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
 a VARCHAR(15):='MALAYALAM';
b VARCHAR(15);  n NUMBER;
BEGIN  n:=LENGTH(a);
 FOR i IN REVERSE 1..n
LOOP
  b:=b || SUBSTR(a,I,1);
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Reversed String:'||b);
n:=INSTR(a,b);
 IF n!=1 THEN
 DBMS_OUTPUT.PUT_LINE(b ||'is not a paliandrome);
 ELSE
 DBMS_OUTPUT.PUT_LINE(b ||'is a paliandrome);
 END IF;
END;
```

OUTPUT:

```
DECLARE
 a VARCHAR(15):='MALAYALAM';
 b VARCHAR(15);
 n NUMBER;
BEGIN
 n:=LENGTH(a);
 FOR i IN REVERSE 1..n
 LOOP
  b:=b || SUBSTR(a,I,1);
 END LOOP;
 DBMS_OUTPUT.PUT_LINE('Reversed String:'||b);
 n:=INSTR(a,b);
 IF n!=1 THEN
 DBMS_OUTPUT.PUT_LINE(b ||'is not a paliandrome');
 ELSE
 DBMS_OUTPUT.PUT_LINE(b ||'is a paliandrome');
 END IF;
END;
```

```
Statement processed.
Reversed String:MALAYALAM
MALAYALAMis a paliandrome
```

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);

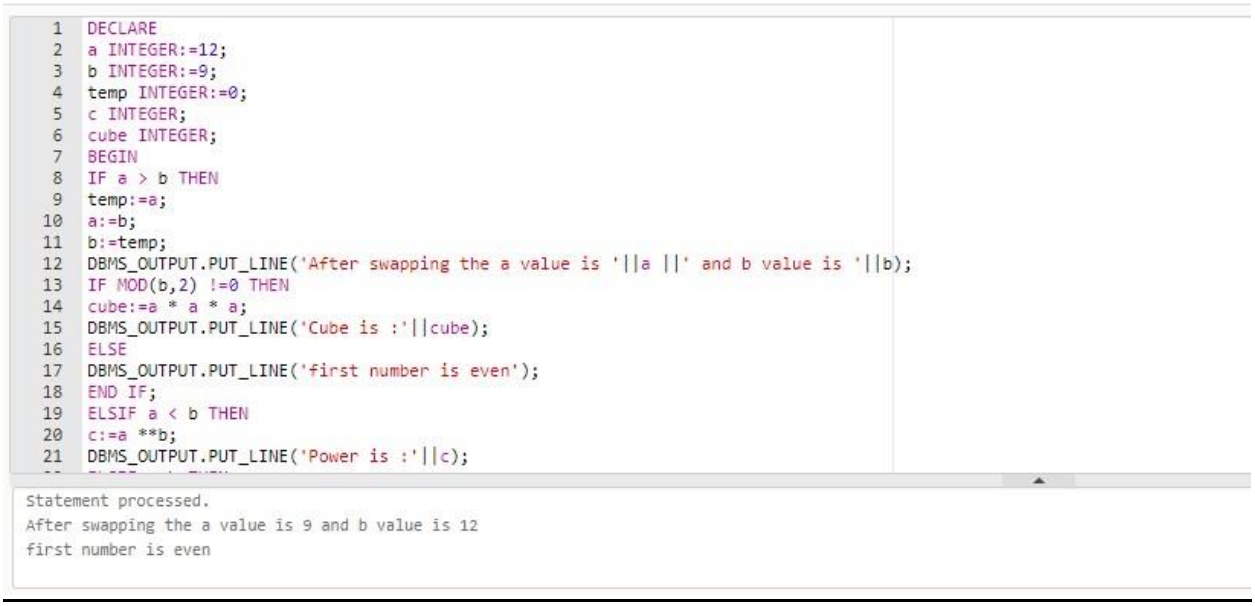
ELSIF a=b THEN

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;

OUTPUT:



