

**FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)**  
**DEPARTMENT OF COMPUTER APPLICATIONS**

**ADVANCED DBMS 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.

**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');

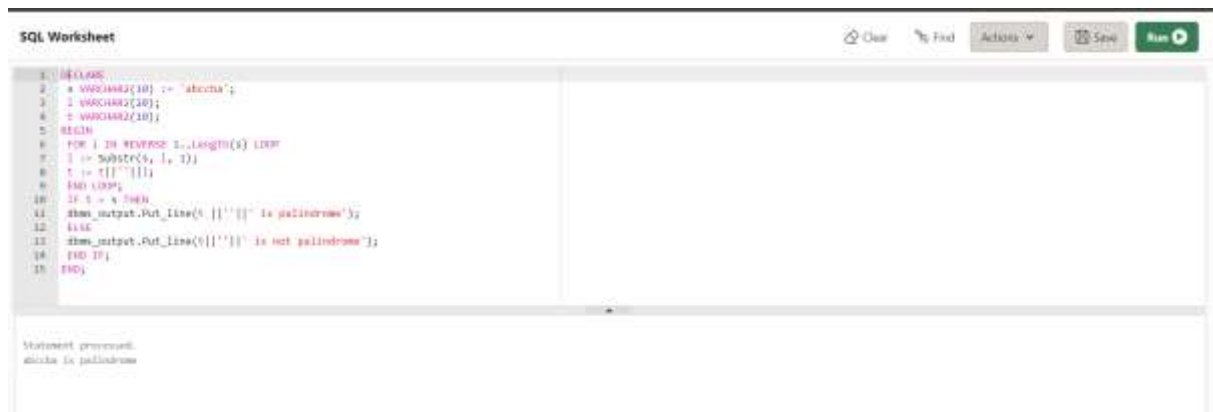
ELSE

dbms_output.Put_line(t||' is not palindrome');

END IF;

END;
```

## OUTPUT



```
SQL Worksheet
1 DECLARE
2   s VARCHAR2(10) := 'abcba';
3   t VARCHAR2(10);
4   i VARCHAR2(10);
5 BEGIN
6   FOR i IN REVERSE 1..LENGTH(s) LOOP
7     t := SUBSTR(s, i, 1);
8     s := t||s||t;
9   END LOOP;
10  IF s = s REVERSE THEN
11    DBMS_OUTPUT.PUT_LINE(s||' is palindrome');
12  ELSE
13    DBMS_OUTPUT.PUT_LINE(s||' is not palindrome');
14  END IF;
15 END;
```

Statement processed.  
abcba 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:=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



The screenshot shows an SQL Worksheet interface with a toolbar at the top containing buttons for 'Clear', 'First', 'Actions', 'Save', and 'Run'. The main area displays a PL/SQL script with line numbers 1 through 26. The script includes variable declarations, a swap function, and conditional logic for calculating power and square roots. Below the script, the output of the execution is shown, indicating successful completion and the values of the variables.

```

1 DECLARE
2   a INTEGER:=12;
3   b INTEGER:=9;
4   temp INTEGER:=0;
5   c INTEGER;
6   vala INTEGER;
7 BEGIN
8   IF a > b THEN
9     temp:=a;
10    a:=b;
11    b:=temp;
12    DBMS_OUTPUT.PUT_LINE('After swapping the a value is '||a||' and b value is '||b||);
13   IF MOD(b,2) =0 THEN
14     vala:= a ** b;
15     DBMS_OUTPUT.PUT_LINE('Cube is : '||vala);
16   ELSE
17     DBMS_OUTPUT.PUT_LINE('First number is even');
18
19
20   END IF;
21   ELSIF a < b THEN
22     c:=a **b;
23     DBMS_OUTPUT.PUT_LINE('Power is : '||c);
24   ELSIF a=b THEN
25     DBMS_OUTPUT.PUT_LINE('Square root of a is : '||(SQRT(a)));
26     DBMS_OUTPUT.PUT_LINE('Square root of b is : '||(SQRT(b)));
27   END IF;
28 END;

```

Statement succeeded.  
 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

### **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;
```

### **OUTPUT**



SQL Worksheet

Clear Find Actions Save Run

```
1 DECLARE
2 a NUMBER:=0;
3 b NUMBER:=1;
4 c NUMBER;
5 BEGIN
6 DBMS_OUTPUT.PUT(a||"||B||");
7 FOR I IN 3..10 LOOP
8 c:=a+b;
9 DBMS_OUTPUT.PUT(c||");
10 a:=b;
11 b:=c;
12 END LOOP;
13 DBMS_OUTPUT.PUT_LINE("");
14 END;
```

