Department of Computer Science and Engineering

Islamic University of Technology (IUT)

A subsidiary organ of OIC

Lab Report 01

CSE 4508: RDBMS Lab

Name:Md. Abdullah Al Jubaer Gem
Student ID:210041226
Section:2B
Semester:5th
Academic Year:2023-2024

Date of Submission:23-09-2024

Task 1:

Write a block of PL/SQL code that checks whether the current year is the starting year of a new decade (years such as 2000, 2010, 2020) and prints either “Yes” or “no”. After this, it should print the current decade (e.g., for 2000 to 2009, print ‘The 2000s’, for 2010 to 2019, print ‘The 2010s’).

Solution:

```
DECLARE
  current_year NUMBER;
  start_of_decade NUMBER;
BEGIN
  -- Getting the current year
  current_year := EXTRACT(YEAR FROM SYSDATE);

  -- Checking if the current year is the starting year of a new decade
  IF MOD(current_year, 10) = 0 THEN
    DBMS_OUTPUT.PUT_LINE('Yes Current Year '|| TO_CHAR(current_year) || '
is the starting of a new decade');
  ELSE
    DBMS_OUTPUT.PUT_LINE('No Current Year '|| TO_CHAR(current_year) || '
is not the starting of a new decade');
  END IF;

  -- Calculating the start of the current decade
  start_of_decade := current_year - MOD(current_year, 10);

  DBMS_OUTPUT.PUT_LINE('The start of the current decade is the ' ||
TO_CHAR(start_of_decade) || 's');
END;
/
```

Task 2:

Write a PL/SQL procedure (or function) called prime_generator which takes only one input: s.The function will keep generating prime numbers, starting from 2, until the sum of all the prime numbers generated so far is less than or equal to s. For example, if s = 20, the output will be: 2, 3, 5, 7 (since 2 + 3 + 5 + 7 = 17. "11" is not included since that would make the sum greater than 20). Execute this function from a PL/SQL block.

Solution:

```
SET SERVEROUTPUT ON;

DECLARE
  s NUMBER := 20;
  primes VARCHAR2(4000);

  FUNCTION prime_generator(s IN NUMBER)
  RETURN VARCHAR2 IS
    sum_of_primes NUMBER := 0;
    current_number NUMBER := 2;
    result VARCHAR2(4000) := '';

    -- Function to check if a number is prime
    FUNCTION is_prime(n IN NUMBER) RETURN BOOLEAN IS
      i NUMBER;
    BEGIN
      IF n < 2 THEN
        RETURN FALSE;
      END IF;
      FOR i IN 2..FLOOR(SQRT(n)) LOOP
        IF n MOD i = 0 THEN
          RETURN FALSE;
        END IF;
      END LOOP;
      RETURN TRUE;
    END is_prime;
  BEGIN
    -- Looping to find primes and sum them
    WHILE sum_of_primes + current_number <= s LOOP
      IF is_prime(current_number) THEN
        -- Adding the prime number to the result string
        result := result || current_number || ', ';
        sum_of_primes := sum_of_primes + current_number;
      END IF;
      current_number := current_number + 1;
    END LOOP;


    RETURN result;
  END prime_generator;

BEGIN
  -- Calling the prime_generator function and displaying the result
  primes := prime_generator(s);
  DBMS_OUTPUT.PUT_LINE('Primes: ' || primes);
END;
/
```

Task 3:

As a course teacher, you are asked to calculate the final marks of your students based on various assessments. The final marks should be calculated based on the following criteria:

1. Attendance: 10%

2. Quiz: 15%

3. Mid: 25%

4. Final: 50%

Task 3.1

Design a student table with the required columns and insert 5 dummy samples.

Task 3.2

Write a PL/SQL procedure that calculates the total marks based on the aforementioned criteria.

