

FEDERAL INSTITUTE OF SCIENCE AND TECHNOLOGY (FISAT)
DEPARTMENT OF COMPUTER APPLICATIONS
ADVANCED DBMS LAB

Lab Cycle 2

Rahul Roy

- 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) := 'abccba';
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  IF t = s THEN
11    dbms_output.put_line(t||' is palindrome');
12  ELSE
13    dbms_output.put_line(t||' is not palindrome');
14  END IF;
15 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.

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 'Clear', 'Find', 'Actions', 'Save', and 'Run' buttons. The main area is divided into two panes. The left pane contains a PL/SQL program with line numbers 1 through 26. The right pane shows the output of the program.

```
1 DECLARE
2   a INTEGER:=12;
3   b INTEGER:=9;
4   temp INTEGER:=0;
5   c INTEGER;
6   cube 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     cube:=a * a * a;
15     DBMS_OUTPUT.PUT_LINE('Cube is '||cube);
16   ELSE
17     DBMS_OUTPUT.PUT_LINE('first number is even');
18
19   END IF;
20   ELSEIF a < b THEN
21     c:=a **b;
22     DBMS_OUTPUT.PUT_LINE('Power is :'||c);
23   ELSEIF a=b THEN
24     DBMS_OUTPUT.PUT_LINE('Square root of a is :'||(SQRT(a)));
25     DBMS_OUTPUT.PUT_LINE('Square root of b is :'||(SQRT(b)));
26 END IF;
```

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

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

The screenshot shows an SQL Worksheet interface. At the top, there are buttons for 'Clear', 'Find', 'Actions', 'Save', and 'Run'. The main area contains a PL/SQL program with line numbers 1 through 14. The program declares three variables: a, b, and c, all of type NUMBER. a is initialized to 0, b to 1, and c to 0. A BEGIN block follows, containing a loop that runs from 1 to 10. Inside the loop, a is incremented by b, c is incremented by a, and b is incremented by c. After the loop, the final values of a, b, and c are printed. The output at the bottom shows 'Statement processed.' and a timestamp '0112358132134'.

```
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 1..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;
```

Statement processed.
0112358132134

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

SQL Worksheet
Clear
Find
Actions
Save
Run

```

1 create table employee(emp_no int,emp_name varchar(20),emp_post
2 varchar(20),emp_salary decimal(10,2));
3 insert into employee values(103,'Rahul','MD',25000);
4 insert into employee values(105,'Ravi','HR',20000);
5 insert into employee values(107,'Rani','Accountant',15000);
6 insert into employee values(109,'Renu','Clerk',10000);
7 insert into employee values(201,'Ramu','Peon',5000);
8 Declare
9 emno employee.emp_no%type;
10 salary employee.emp_salary%type;
11 emp_rec employee%rowtype;
12 begin
13 emno:=109;
14 select emp_salary into salary from employee where emp_no=emno;
15 if salary<7500 then
16 update employee set emp_salary=emp_salary * 15/100 where
17 emp_no=emno;
18 else
19 dbms_output.put_line('No more increment');
20 end if;
21
22 select * into emp_rec from employee where emp_no=emno;
23 dbms_output.put_line('Employee num: '||emp_rec.emp_no);
24 dbms_output.put_line('Employee name: '||emp_rec.emp_name);
25 dbms_output.put_line('Employee post: '||emp_rec.emp_post);
26 dbms_output.put_line('Employee salary: '||emp_rec.emp_salary);
27 end;

```

Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

Statement processed.
No more increment
Employee num: 109
Employee name: Renu
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);

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,'mca',60);
3 insert into class values(202,'mca',60);
4 insert into class values(203,'bca',57);
5 insert into class values(204,'bca',59);
6 insert into class values(205,'mca',62);
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='mca';
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 increSalary
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 = 104;
8 IF dpt = 'clerk' THEN
9     UPDATE emp SET salary = salary+salary* 5/100 ;
10 ELSIF dpt = 'salesman' THEN
11     UPDATE emp SET salary = salary+salary* 7/100 ;
12 ELSIF dpt = 'analyst' THEN
13     UPDATE emp SET salary = salary+salary* 10/100 ;
14 ELSIF 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(20),salary int,emp_dpt varchar(20));
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     increSalary();
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: 8250
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),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

