EXP. NO. 5

Sachin Kolhe **2020510034**

PLSQL Basics, Cursors and Triggers

- **PlSQL Basics**

1)Write a PL/SQL block to take the number and string from user and display it.

Code:

```
DECLARE
 s varchar2(20):='Sachin';
 n number:=12;
BEGIN
 dbms_output.put_line(s);
 dbms_output.put_line(n);
END;
```

Output:

```
Statement processed.
Sachin
12
```

2) Write a PL/SQL block to add two numbers.

Code:

```
DECLARE
x NUMBER:=34;
y NUMBER:=21;
BEGIN
dbms_output.put_line(x+y);
END;
```

Output:

```
Statement processed.
55
```

3) Write a PL/SQL block to find the greatest among three numbers.

Code:

```
DECLARE
    a NUMBER:=45;
    b NUMBER:=23;
    c NUMBER:=58;
BEGIN
   IF a > b AND a > c THEN
   dbms_output.put_line(a || ' is the greatest.');
    ELSIF b > a AND b > c THEN
    dbms_output.put_line(b || ' is the greatest.');
    ELSE
     dbms_output.put_line(c || ' is the greatest.');
     END IF;
END;
```

Output:

```
Statement processed.
58 is the greatest.
```

4) Write a PL/SQL block to find out sum of first five numbers.

Code:

```
DECLARE
        a NUMBER := 0;
BEGIN
        FOR i IN 1..5
        LOOP
        a := a + i;
        END LOOP;
dbms_output.put_line('The sum of the first 5 numbers is ' || a);
END;
```

Output:

```
Statement processed.
The sum of the first 5 numbers is 15
```

5) Write a PL/SQL block to retrieve values from the table

Code:

```
 DECLARE
    cursor c is SELECT * FROM customer_34;
    z c%rowtype;
BEGIN
    open c;
    fetch c into z;
    while(c%found) loop
    dbms_output.put_line(z.cname);
    fetch c into z;
```

```
    end loop;
    close c;
END;
```

Output:

```
Statement processed.
sunil
mehul
madhuri
sandip
shivani
pramod
kranti
anil
mandar
naren
```

Cursor

1) Display the depositor names and amount of virar branch using cursor.

Code:

```
DECLARE
    cursor c is SELECT cname, amount FROM deposit_34 WHERE bname =
    'virar';
    name deposit_34.cname%type;
    amt deposit_34.amount%type;
BEGIN
    FOR i in c LOOP
    dbms_output.put_line('Name: ' || i.cname);
    dbms_output.put_line('Amount: ' || i.amount);
    END LOOP;
END;
```

```
Statement processed.
Name: shivani
Amount: 1001
```

2) Display the name and amount of virar branch using parametric cursor.

Code:

```
DECLARE
    cursor c(branch varchar2) is SELECT cname, amount FROM deposit_34
    WHERE bname = branch;
BEGIN
    FOR i in c('virar') LOOP
    dbms_output.put_line('Name: ' || i.cname);
    dbms_output.put_line('Amount: ' || i.amount);
    END LOOP;
```

```
END;
```

Output:

```
Statement processed.
Name: shivani
Amount: 1001
```

3) Display total number of rows of a customer table using for loop.

Code:

```
DECLARE
   cursor c is SELECT * FROM customer_34;
   x NUMBER := 0;
BEGIN
   FOR i IN c LOOP
   x := x+1;
  END LOOP;
  dbms_output.put_line('There are ' || x || ' rows in the table.');
END;
```

```
Statement processed.
There are 10 rows in the table.
```

4) Display total number of rows of a customer table using while loop.

Code:

```
DECLARE
    cursor c is SELECT * FROM customer_34;
    z c%rowtype;
    x NUMBER := 0;
BEGIN
    open c;
```

```
    fetch c into z;
    WHILE c%found LOOP
    x := x+1;
    fetch c into z;
    END LOOP;
    close c;
    dbms_output.put_line('There are ' || x || ' rows in the table.');
END;
```

```
Statement processed.
There are 10 rows in the table.
```

5) Display total number of rows of a customer table using LOOP..END LOOP and %NOTFOUND.

Code:

```
DECLARE
    cursor c is SELECT * FROM customer_34;
    x NUMBER := 0;
    z c%rowtype;
BEGIN
    open c;
    fetch c into z;
    LOOP
    x := x+1;
    fetch c into z;
    exit when c%notfound;
    END LOOP;
   close c;
```

```
    dbms_output.put_line('There are ' || x || ' rows in the table.');
END;
```

```
Statement processed.
There are 10 rows in the table.
```

6) Diplay total amount of the depositors of virar branch.

