Ex.No: 7 *PL/SQL BLOCK CREATION AND USAGE OF VARIOUS COMPOSITE DATA TYPES*

Aim:

To practice with PL/SQL block creation and various composite data types

Description:

PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation to enhance the capabilities of SQL.

Syntax of a basic loop in PL/SQL programming language is:

```
LOOP
 //Sequence of statements;//
END LOOP;
```

Procedure:

1. **Execute any 5 sample PL/SQL programs**
2. **Execute any 5 PL/SQL program for your application**

Example:

Adding two Numbers

```
Declare
Var1 integer;
Var2 integer;
Var3 integer;
Begin
Var1:=&var1;
Var2:=&var2;
Var3:=var1+var2;
Dbms_output.put_line(var3);
End;
/
```

```
SQL> declare
  2  var1 integer;
  3  var2 integer;
  4  var3 integer;
  5  begin
  6  var1 :=&var1;
  7  var2 :=&var2;
  8  var3 := var1+var2;
  9  dbms_output.put_line (var3);
 10  end;
 11  /
Enter value for var1: 2
old   6: var1 :=&var1;
new   6: var1 :=2;
Enter value for var2: 7
old   7: var2 :=&var2;
new   7: var2 :=7;
9

PL/SQL procedure successfully completed.
```

PL/SQL Program to Find Factorial of a Number

```
declare
        n number;
        fac number:=1;
        i number;
begin
        n:=&n;
        for i in 1..n
        loop
                fac:=fac*i;
        end loop;
        dbms_output.put_line('factorial='||fac);
end;
/
```

```
SQL> declare
  2  n number;
  3  fac number:=1;
  4  i number;
  5  begin
  6  n:=&n;
  7  for i in 1..n
  8  loop
  9  fac:=fac*i;
 10  end loop;
 11  dbms_output.put_line('Factorial '||fac);
 12  end;
 13  /
Enter value for n: 5
old   6: n:=&n;
new   6: n:=5;
Factorial 120

PL/SQL procedure successfully completed.
```

PL/SQL Program for Armstrong Number

```
declare
   n number:=407;
   s number:=0;
   r number;
   len number;
   m number;
begin
   m:=n;
   len:=length(to_char(n));
   while n>0
   loop
      r:=mod(n,10);
      s:=s+power(r,len);
      n:=trunc(n/10);
   end loop;
   if m=s

   then

      dbms_output.put_line('armstrong number');

   else

      dbms_output.put_line('not armstrong number');

   end if;

      end;

      /
```

Output

```
SQL> set serveroutput on;
SQL> declare
  2   n number:=909;
  3   s number:=0;
  4   r number;
  5   len number;
  6   m number;
  7  begin
  8   m:=n;
  9   len:=length(to_char(n));
 10   while n>0
 11   loop
 12   r:=mod(n,10);
 13   s:=s+power(r,len);
 14   n:=trunc(n/10);
 15   end loop;
 16   if m=s
 17   then
 18   dbms_output.put_line('armstrong number');
 19   else
 20   dbms_output.put_line('not armstrong number');
 21   end if;
 22  end;
 23  /
not armstrong number

PL/SQL procedure successfully completed.

SQL>
```

PL/SQL Program to Check Number is Odd or Even

```
declare
n number:=&n;
begin
if mod(n,2)=0 then
dbms_output.put_line('number is even');
else
dbms_output.put_line('number is odd');
end if;
end;
/
```

```
SQL> declare
  2  n number:=&n;
  3  begin
  4  if mod(n,2)=0
  5  then
  6  dbms_output.put_line('number is even');
  7  else
  8  dbms_output.put_line('number is odd');
  9  end if;
 10  end;
 11  /
Enter value for n: 5
old   2: n number:=&n;
new   2: n number:=5;
number is odd

PL/SQL procedure successfully completed.
```