SQL Worksheet

Clear

Find

Actions

Save

Run

```

1 create table emp(emp_no int,emp_name varchar(20),salary int,emp_dpt varchar(20),dsge varchar(20));
2 insert into emp values(101,'arun',50000,'sales','president');
3 insert into emp values(102,'appu',6500,'Ac','president');
4 insert into emp values(103,'ammu',7500,'HR','manager');
5 insert into emp values(104,'anitha',7500,'Ac','snr grade');
6 insert into emp values(105,'anitha.c',7500,'HR','president');
7

```

Table created.

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
6     UPDATE emp SET salary = salary + salary * 50/100 where dsge = 'president';
7     IF sql%notfound THEN
8         dbms_output.put_line('no employee salary updated');
9     ELSEIF sql%found THEN
10        total_rows := sql%rowcount;
11        dbms_output.put_line( total_rows || ' employee salary details updated');
12    end if;
13 end;
14

```

Statement processed.

3 employee salary details updated

EMP_NO	EMP_NAME	SALARY	EMP_DPT	DSGE
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

Download CSV

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

SQL Worksheet

Clear

Find

Actions

Save

Run

```

1 create table emp(emp_no varchar(20),emp_name varchar(20),salary int,emp_dpt varchar(20),gender varchar(10));
2 insert into emp values('101','arun',50000,'sales','male');
3 insert into emp values('102','sandeep',6500,'Ac','male');
4 insert into emp values('103','ammu',7500,'HR','female');
5 insert into emp values('104','snitha',7500,'Ac','female');
6 insert into emp values('105','anitha.c',7500,'HR','female');
7

```

Table created.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

```

1 DECLARE
2 CURSOR emp1 is SELECT * FROM emp WHERE emp_name like ('s%');
3 emp2 emp1%rowtype;
4 BEGIN
5 open emp1;
6 loop
7 fetch emp1 into emp2;
8 exit when emp1%notfound;
9 dbms_output.put_line('employee information: '||emp2.emp_no || ' ' || emp2.emp_name || ' ' || emp2.salary|| ' '||emp2.emp_dpt|| ' '||emp2.gender);
10 end loop;
11 dbms_output.put_line('Total number of rows :'||emp1%rowcount);
12 close emp1;
13 end;
14

```

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.

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

Clear

Find

Actions

Save

Run

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

Table created.

SQL Worksheet

Clear

Find

Actions

Save

Run

```

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;
14

```

Trigger created.

SQL Worksheet

Clear Find Actions Save Run

```

1 insert into book values( 2511,'abcd','cp','21-jan-2009','john','issued');
2 insert into book values( 2512,'efhj','cp','30-mar-2010','malik','present in the library');
3 insert into book values( 2513,'hijk','cp','21-june-2011','sonu','sent for binding');
4 insert into book values( 2514,'lmno','cp','01-sep-2016','johns','issued');
5 insert into book values( 2515,'pqrst','cp','21-jan-2004','joppy','cannot be issued');
6 insert into book values( 2516,'uvwx','cp','21-jan-2006','juosoop','issued');
7

```

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	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	uvwx	cp	21-JAN-06	juosoop	cannot be issued

Download CSV
6 rows selected.

© 2021 Oracle Corporation - Privacy - Terms of Use
Oracle Learning Library - Ask Tom - Dev Gym - Database Documentation - Follow on Twitter

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(pdtid number primary key, pdtname varchar(10), 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,'tap',50,100);
3 insert into inventory values(121,'marker',200,150);
4 insert into inventory values(151,'notbook',500,250);
5 select * from inventory;
```

1 row(s) inserted.

1 row(s) inserted.

1 row(s) inserted.

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

Download CSV

4 rows selected.