A	B
0	1
1	1
2	2
3	3
4	5
5	8
6	13
7	21
8	34
9	55
10	89

Statement processed  
2022/08/13 15:58

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



The screenshot shows an SQL Worksheet interface with a toolbar at the top containing buttons for 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The main area displays a PL/SQL script with line numbers 1 through 37. The script includes table creation, data insertion, a salary update loop, and output statements. Below the script, the output is displayed, showing the execution of the script and the resulting data for the 'employee' table.

```
1 create table employee(emp_no int,emp_name varchar(20),emp_post
2 varchar(20),emp_salary double(10,2));
3 insert into employee values(100,'Hans','DB',25000);
4 insert into employee values(101,'Hans','RM',20000);
5 insert into employee values(102,'Hans','Accountant',15000);
6 insert into employee values(103,'Hans','Clock',10000);
7 insert into employee values(104,'Hans','Person',5000);
8 declare
9 emp employee%rowtype;
10 salary employee.emp_salary%type;
11 emp_rec employee%rowtype;
12 begin
13 emno:=100;
14 select emp_salary into salary from employee where emp_no=emno;
15 if salary<7000 then
16 update employee set emp_salary=emp_salary * (5/100 where
17 emp_no=emno;
18 with
19 dbms_output.put_line('no more increment');
20 end if;
21
22 select * into emp_rec from employee where emp_no=emno;
23
24 dbms_output.put_line('Employee num: '||emp_rec.emp_no);
25 dbms_output.put_line('Employee name: '||emp_rec.emp_name);
26 dbms_output.put_line('Employee post: '||emp_rec.emp_post);
27 dbms_output.put_line('Employee salary: '||emp_rec.emp_salary);
28
29
30
31
32
33
34
35
36
37 end;
```

Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

Statements processed.

No more increment

Employee num: 100

Employee name: Hans

Employee post: Clock

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);

**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,'mca',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.PUT_LINE('Total students in MCA department is:'||c);
```

```
END;
```

## OUTPUT

```
1 create table class(cls_id int,cls_name varchar(20),cls_std int);
2 insert into class values(201,'mcu',90);
3 insert into class values(202,'mcu',90);
4 insert into class values(203,'bca',57);
5 insert into class values(204,'bca',55);
6 insert into class values(205,'mcu',52);
7 CREATE OR REPLACE FUNCTION total_std
8 RETURN NUMBER IS
9     total NUMBER(5):=0;
10 BEGIN
11     SELECT sum(cls_std) INTO total FROM class WHERE cls_name='mcu';
12     RETURN total;
13 END;
14 DECLARE
15     c NUMBER(5);
16 BEGIN
17     c:=total_std();
18     DBMS_OUTPUT.PUT_LINE('Total students in mca department is: '||c);
19 END;
```

Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

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.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
Clear Find Actions Save Run

1 CREATE OR REPLACE PROCEDURE IncrSalary
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 = 100;
8 IF dpt = 'clerk' THEN
9 UPDATE emp SET salary = salary*salary* 5/100 ;
10 ELSEIF dpt = 'salesman' THEN
11 UPDATE emp SET salary = salary*salary* 5/100 ;
12 ELSEIF dpt = 'analyst' THEN
13 UPDATE emp SET salary = salary*salary* 10/100 ;
14 ELSEIF 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 = 104;
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;
25
26 Create table emp(emp_no int,emp_name varchar(100),salary int,emp_dpt varchar(10));
27 insert into emp values(101,'arun',50000,'salesman');
28 insert into emp values(102,'appu',6500,'manager');
29 insert into emp values(103,'ammu',7500,'clerk');
30 insert into emp values(104,'anitha',7500,'analyst');
31
32 DECLARE
33 BEGIN
34 IncrSalary();
35 END;
```

```
SQL Worksheet
Clear Find Actions Save Run

Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

Procedure created.

Statement processed.
Name: anitha
employee number: 104
salary: 8750
department: analyst
```

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

## PROGRAM CODE

```
create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20),dsgrt
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 dsgr = '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



The screenshot displays an SQL Worksheet interface. The top toolbar includes buttons for 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The main area contains a SQL script with the following lines:

```
1 create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20),dsgr varchar(20));
2 insert into emp values(201,'aron',70000,'sales','president');
3 insert into emp values(202,'jppu',8500,'ac','president');
4 insert into emp values(183,'anna',7500,'hr','manager');
5 insert into emp values(184,'wilma',7500,'hr','hr grade');
6 insert into emp values(185,'wilma',7500,'ac','president');
7
```

Below the script, the execution output is shown, indicating that the table was created and six rows were inserted:

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

```

1 DECLARE
2   total_rows number(2);
3   emp1 emp%rowtype;
4 BEGIN
5   UPDATE emp SET salary = salary + salary * 50/100 where dept = 'president';
6   IF sql%found THEN
7     dbms_output.put_line('no employee salary updated');
8   ELSE
9     total_rows := sql%rowcount;
10    dbms_output.put_line('total_rows || ' employee salary details updated');
11  END IF;
12 END;
13
14

```

Statement processed.  
1 employee salary details updated

EMP_NO	EMP_NAME	SALARY	EMP_DEPT	GENDER
101	arun	75000	sales	president
102	agge	6500	Ac	president
103	ammu	7500	HR	manager
104	snitha	7500	Ac	usr grade
105	anitha.c	12250	HR	president

Execution of SQL  
5 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;

```

```

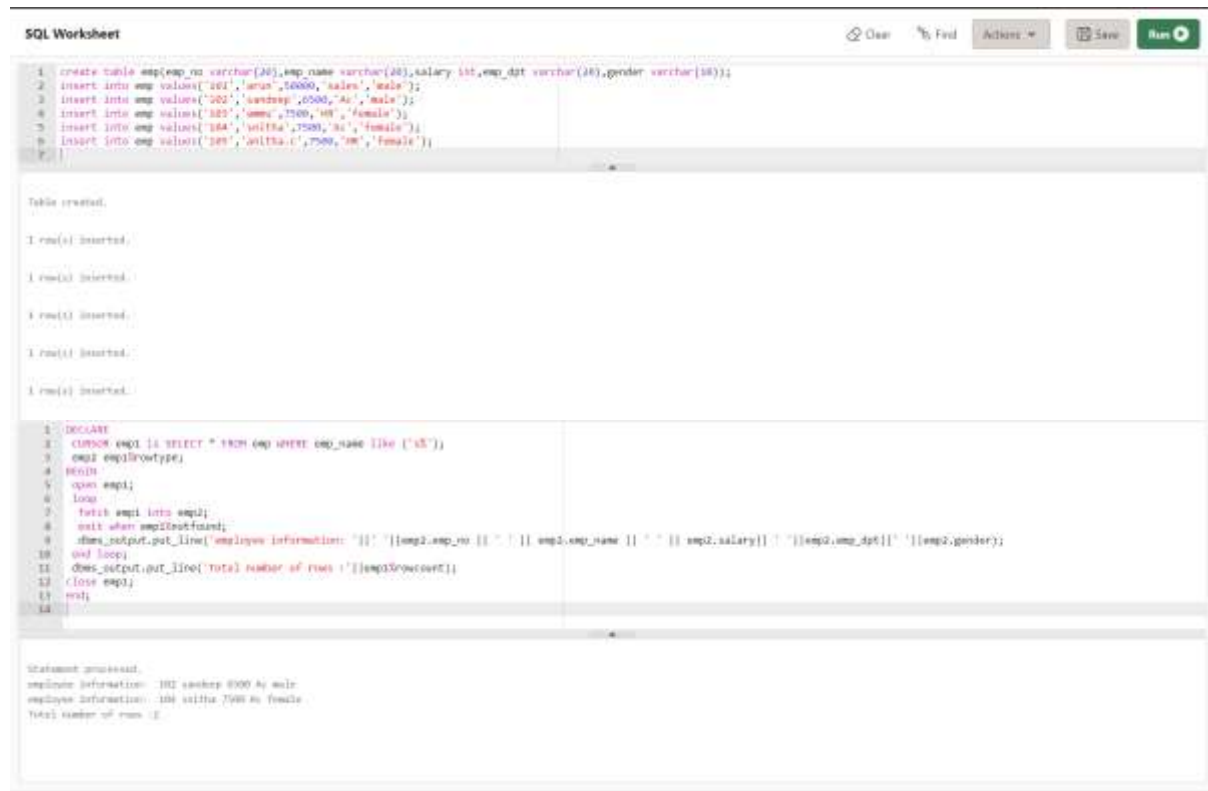
  exit when emp1%notfound;