Code:

```
DECLARE
    cursor c is SELECT amount FROM deposit_34 d WHERE
bname='VIRAR';
    amt NUMBER := 0;
    z c%rowtype;
BEGIN
    open c;
    fetch c into z;
    while c%found loop
    amt := amt + z.amount;
    fetch c into z;
    end loop;
    dbms_output.put_line('Total amount: ' || amt);
END;
```

```
Statement processed.
Total amount: 0
```

7) Calculate and display depositor name having forth maximum amount.

```
DECLARE
    cursor c is SELECT cname FROM deposit_34 ORDER BY amount DESC;
```

```
    z c%rowtype;
    x INTEGER := 0;
BEGIN
    for i in c loop
    if(x = 3) THEN
    dbms_output.put_line('Name: ' || i.cname);
    end if;
    x := x + 1;
    end loop;
END;
```

```
Statement processed.
Name: mehul
```

Trigger

Scenario 1 : (Q1 to 4)

Create a table emp(empid,ename,salary)

1) Create a trigger on emp table that does not allow salary to be less than 10000.

```
create or replace trigger emp_newtrig
after insert on emp_34
for each row
DECLARE
BEGIN
        if(:new.salary<10000) then
        raise_application_error(-20001,'Salary cant be less than 10000');
        end if;
END;
/
```

Output:

```
Trigger created.
```

```
insert into emp_34 values (1, 'Sham', 2000);
```

Output:

```
ORA-20001: Salary cant be less than 10000 ORA-06512: at
"SQL_MGGQKRTUYUBWFJIBZZEEIRICV.EMP_NEWTRIG", line 4 ORA-06512: at "SYS.DBMS_SQL", line
1721
```

2) Create a trigger on emp table that does not allow empid to be more than 2 digits.

```
create or replace trigger emp_newtrig
after insert on emp_34
for each row
declare
begin
if(:new.empid>99) then
raise_application_error(-20002,'Employee Id cant be more than 2 digits');
end if;
end;
/
```

Output:

```
Trigger created.
```

```
insert into emp_34 values (110, 'Rahul', 20000);
```

```
ORA-20002: Employee Id cant be more than 2 digits ORA-06512: at
"SQL_MGGQKRTUYUBWFJIBZZEEIRICV.EMP_NEWTRIG", line 4 ORA-06512: at "SYS.DBMS_SQL", line 1721
```

3) Create a trigger which does not allow DML operations on emp table if the user name is System.

```
create or replace trigger emp_newtrig
after insert on emp_34
for each row
DECLARE
uname varchar2(50);
BEGIN
select user into uname from dual;
if(uname='APEX_PUBLIC_USER') then
raise_application_error(-20001,'USER SHOULD NOT BE APEX PUBLC USER');
end if;
END;
```

```
insert into emp_34 values (10, 'Nikhil', 20000);
```

```
ORA-20003: System cannot do DML operation ORA-06512: at
"SQL_MGGQKRTUYUBWFJIBZZEEIRICV.EMP_NEWTRIG_4", line 6 ORA-06512: at "SYS.DBMS_SQL", line
1721
```

4) **Create a trigger that allows no DML operations on emp table to be performed on any weekdays but allow insertion on Sunday.**

Code:

```
CREATE OR REPLACE TRIGGER trg_sunday

AFTER INSERT or UPDATE OR DELETE ON emp_34

FOR EACH ROW

BEGIN

   IF (TO_CHAR(SYSDATE, 'D')!= '7' OR TO_CHAR(SYSDATE, 'D')!= '6') THEN

      RAISE_APPLICATION_ERROR(-20004, 'Cannot Perform DML Operation :(
:(');


   elsif (TO_CHAR(SYSDATE, 'D')= '7') THEN

   if(updating or deleting) THEN

      RAISE_APPLICATION_ERROR(-20005, 'You can only perform insert
operation :( :(');

   END IF;

    END IF;

END;
```

```
Trigger created.
```

Scenario 2 :

(Q 5)

Create 2 tables, tempfees (amount) and finalfees(amount) Insert some values in tempfees.

5) Create a trigger on tempfees when updation is performed then the old values of tempfees are copied into final fees.

```
Create  table tempfees_34(amount number);

Create  table finalfees_34(amount number);


insert into tempfees_34 values(2000);

1 row(s) inserted.

insert into tempfees_34 values(3000);

1 row(s) inserted.


create or replace trigger temp_newtrig

after update on tempfees_34

for each row

declare

begin

insert into finalfees_34 values(:old.amount);

end;

Trigger created.
```

update tempfees_34 set amount=5000 where amount=3000;

```
1 row(s) updated
```

select * from tempfees_34;

AMOUNT
2000
5000

select * from finalfees_34;

AMOUNT
3000