Exp No. 9 Date : 19-08-2019

PL/SQL AND SEQUENCE

AIM:

To study the basic pl/sql and sequence queries.

QUESTIONS:

1. To print the first ‘n’ prime numbers.

```
postgres=# CREATE OR REPLACE FUNCTION prime(n integer) RETURNS INTEGER AS $$
postgres$# DECLARE
postgres$#     i INTEGER;
postgres$#     j INTEGER;
postgres$#     flag INTEGER;
postgres$#     count INTEGER;
postgres$# BEGIN
postgres$#     IF n >= 1 THEN
postgres$#         RAISE NOTICE 'first % prime numbers are :', n;
postgres$#     END IF;
postgres$#     count := 1;
postgres$#     i := 2;
postgres$#     WHILE count <= n LOOP
postgres$#         flag = 1;
postgres$#
postgres$#         FOR j IN 2..i-1 LOOP
postgres$#             IF mod(i, j) = 0 THEN
postgres$#                 flag = 0;
postgres$#                 EXIT;
postgres$#             END IF;
postgres$#         END LOOP;
postgres$#
postgres$#         IF flag = 1 THEN
postgres$#             RAISE NOTICE '%', i;
postgres$#             count := count + 1;
postgres$#         END IF;
postgres$#         i := i + 1;
postgres$#     END LOOP;
postgres$# END;
postgres$# $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
postgres=# SELECT prime(7);
NOTICE:  first 7 prime numbers are :
NOTICE:  2
NOTICE:  3
NOTICE:  5
NOTICE:  7
NOTICE:  11
NOTICE:  13
NOTICE:  17
ERROR:  control reached end of function without RETURN
CONTEXT:  PL/pgSQL function prime(integer)
postgres=#
```

2. Display the Fibonacci series up to ‘n’ terms

```
postgres=# CREATE OR REPLACE FUNCTION fibonacci(n INTEGER) RETURNS INTEGER AS $$
postgres$#  DECLARE
postgres$#      first INTEGER :=  0;
postgres$#       second INTEGER := 1;
postgres$#       temp INTEGER;
postgres$#       k INTEGER := n;
postgres$#       i INTEGER;
postgres$#  BEGIN
postgres$#   RAISE NOTICE 'First % Fibonacci Series is', n;
postgres$#       RAISE NOTICE '%', first;
postgres$#       RAISE NOTICE '%', second;
postgres$#
postgres$#       FOR i IN 2..k-1
postgres$#       LOOP
postgres$#    temp := first + second;
postgres$#    first := second;
postgres$#    second := temp;
postgres$#            RAISE NOTICE '%', temp;
postgres$#       END LOOP;
postgres$#  END;
postgres$#  $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
postgres=# SELECT fibonacci(7);
NOTICE:  First 7 Fibonacci Series is
NOTICE:  0
NOTICE:  1
NOTICE:  1
NOTICE:  2
NOTICE:  3
NOTICE:  5
NOTICE:  8
ERROR:  control reached end of function without RETURN
CONTEXT:  PL/pgSQL function fibonacci(integer)
postgres=#
```

3. Create a table named student_grade with the given attributes:
roll, name ,mark1, mark2, mark3, grade. Read the roll, name and marks from the user. Calculate the grade of the student and insert a tuple into the table using PL/SQL.
(Grade= ‘PASS’ if AVG >40, Grade = ‘FAIL’ otherwise)

```
plsql_sequence=# CREATE TABLE student_grade(roll INT NOT NULL PRIMARY KEY, name VARCHAR(10) NOT NULL,
plsql_sequence(#                            mark1 INT NOT NULL, mark2 INT NOT NULL, mark3 INT NOT NULL,
plsql_sequence(#                            grade VARCHAR(10) );
CREATE TABLE
plsql_sequence=# INSERT INTO student_grade(roll, name, mark1, mark2, mark3)
plsql_sequence-# VALUES (1, 'anu', 50, 45, 48), (2, 'manu', 50, 50, 50), (3, 'vinu', 35, 40, 40);
INSERT 0 3
plsql_sequence=# SELECT * FROM student_grade;
 roll | name | mark1 | mark2 | mark3 | grade
------+------+-------+-------+-------+-------
    1 | anu  |    50 |    45 |    48 |
    2 | manu |    50 |    50 |    50 |
    3 | vinu |    35 |    40 |    40 |
(3 rows)

plsql_sequence=# CREATE OR REPLACE FUNCTION calculate_grade() RETURNS VOID AS $$
plsql_sequence$#     UPDATE student_grade
plsql_sequence$#         SET grade = CASE
plsql_sequence$#                         WHEN (mark1 + mark2 + mark3) / 3 > 40 THEN 'Pass'
plsql_sequence$#                         ELSE 'Fail'
plsql_sequence$#                     end;
plsql_sequence$# $$ LANGUAGE SQL;
CREATE FUNCTION
plsql_sequence=# SELECT calculate_grade();
 calculate_grade
-----------------

(1 row)

plsql_sequence=# SELECT * FROM student_grade;
 roll | name | mark1 | mark2 | mark3 | grade
------+------+-------+-------+-------+-------
    1 | anu  |    50 |    45 |    48 | Pass
    2 | manu |    50 |    50 |    50 | Pass
    3 | vinu |    35 |    40 |    40 | Fail
(3 rows)

plsql_sequence=#
```