```
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);
```

```
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

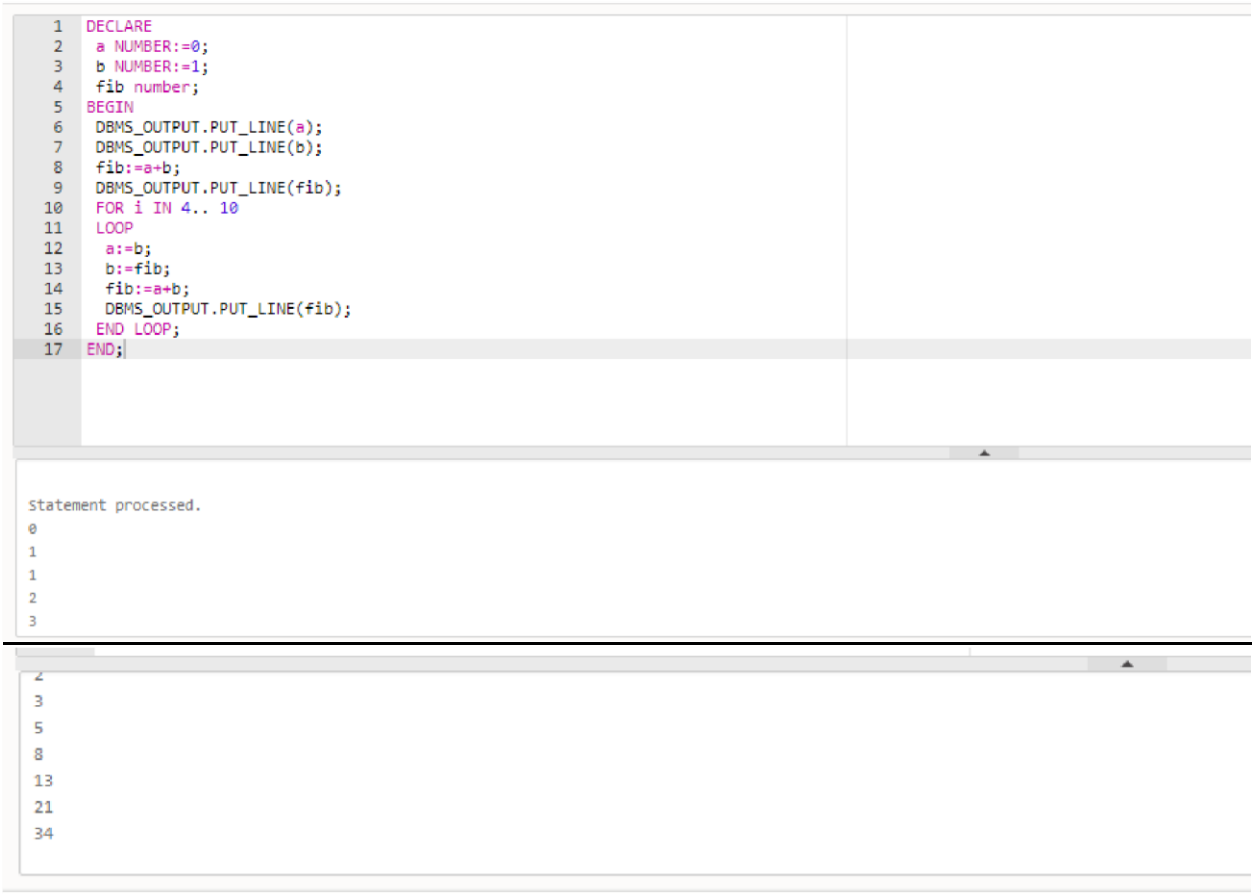
```
NUMBER:=1;
fib number;

BEGIN
 DBMS_OUTPUT.PUT_LINE(a);
 DBMS_OUTPUT.PUT_LINE(b);
 fib:=a+b;
 DBMS_OUTPUT.PUT_LINE(fib);
 FOR i IN 4.. 10
LOOP

  a:=b;
b:=fib;
fib:=a+b;

  DBMS_OUTPUT.PUT_LINE(fib);
 END LOOP;
END;
```

