

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

Code:

```
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:

```

1 DECLARE
2   s VARCHAR2(10) := 'malayalam';
3   l VARCHAR2(20);
4   t VARCHAR2(10);
5 BEGIN
6   FOR i IN REVERSE 1..Length(s) LOOP
7     l := Substr(s, i, 1);
8     t := t || l;
9   END LOOP;
10
11   IF t = s THEN
12     dbms_output.Put_line(t || ' is palindrome');
13   ELSE
14     dbms_output.Put_line(t || ' is not palindrome');
15   END IF;
16 END;
17
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.

Code:

DECLARE

a INTEGER :=5;

b INTEGER :=4;

temp INTEGER:=0;

c INTEGER;

d INTEGER :=2;

cube INTEGER;

BEGIN

IF a > b THEN

temp :=a;

a :=b;

b := temp;

DBMS_OUTPUT.PUT_LINE('After the swapping the a value is ' || a || ' and b value is ' || b);

IF MOD(b,d) !=0 THEN

cube :=a* a * a;

DBMS_OUTPUT.PUT_LINE('cube of a is:' || cube);

ELSE

```

        DBMS_OUTPUT.PUT_LINE('The 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('sqare root of a is:' || (SQRT(a)));
    DBMS_OUTPUT.PUT_LINE('sqare root of b is:' || (SQRT(b)));
END IF;
END;

```

Output:

Statement processed.
 After the swapping the a value is 4 and b value is 5
 cube of a is:64

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

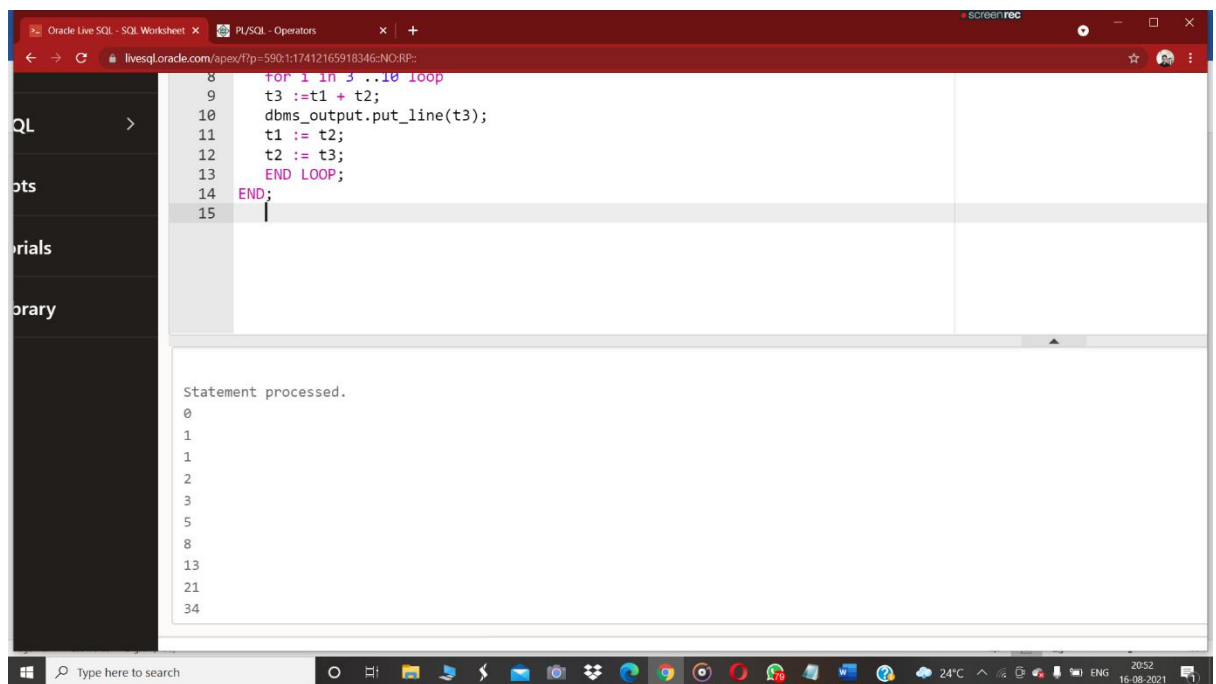
Code:

```

DECLARE
    t1 NUMBER :=0;
    t2 NUMBER :=1;
    t3 NUMBER ;
BEGIN
    dbms_output.put_line(t1);
    dbms_output.put_line(t2);
    for i in 3 ..10 loop
        t3 :=t1 + t2;
        dbms_output.put_line(t3);
        t1 := t2;
        t2 := t3;
    END LOOP;
END;