4. Create table circle_area (rad, area). For radius 5,10,15,20 &25., find the area and insert the corresponding values into the table by using loop structure in PL/SQL.

```
plsql_sequence=# CREATE TABLE circle_area(radius INT, area FLOAT);
CREATE TABLE
plsql_sequence=# /* Read step size and limit from user as function parameter */
plsql_sequence-# CREATE OR REPLACE FUNCTION area_calculation(step integer, lim integer) RETURNS VOID AS $$
plsql_sequence$# DECLARE
plsql_sequence$#     area INTEGER;
plsql_sequence$#     temp INTEGER := step;
plsql_sequence$# BEGIN
plsql_sequence$#     TRUNCATE TABLE circle_area;                        -- truncating the table
plsql_sequence$#     LOOP
plsql_sequence$#         area := CEIL(3.14 * temp * temp);              -- area calculation
plsql_sequence$#         INSERT INTO circle_area VALUES(temp, area);    -- add corresponding values into table
plsql_sequence$#         temp := temp + step;                           -- increasing next radius with step
plsql_sequence$#         lim := lim - 1;                                -- decrasing the limit
plsql_sequence$#         EXIT WHEN 0 >= lim;                            -- exit if limit reached
plsql_sequence$#     END LOOP;
plsql_sequence$# END
plsql_sequence$# $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
plsql_sequence=# SELECT area_calculation(5, 7);
 area_calculation
------------------

(1 row)

plsql_sequence=# SELECT * FROM circle_area;
 radius | area
--------+------
      5 |   79
     10 |  314
     15 |  707
     20 | 1256
     25 | 1963
     30 | 2826
     35 | 3847
(7 rows)

plsql_sequence=#
```

5. Use an array to store the names, marks of 10 students in a class. Using Loop structures in PL/SQL insert the ten tuples to a table named student

```
plsql_sequence=# CREATE TABLE student(name TEXT, mark INT);
CREATE TABLE
plsql_sequence=# CREATE OR REPLACE FUNCTION array_input() RETURNS VOID AS $$
plsql_sequence$#     DECLARE
plsql_sequence$#         i INTEGER;
plsql_sequence$#         name_array TEXT[] := '{"ARUN", "AMAL", "PETER", "JOSE", "ANNIE", "MARY", "JOSEPH", "MARK", "MIDHUN", "KEVIN"}';
plsql_sequence$#         mark_array INTEGER[] := '{25, 76, 43, 45, 67, 57, 97, 56, 89, 8}';
plsql_sequence$#
plsql_sequence$#     BEGIN
plsql_sequence$#         FOR i IN array_lower(name_array, 1) .. array_upper(name_array, 1) LOOP
plsql_sequence$#             INSERT INTO student(name, mark) VALUES (name_array[i], mark_array[i]);
plsql_sequence$#     END LOOP;
plsql_sequence$#     END
plsql_sequence$# $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
plsql_sequence=# SELECT array_input();
 array_input
-------------

(1 row)

plsql_sequence=# SELECT * FROM student;
  name  | mark
--------+------
 ARUN   |   25
 AMAL   |   76
 PETER  |   43
 JOSE   |   45
 ANNIE  |   67
 MARY   |   57
 JOSEPH |   97
 MARK   |   56
 MIDHUN |   89
 KEVIN  |    8
(10 rows)

plsql_sequence=#
```

6. Create a sequence using PL/SQL. Use this sequence to generate the primary key values for a table named class_cse with attributes roll,name and phone. Insert some tuples using PL/SQL programming.

```
plsql_sequence=# CREATE OR REPLACE FUNCTION my_sequence() RETURNS VOID AS $$
plsql_sequence$# BEGIN
plsql_sequence$#   CREATE TABLE class_cse(roll INT NOT NULL PRIMARY KEY, name VARCHAR(20), phone INT);
plsql_sequence$#   CREATE SEQUENCE roll_no START 1;
plsql_sequence$#   INSERT INTO class_cse VALUES(nextval('roll_no'), 'ARUN', 0482239091),
plsql_sequence$#                               (nextval('roll_no'), 'AMAL', 0484234562),
plsql_sequence$#                               (nextval('roll_no'), 'PETER', 048511234),
plsql_sequence$#                               (nextval('roll_no'), 'JOSE', 048943617),
plsql_sequence$#                               (nextval('roll_no'), 'ANNIE', 048123145);
plsql_sequence$# END
plsql_sequence$# $$ LANGUAGE PLPGSQL;
CREATE FUNCTION
plsql_sequence=# SELECT my_sequence();
 my_sequence
-------------

(1 row)

plsql_sequence=# SELECT * FROM class_cse;
 roll | name  |   phone
------+-------+-----------
    1 | ARUN  | 482239091
    2 | AMAL  | 484234562
    3 | PETER |  48511234
    4 | JOSE  |  48943617
    5 | ANNIE |  48123145
(5 rows)

plsql_sequence=#
```

RESULT:

The PL/SQL program was executed successfully and the output was obtained.