PL\SQL Program to print Fibonacci Series:

```
DECLARE
   n1 NUMBER := 0;
   n2 NUMBER := 1;
   next_term NUMBER;
   limit NUMBER := &limit;
BEGIN
   DBMS_OUTPUT.PUT_LINE('Fibonacci Series:');
   DBMS_OUTPUT.PUT(n1 || ' ' || n2 || ' ');
   FOR i IN 3..limit LOOP
      next_term := n1 + n2;
      DBMS_OUTPUT.PUT_LINE(next_term || ' ');
      n1 := n2;
      n2 := next_term;
   END LOOP;
END;
/
```

```
SQL> set serveroutput on;
SQL>  DECLARE
  2      n1 NUMBER := 0;
  3      n2 NUMBER := 1;
  4      next_term NUMBER;
  5      limit NUMBER := &limit;
  6  BEGIN
  7      DBMS_OUTPUT.PUT_LINE('Fibonacci Series:');
  8      DBMS_OUTPUT.PUT(n1 || ' ' || n2 || ' ');
  9      FOR i IN 3..limit LOOP
 10          next_term := n1 + n2;
 11          DBMS_OUTPUT.PUT_LINE(next_term || ' ');
 12          n1 := n2;
 13          n2 := next_term;
 14      END LOOP;
 15  END;
 16  /
Enter value for limit: 6
old   5:     limit NUMBER := &limit;
new   5:     limit NUMBER := 6;
Fibonacci Series:
0 1 1
2
3
5

PL/SQL procedure successfully completed.
```

PL/SQL program for your application

1.PREVIOUS CUSTOMERS PAYMENT DETAILS

```
BEGIN
DBMS_OUTPUT.PUT_LINE('Printing previous customer Payment Details');
  FOR payment_disp IN (SELECT * FROM payment) LOOP
    DBMS_OUTPUT.PUT_LINE(Payment_disp.Payment_id || ',' ||Payment_disp.Rental_id || ','
||Payment_disp.Amount || ',' || Payment_disp.Payment_date || ',' || Payment_disp.Payment_Mode );
end loop;
end;
/
```

Output:

```
SQL> BEGIN
  2  DBMS_OUTPUT.PUT_LINE('Printing previous customer Payment Details');
  3      FOR payment_disp IN (SELECT * FROM payment) LOOP
  4          DBMS_OUTPUT.PUT_LINE(Payment_disp.Payment_id || ',' ||Payment_disp.Rental_id ||
 ',' ||Payment_disp.Amount || ',' || Payment_disp.Payment_date || ',' || Payment_disp.Paymen
t_Mode );
  5  end loop;
  6  end;
  7  /
Printing previous customer Payment Details
56894,85454,1500,17/2/24,Cash
56895,85455,12450,18/2/24,GPAY
56896,85456,20000,18/2/24,NetBanking
56897,85457,250,19/2/24,NetBanking
56898,85458,2000,19/2/24,

PL/SQL procedure successfully completed.
```

2.TOTAL AMOUNT RECEIVED from customers

```
DECLARE
   total_amount NUMBER;
BEGIN
  SELECT SUM(amount) INTO total_amount FROM payment;
   DBMS_OUTPUT.PUT_LINE('Total amount: ' || total_amount);
END;
/
```

Output:

```
SQL> DECLARE
  2      total_amount NUMBER;
  3  BEGIN
  4      SELECT SUM(amount) INTO total_amount FROM payment;
  5      DBMS_OUTPUT.PUT_LINE('Total amount: ' || total_amount);
  6  END;
  7  /
Total amount: 36200

PL/SQL procedure successfully completed.
```

3.Append '@gmail.com' in the email of the Employee Table:

```
SQL> alter table emp_table modify email_id varchar(20);

Table altered.

SQL> DECLARE
  2    v_emp_id emp_table.emp_id%TYPE;
  3    v_email emp_table.email_id%TYPE;
  4  BEGIN
  5    FOR emp_rec IN (SELECT emp_id, email_id FROM emp_table) LOOP
  6      v_email := emp_rec.email_id || '@gmail.com';
  7      UPDATE emp_table
  8      SET email_id = v_email
  9      WHERE emp_id = emp_rec.emp_id;
 10    END LOOP;
 11    COMMIT;
 12  END;
 13  /

PL/SQL procedure successfully completed.

SQL> select * from Emp_table;

EMP ENAME EMAIL_ID                  PHONE     SALARY SHI
--- ----- -------------------- ---------- ---------- ---
1   rithi rit@gmail.com        7334512309      60000 day
2   harin hari@gmail.com       9003402381      50000 day
3   Rajii raji@gmail.com       6878900123      35000 nyt
4   Nivas niva@gmail.com       9345542103      30000 day
```