```

Output:



The screenshot shows the Oracle Live SQL interface. The top bar indicates the session is 'PL/SQL - Operators'. The left sidebar contains navigation links: 'QL', 'ots', 'rials', and 'brary'. The main editor area contains the following PL/SQL code:

```
8  FOR i IN 3 ..10 LOOP
9      t3 :=t1 + t2;
10     dbms_output.put_line(t3);
11     t1 := t2;
12     t2 := t3;
13     END LOOP;
14 END;
15
```

The output area below the editor shows the result of the statement execution:

```
Statement processed.
0
1
1
2
3
5
8
13
21
34
```

The Windows taskbar at the bottom shows the system clock as 20:52 on 16-08-2021.

- 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 EMP(emp_no int primary key,emp_name varchar(20),salary int);
```

```
insert into EMP values(101,'arun',50000);
```

```
insert into EMP values(102,'arun',6500);
```

```
insert into EMP values(103,'arun',7500);
```

DECLARE

```
emp1 EMP%rowtype;
```

```
sal EMP.salary%type;
```

BEGIN

```
SELECT salary INTO sal FROM EMP WHERE emp_no = 102;
```

```
IF sal <= 7500 THEN
```

```

UPDATE EMP SET salary = salary+salary* 15/100 WHERE emp_no = 102;

ELSE

DBMS_OUTPUT.PUT_LINE ('NO INCREMENT');

END IF;

SELECT * into emp1 FROM EMP WHERE emp_no = 102;

DBMS_OUTPUT.PUT_LINE ('Name: ' || emp1.emp_name);

DBMS_OUTPUT.PUT_LINE ('employee number: ' || emp1.emp_no);

DBMS_OUTPUT.PUT_LINE ('salary: ' || emp1.salary);

END;

```

OUTPUT:

```

Statement processed.
Name: arun
employee number: 102
salary: 8596

```

- 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);

Table creation And insertion

```

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

insert into class values('MCA21','S2A',59);

insert into class values('MCA21','S2B',58);

insert into class values('MCA20','S5A',40);

insert into class values('MCA20','S5B',34);

```

function code:

```

CREATE OR REPLACE FUNCTION findTotalStrength

RETURN NUMBER IS

s_count NUMBER(20):=0;

BEGIN

```

```
SELECT sum(strength) INTO s_count FROM class;

RETURN (s_count);

END;
```

Function Output:

Function created.

Function call

```
DECLARE

c NUMBER(5):=0;

BEGIN

C:= findTotalStrength();

DBMS_OUTPUT.PUT_LINE('Total students in mca department is:' || c);

END;
```

Output:

Statement processed.

Total students in mca department is:191

- 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.

procedure code

```
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;

```

table creation

```

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');
```

calling function

```
DECLARE
```

```
BEGIN
```

```
    increSalary();
```

```
END;
```

Output:

```
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%

Table creation and insertion command:

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

Cursor code:

```
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

The screenshot shows an Oracle Live SQL Worksheet with a PL/SQL script and its output. The script is as follows:

```
13 UPDATE emp SET salary = salary + salary * 50/100 where dsgt = 'president';
14 IF sql%notfound THEN
15   dbms_output.put_line('no employee salary updated');
16 ELSIF sql%found THEN
17   total_rows := sql%rowcount;
18   dbms_output.put_line( total_rows || ' employee salary details updated');
19 end if;
20 end;*/
21 select * from emp;
```

The output of the script is a table with 5 rows and 5 columns:

EMP_NO	EMP_NAME	SALARY	EMP_DPT	DSGT
101	arun	75000	sales	president
102	appu	9750	Ac	president
103	ammu	7500	HR	manager
104	anitha	7500	Ac	snr grade
105	anitha.c	11250	HR	president

Below the table, there is a link to "Download CSV".

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

Table creation and insert command:

```
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');

```

Cursor code:

```

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:

```

Statement processed.
employee information: 102 sandeep 6500 Ac male
employee information: 104 snitha 7500 Ac female
Total number of rows :2

```

- 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.