Task 3.3

Write another PL/SQL procedure that calculates grade based on the total marks. Use this simplified grading scheme:

A : $\geq$ 80%, B : $\geq$ 70%, C : $\geq$ 60%, D : $\geq$ 40%, F : < 40%

Solution:

```
SET SERVEROUTPUT ON;

-- Dropping procedures if they exist
BEGIN
  EXECUTE IMMEDIATE 'DROP PROCEDURE calculate_total_marks';
  EXECUTE IMMEDIATE 'DROP PROCEDURE calculate_grade';
  EXECUTE IMMEDIATE 'DROP TABLE students';
EXCEPTION
  WHEN OTHERS THEN
    NULL;
END;
/


-- Task 3.1
CREATE TABLE students (
  student_id NUMBER PRIMARY KEY,
  student_name VARCHAR2(100),
  attendance NUMBER(5,2),
  quiz NUMBER(5,2),
  mid_term NUMBER(5,2),
  final_exam NUMBER(5,2)
);

INSERT INTO students (student_id, student_name, attendance, quiz,
mid_term, final_exam)
VALUES (1, 'A', 8.5, 12, 20, 40);

INSERT INTO students (student_id, student_name, attendance, quiz,
mid_term, final_exam)
VALUES (2, 'B', 9, 13, 23, 45);

INSERT INTO students (student_id, student_name, attendance, quiz,
mid_term, final_exam)
VALUES (3, 'C', 7.5, 10, 19, 37);
```

```
INSERT INTO students (student_id, student_name, attendance, quiz,
mid_term, final_exam)
VALUES (4, 'D', 9.5, 14, 24, 47);

INSERT INTO students (student_id, student_name, attendance, quiz,
mid_term, final_exam)
VALUES (5, 'E', 8, 11, 22, 43);

COMMIT;

-- Task 3.2
CREATE OR REPLACE PROCEDURE calculate_total_marks IS
  total_marks NUMBER;
  attendance NUMBER;
  quiz NUMBER;
  mid_term NUMBER;
  final_exam NUMBER;
BEGIN
  FOR id IN 1..5 LOOP
    SELECT attendance, quiz, mid_term, final_exam
    INTO attendance, quiz, mid_term, final_exam
    FROM students
    WHERE student_id = id;

    -- Calculating total marks
    total_marks := (attendance * 0.10) +
                   (quiz * 0.15) +
                   (mid_term * 0.25) +
                   (final_exam * 0.50);

    -- Displaying the student's total marks
    DBMS_OUTPUT.PUT_LINE('Student ID: ' || id || ' Total Marks: ' ||
total_marks);
  END LOOP;
END;
/

-- Task 3.3
CREATE OR REPLACE PROCEDURE calculate_grade IS
  total_marks NUMBER;
  attendance NUMBER;
  quiz NUMBER;
  mid_term NUMBER;
  final_exam NUMBER;
  grade CHAR(1);
BEGIN
  FOR id IN 1..5 LOOP

    SELECT attendance, quiz, mid_term, final_exam
    INTO attendance, quiz, mid_term, final_exam
    FROM students
    WHERE student_id = id;

    -- Calculating total marks
    total_marks := (attendance * 0.10) +
                   (quiz * 0.15) +
                   (mid_term * 0.25) +
                   (final_exam * 0.50);

    -- Determining the grade
    IF total_marks >= 80 THEN
      grade := 'A';
    ELSIF total_marks >= 70 THEN
      grade := 'B';
    ELSIF total_marks >= 60 THEN
      grade := 'C';
    ELSIF total_marks >= 40 THEN
      grade := 'D';
    ELSE
      grade := 'F';
    END IF;

    -- Displaying the student's grade
    DBMS_OUTPUT.PUT_LINE('Student ID: ' || id || ' Grade: ' || grade);
  END LOOP;
END;
/

-- Testing
BEGIN
```

```
  calculate_total_marks;
END;
/

BEGIN
  calculate_grade;
END;
/
```

Challenges:

- Had to go through the documentation more than once to understand how PL/SQL works clearly. As this course was done a few months ago.
- Had some compilation problems which took some time to debug.
- Most of the time I forgot to add '/' which resulted in errors
- Had to drop the procedures and tables as running them to test multiple times led to errors
- Had to look up different normal functionalities like looping , string concatenation and stuffs like that.

References:

https://stackoverflow.com/questions/12801533/how-to-use-commands-like-drop-table-etc-within-a-stored-procedure (For dropping procedures)

N.B: Some other references were also used to look up multiple functionalities like looping and such but those were not kept tracked of