```

```

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

## OUTPUT



The screenshot shows an SQL Worksheet interface with a toolbar at the top containing buttons for 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The main area displays the following SQL code:

```

1 create table emp(emp_no varchar(10),emp_name varchar(40),salary int,emp_dpt varchar(10),gender varchar(10));
2 insert into emp values('401','Arun','50000','sales','male');
3 insert into emp values('302','Sandeep','6500','Ac','male');
4 insert into emp values('325','Omni','7500','HR','female');
5 insert into emp values('344','Anitha','7500','Ac','female');
6 insert into emp values('389','Anitha','7500','Ac','female');
```

Below the code, the output of the execution is shown:

```

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

The next block of code is a PL/SQL block that declares a cursor, opens it, loops through the results, and outputs employee information and the total row count.

```

1 DECLARE
2 cursor emp1 is select * from emp where emp_name like ('%');
3 emp1 emp1%rowtype;
4 BEGIN
5 open emp1;
6 loop
7     fetch emp1 into emp1;
8     exit when emp1%rowcount=0;
9     dbms_output.put_line('employee information: '|| ' '||emp1.emp_no || ' ' || emp1.emp_name || ' ' || emp1.salary|| ' '||emp1.emp_dpt||' '||emp1.gender);
10 end loop;
11 dbms_output.put_line('Total number of rows :'||emp1%rowcount);
12 close emp1;
13 END;
```

The final output of the PL/SQL block is:

```

Statement processed.
employee information: 401 Arun 50000 Ac male
employee information: 302 Sandeep 6500 Ac male
employee information: 325 Omni 7500 HR female
Total number of rows : 6
```

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,'uvwxy','cp','21-jan-2006','juosoop','issued');
```

```
SELECT * FROM book;
```

# OUTPUT

SQL Worksheet

ClearFindActionsSaveRun

```
1 CREATE TABLE book(accession_no INT, title varchar(20), publisher varchar(30), publishedate date, author varchar(20), status varchar(10));
```

Table created.

SQL Worksheet

ClearFindActionsSaveRun

```
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 row.publishedate < add_months(temp, -180) THEN
10 row.status := 'cannot be issued';
11 END IF;
12 END IF;
13 END;
```

Trigger created.

SQL Worksheet

ClearFindActionsSaveRun

```
1 insert into book values( 2511,'abcd','cp','21-Jan-2000','john','issued');
2 insert into book values( 2512,'efgh','cp','30-Mar-2020','malik','present in the library');
3 insert into book values( 2513,'hijk','cp','21-June-2011','sara','sent for binding');
4 insert into book values( 2514,'lmno','cp','01-Sep-2010','johns','issued');
5 insert into book values( 2515,'pqrs','cp','21-Jan-2004','jenny','can not be issued');
6 insert into book values( 2516,'vwxyz','cp','21-Jan-2006','jussamp','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.

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

ACCESSION_NO	TITLE	PUBLISHER	PUBLISHEDATE	AUTHOR	STATUS
2511	abcd	cp	21-JAN-00	John	Issued
2512	efgh	cp	30-MAR-18	malik	present in the library
2513	hijk	cp	21-JUN-11	sara	sent for binding
2514	lmno	cp	01-SEP-10	Johns	Issued
2515	pqrs	cp	21-JAN-04	Jenny	cannot be issued
2516	vwxyz	cp	21-JAN-06	Jussamp	cannot be issued

Download CSV  
0 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.

### **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

ClearFindActionsSaveRun

```
1 create table inventory(petid number primary key, petname varchar(50), qty int,reorder_level number);
2
3
```

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;
12
13
```

Trigger created.

```
1 insert into inventory values(101,'pencil',100,150);
2 insert into inventory values(112,'tag',50,100);
3 insert into inventory values(121,'marker',200,100);
4 insert into inventory values(131,'notebook',500,250);
5 select * from inventory;
```

```
1 row(s) inserted.
2 row(s) inserted.
3 row(s) inserted.
4 row(s) inserted.
```

PETID	PETNAME	QTY	REORDER_LEVEL
101	pencil	100	150
112	tag	50	100
121	marker	200	0
131	notebook	500	0

Download CSV  
4 rows selected