Table creation:

```

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

```

Trigger code:

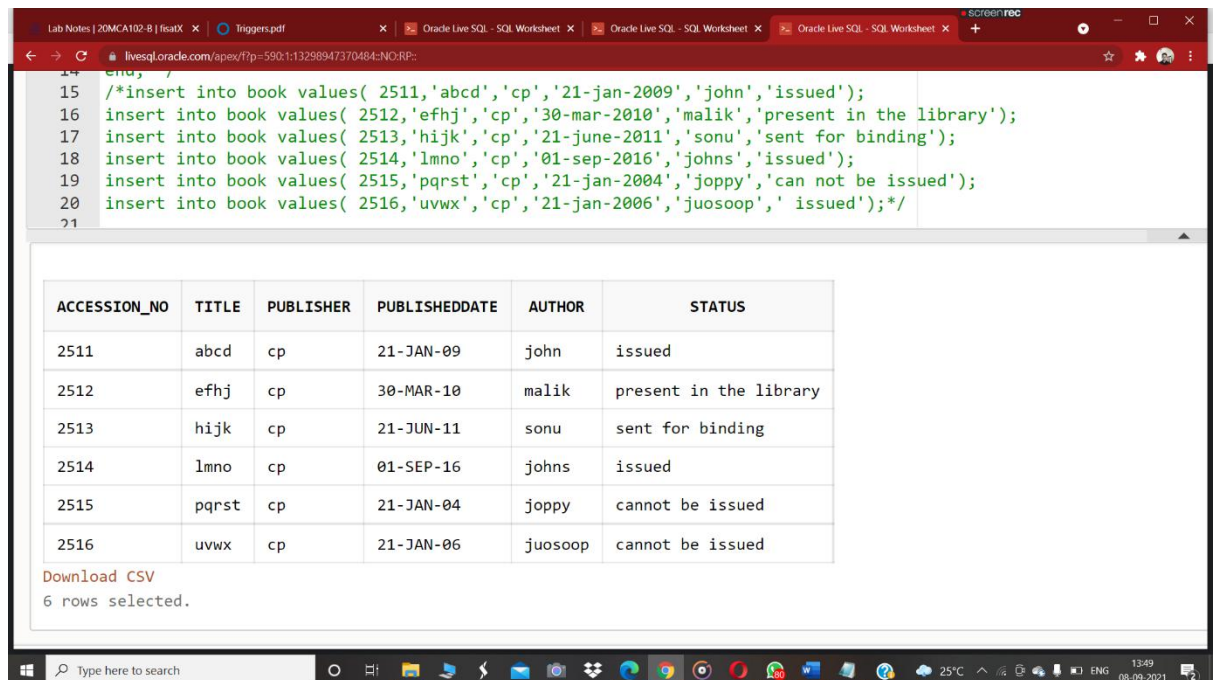
```
CREATE OR REPLACE TRIGGER search1
before insert ON book
FOR EACH ROW
declare
temp date;
BEGIN
select sysdate into temp from dual;
if inserting then
if :new.publishedDate < add_months(temp, -180) then
:new.status:='cannot be issued' ;
end if;
end if;
end;
```

inserting command:

```
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:



The screenshot shows a web browser window with multiple tabs, including 'Lab Notes | 20MCA102-B | fisatX', 'Triggers.pdf', and several 'Oracle Live SQL - SQL Worksheet' tabs. The active worksheet displays a series of SQL insert statements for a table named 'book'. Below the code, a table view shows the results of the inserts, with 6 rows selected. The table has columns: ACCESSION_NO, TITLE, PUBLISHER, PUBLISHEDDATE, AUTHOR, and STATUS.