4.Increment the salary of the Employee Table :

```
SQL> select * from emp_table;

EMP ENAME EMAIL_ID        PHONE     SALARY SHI
--- ----- ---------- ---------- ---------- ---
1   rithi rit        7334512309      50000 day
2   harin hari       9003402381      45000 day
3   Rajii raji       6878900123      30000 nyt
4   Nivas niva       9345542103      25000 day

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2    v_emp_id emp_table.emp_id%TYPE;
  3    v_ename emp_table.ename%TYPE;
  4    v_salary emp_table.salary%TYPE;
  5  BEGIN
  6    FOR emp_rec IN (SELECT emp_id, ename, salary FROM emp_table) LOOP
  7      IF emp_rec.salary > 1000 AND emp_rec.salary < 50000 THEN
  8        UPDATE emp_table SET salary = emp_rec.salary + 5000 WHERE emp_id = emp_rec.emp_id;
  9      ELSIF emp_rec.salary >= 50000 AND emp_rec.salary < 100000 THEN
 10        UPDATE emp_table SET salary = emp_rec.salary + 10000 WHERE emp_id = emp_rec.emp_id;
 11      END IF;
 12    END LOOP;
 13    COMMIT;
 14    FOR emp_rec IN (SELECT emp_id, ename, salary FROM emp_table) LOOP
 15      DBMS_OUTPUT.PUT_LINE('Emp ID: ' || emp_rec.emp_id || ', Ename: ' || emp_rec.ename || ', Salary: ' || emp_rec.salary);
 16    END LOOP;
 17  END;
 18  /
Emp ID: 1, Ename: rithi, Salary: 60000
Emp ID: 2, Ename: harin, Salary: 50000
Emp ID: 3, Ename: Rajii, Salary: 35000
Emp ID: 4, Ename: Nivas, Salary: 30000

PL/SQL procedure successfully completed.
```

5.Display vehicle model with highest and lowest mileage

```
SQL> DECLARE
         total_vehicles NUMBER;
         avg_mileage NUMBER;
         max_mileage_vehicle VARCHAR2(100);
         min_mileage_vehicle VARCHAR2(100);
         max_mileage NUMBER := 0;
         min_mileage NUMBER := 999999;         -- Set to a high value initially
     BEGIN
         -- Get total number of vehicles
         SELECT COUNT(*) INTO total_vehicles FROM vehicles;

         -- Get average mileage of all vehicles
         SELECT AVG(mileage) INTO avg_mileage FROM vehicles;

         -- Get vehicle with highest mileage
         SELECT model || ' (' || manufacture || ')' INTO max_mileage_vehicle
         FROM vehicles
         WHERE mileage = (SELECT MAX(mileage) FROM vehicles);

         -- Get vehicle with lowest mileage
         SELECT model || ' (' || manufacture || ')' INTO min_mileage_vehicle
         FROM vehicles
         WHERE mileage = (SELECT MIN(mileage) FROM vehicles);

         DBMS_OUTPUT.PUT_LINE('Total number of vehicles: ' || total_vehicles);
         DBMS_OUTPUT.PUT_LINE('Average mileage of all vehicles: ' || avg_mileage);
         DBMS_OUTPUT.PUT_LINE('Vehicle with the highest mileage: ' || max_mileage_vehicle);
         DBMS_OUTPUT.PUT_LINE('Vehicle with the lowest mileage: ' || min_mileage_vehicle);
     END;
     /
Total number of vehicles: 8
Average mileage of all vehicles: 48.71428571428571428571428571428571428571
Vehicle with the highest mileage: mahi (2004)
Vehicle with the lowest mileage: mercedes (2014)

PL/SQL procedure successfully completed.
```

Result

Thus the PL/SQL block has been created and executed successfully