OUTPUT:



```
DECLARE
 a NUMBER:=0;
 b NUMBER:=1;
 fib number;
BEGIN
 DBMS_OUTPUT.PUT_LINE(a);
 DBMS_OUTPUT.PUT_LINE(b);
 fib:=a+b;
 DBMS_OUTPUT.PUT_LINE(fib);
 FOR i IN 4.. 10
 LOOP
  a:=b;
  b:=fib;
  fib:=a+b;
  DBMS_OUTPUT.PUT_LINE(fib);
 END LOOP;
END;
```

```
Statement processed.
0
1
1
2
3
```

```
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;
```

OUTPUT:

```
1  create table pmployee(emp_no int,emp_name varchar(20),emp_post
2  varchar(20),emp_salary decimal(10,2));
3  Table created.
4  insert into pmployee values(103,'Rahul','MD',25000);
5  1 row(s) inserted.
6  insert into pmployee values(105,'Ravi','HR',20000);
7  1 row(s) inserted.
8  insert into pmployee values(107,'Rani','Accountant',15000);
9  1 row(s) inserted.
10 insert into pmployee values(109,'Rema','Clerk',10000);
11 1 row(s) inserted.
12 insert into pmployee values(201,'Ramu','Peon',5000);
13 1 row(s) inserted.
14 Declare
15 emno employee.emp_no%type;
16 salary employee.emp_salary%type;
17 emp_rec employee%rowtype;
18 begin
19 emno:=109;
20 select emp_salary into salary from pmployee where emp_no=emno;
21 if salary<7500 then

No more increment
Employee num: 109
Employee name: Rema
Employee post: Clerk
```

---

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(101,'mca',60); 1 row(s)
inserted.

insert into class values(102,'mca',60); 1 row(s)
inserted.
```

```
insert into class values(103,'bca',57); 1 row(s)
inserted.

insert into class values(104,'bca',59); 1 row(s)
inserted.

insert into class values(105,'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;
```

OUTPUT:

```
Statement processed.
Total students in MCA department is:720
```

---

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 emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20));
insert into emp values(301,'Manu',50000,'salesman');
insert into emp values(302,'Mevin',6500,'manager');
insert into emp values(303,'Jesvin',7500,'clerk');
insert into emp values(304,'Viju',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 = 304;
  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 = 304;
  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:

```
1  create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20));
2  insert into emp values(301,'Manu',50000,'salesman');
3  insert into emp values(302,'Mevin',6500,'manager');
4  insert into emp values(303,'Jesvin',7500,'clerk');
5  insert into emp values(304,'Viju',7500,'analyst');insert into emp values(304,'Viju',7500,'analyst');
6
7
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```sql
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 = 304;
   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 = 304;
    DBMS_OUTPUT.PUT_LINE ('Name: '||emp1.emp_name);
    DBMS_OUTPUT.PUT_LINE ('employee number: '||emp1.emp_no);
    DBMS_OUTPUT.PUT_LINE ('salary: '|| emp1.salary);
```

```
Procedure created.
```

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

```
Statement processed.
Name: Viju
employee number: 304
salary: 8250
department: analyst
```

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

PL/SQL CODE:

```
create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20),dsgt
varchar(20));
insert into emp values(301,'Manu',50000,'sales','president');
insert into emp values(302,'Mevin',6500,'Ac','president');
insert into emp values(303,'Jesvin',7500,'HR','manager');
insert into emp values(304,'Viju',7500,'Ac','snr grade');
insert into emp values(305,'Kevin',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:

```
1  select * from emp;
2  DECLARE
3      total_rows number(2);
4      emp1 EMP%rowtype;
5  BEGIN
6
```

EMP_NO	EMP_NAME	SALARY	EMP_DPT	DSGT
301	Manu	75000	sales	president
302	Mevin	9750	Ac	president
303	Jesvin	7500	HR	manager
304	Viju	7500	Ac	snr grade
305	Kevin	11250	HR	president

Download CSV
5 rows selected.