```
15 /*insert into book values( 2511,'abcd','cp','21-jan-2009','john','issued');
16 insert into book values( 2512,'efhj','cp','30-mar-2010','malik','present in the library');
17 insert into book values( 2513,'hijk','cp','21-june-2011','sonu','sent for binding');
18 insert into book values( 2514,'lmno','cp','01-sep-2016','johns','issued');
19 insert into book values( 2515,'pqrst','cp','21-jan-2004','joppy','can not be issued');
20 insert into book values( 2516,'uvwxy','cp','21-jan-2006','juosoop',' issued');*/
21
```

ACCESSION_NO	TITLE	PUBLISHER	PUBLISHEDDATE	AUTHOR	STATUS
2511	abcd	cp	21-JAN-09	john	issued
2512	efhj	cp	30-MAR-10	malik	present in the library
2513	hijk	cp	21-JUN-11	sonu	sent for binding
2514	lmno	cp	01-SEP-16	johns	issued
2515	pqrst	cp	21-JAN-04	joppy	cannot be issued
2516	uvwxy	cp	21-JAN-06	juosoop	cannot be issued

Download CSV
6 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.

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

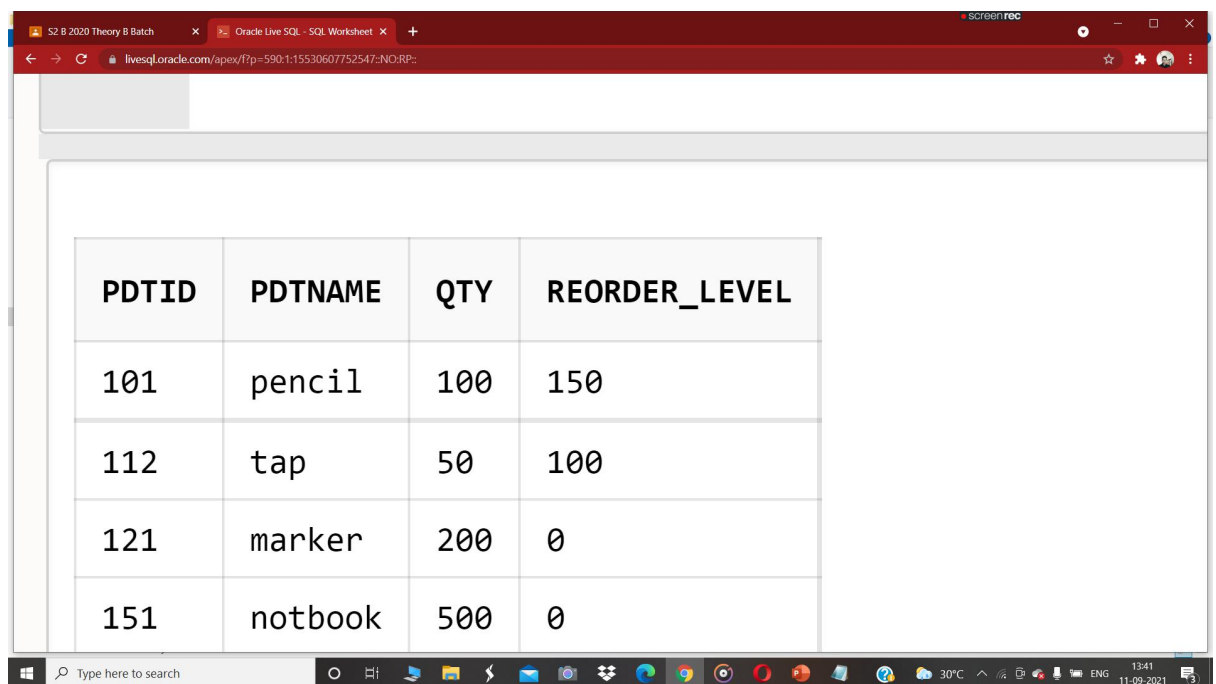
if inserting then

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:



PDTID	PDTNAME	QTY	REORDER_LEVEL
101	pencil	100	150
112	tap	50	100
121	marker	200	0
151	notbook	500	0