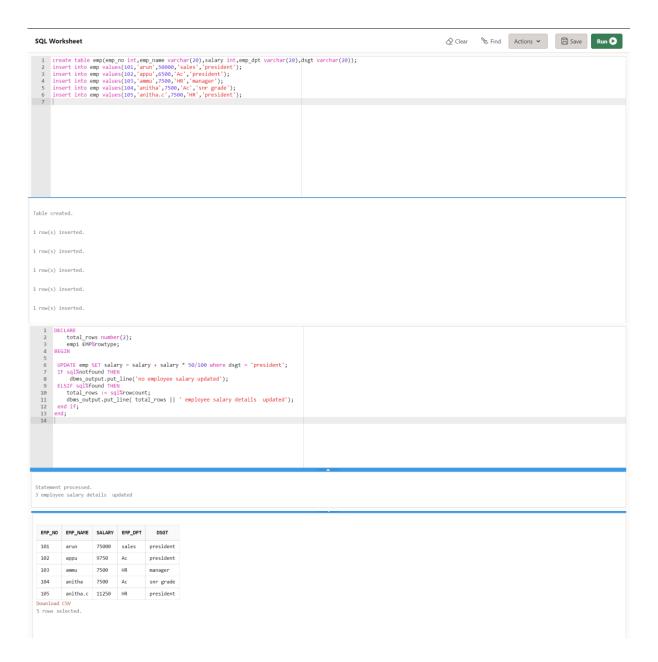
Statement processed.
Name: anitha
employee number: 104
salary: 8250
department: analyst
```

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

PROGRAM CODE

```
table
                 emp(emp_no
                                  int,emp_name
                                                   varchar(20), salary
                                                                         int,emp_dpt
create
varchar(20),dsgt varchar(20));
insert into emp values(101, 'arun', 50000, 'sales', 'president');
insert into emp values(102,'appu',6500,'Ac','president');
insert into emp values(103,'ammu',7500,'HR','manager');
insert into emp values(104,'anitha',7500,'Ac','snr grade');
insert into emp values(105, 'anitha.c', 7500, '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



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

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

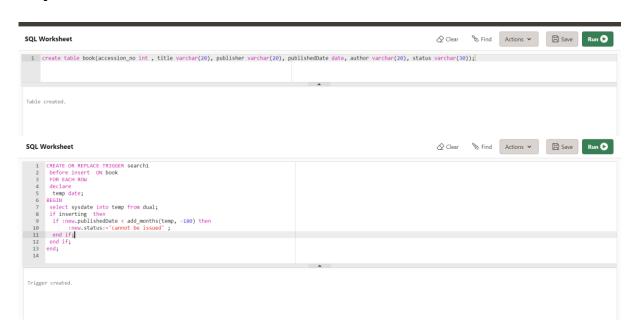
create table book(accession_no int , title varchar(20), publisher varchar(20), publishedDate date, author varchar(20), status varchar(30));

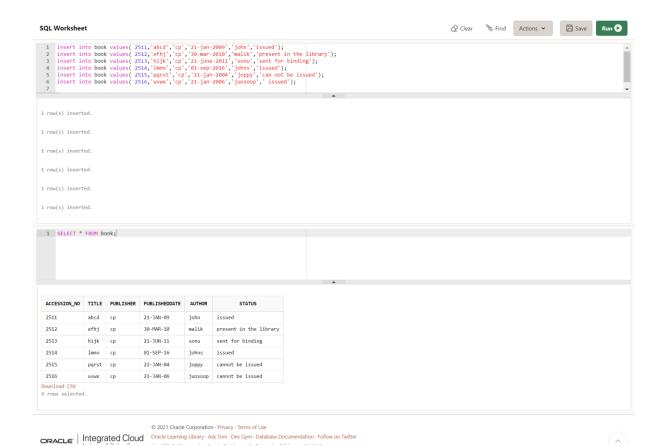
CREATE OR REPLACE TRIGGER search1

```
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





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;
```