Statement processed.
3 employee salary details updated

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

PL/SQL 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('301','Manu',50000,'sales','male');
insert into emp values('302','Sandeep',6500,'Ac','male');
insert into emp values('303','Sidharth',7500,'HR','male');
insert into emp values('304','Seenath',7500,'Ac','female');
insert into emp values('305','Mallika',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:

```
create table emp(emp_no varchar(20),emp_name varchar(20),salary int,emp_dpt varchar(20),gender varchar(10));
insert into emp values('301','Manu',50000,'sales','male');
insert into emp values('302','Sandeep',6500,'Ac','male');
insert into emp values('303','Sidharth',7500,'HR','male');
insert into emp values('304','Seenath',7500,'Ac','female');
insert into emp values('305','Mallika',7500,'HR','female');
```

```
Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
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('Total number of rows :'||emp1%rowcount);
close emp1;
end;
```

```
Statement processed.
Employee information:  302 Sandeep 6500 Ac male
Employee information:  303 Sidharth 7500 HR male
Employee information:  304 Seenath 7500 Ac female
Total number of rows :3
```

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(accession_no int , title varchar(20), publisher varchar(20), publishedDate
date, author varchar(20), status varchar(30));
insert into book values( 2511,'Golden Days','cp','21-jan-2009','john','issued');
insert into book values( 2512,'Heal','cp','30-mar-2010','malik','present in the library');
insert into book values( 2513,'Heaven','cp','21-june-2011','sonu','sent for binding');
insert into book values( 2514,'Secret Mission','cp','01-sep-2016','johns','issued');
insert into book values( 2515,'Violin','cp','21-jan-2004','joppy','can not be issued');
insert into book values( 2516,'Mimics','cp','21-jan-2006','juosoop',' 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;
```

OUTPUTS:

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

Table created.
```

```
insert into book values( 2511,'Golden Days','cp','21-jan-2009','john','issued');
insert into book values( 2512,'Heal','cp','30-feb-2010','malik','present in the library');
insert into book values( 2513,'Heaven','cp','21-march-2011','sonu','sent for binding');
insert into book values( 2514,'Secret Mission','cp','01-sep-2016','johns','issued');
insert into book values( 2515,'Voilin','cp','21-jan-2004','joppy','can not be issued');
insert into book values( 2516,'Mimics','cp','21-jan-2006','juosoop',' issued');

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.
```

```
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.
```

```
1  SELECT * FROM book;
```

ACCESSION_NO	TITLE	PUBLISHER	PUBLISHEDDATE	AUTHOR	STATUS
2511	Golden Days	cp	21-JAN-09	john	issued
2513	Heaven	cp	21-MAR-11	sonu	sent for binding
2514	Secret Mission	cp	01-SEP-16	johns	issued
2515	Voilin	cp	21-JAN-04	joppy	can not be issued
2516	Mimics	cp	21-JAN-06	juosoop	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.

PL/SQL 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(211,'Crayon',100,150);
insert into inventory values(311,'Pen',50,100);
insert into inventory values(411,'Marker',200,150);
insert into inventory values(511,'Pencil',500,250);
select * from inventory;
```

OUTPUT:

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

```
Table created.
```

```
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.
```

```
1  insert into inventory values(211,'Crayon',100,150);
2  insert into inventory values(311,'Pen',50,100);
3  insert into inventory values(411,'marker',200,150);
4  insert into inventory values(511,'Pencil',500,250);
5  select * from inventory;
```

```
1 row(s) inserted.


1 row(s) inserted.


1 row(s) inserted.
```

PDTID	PDTNAME	QTY	REORDER_LEVEL
211	Crayon	100	150
311	Pen	50	100
411	marker	200	0
511	Pencil	500	0

Download CSV

4 rows selected.