PL/SQL ASSIGNMENT 1

1. Execute the command: *set serveroutput on;* What does this command do? Google it on the Internet.

Answer: *set serveroutput on;* is a command in SQL, particularly used in Oracle SQL, to enable the output of messages generated by PL/SQL code. When this command is executed, it allows PL/SQL code to display messages using the `DBMS_OUTPUT.PUT_LINE` procedure. This is often used for debugging or informational purposes within stored procedures, triggers, or anonymous blocks.

2. Type the following program and execute it. Can you tell from the result what the command DBMS_OUTPUT.PUT_LINE does?

```
--PL/SQL program to display current date

DECLARE

    today_date DATE;

BEGIN

    today_date := SYSDATE;

    DBMS_OUTPUT.PUT_LINE('Today''s date is ');

    DBMS_OUTPUT.PUT_LINE(today_date);

END;
```

Answer: This program will display the current date using the 'DBMS_OUTPUT.PUT_LINE' procedure in Oracle SQL. When executed, it will output the current date. The 'DBMS_OUTPUT.PUT_LINE' procedure is used to print the provided message or value to the output buffer. In this case, it's printing the message 'Today's date is' followed by the actual date stored in the variable 'today_date'.

Here's the screenshot of the output of this program:

3. Before you begin to work on the following questions, first download createDB.sql from the course website and execute it. Read the following program and figure out what it does. Then type it and execute it to verify if you are right.

```
--PL/SQL program to get grade statistics
DECLARE
        na integer :=0;
BEGIN
        Select count(*)
        into na
        from gradeReport
        where grade='A';
        if (na > 0) then
        DBMS_OUTPUT.PUT_LINE('there are total ' || na || ' A''s');
        else
        DBMS_OUTPUT.PUT_LINE('No student makes an A');
        end if;
END;
```

Answer: The provided PL/SQL program is aimed at retrieving statistics regarding the grades of students from the **GradeReport** table.

Here's the breakdown of the program:

- It declares a variable **na** of type integer and initializes it to 0.
- It begins the execution block of the PL/SQL program.
- It performs a SQL query to count the number of rows in the **GradeReport** table where the grade is 'A'. This count is stored in the variable **na**.
- It checks if the count (**na**) is greater than 0.
- If there are students with grade 'A', it prints a message indicating the total count of 'A's. Otherwise, it prints a message indicating that no students achieved an 'A'.

Here’s the screenshot of the output of this program:

4. Revise the above program, so that it will output the statistics for the grades A, B, C, D, and F.

Answer: To revise the program to output statistics for grades A, B, C, D, and F, we need to modify the SQL query to count the occurrences of each grade separately. Then, we can output the statistics for each grade accordingly. Here's the revised program:

```
DECLARE

        na INTEGER := 0;
        nb INTEGER := 0;
        nc INTEGER := 0;
        nd INTEGER := 0;
        nf INTEGER := 0;


BEGIN
   SELECT COUNT(*) INTO na FROM GradeReport WHERE Grade = 'A';
   SELECT COUNT(*) INTO nb FROM GradeReport WHERE Grade = 'B';
   SELECT COUNT(*) INTO nc FROM GradeReport WHERE Grade = 'C';
   SELECT COUNT(*) INTO nd FROM GradeReport WHERE Grade = 'D';
   SELECT COUNT(*) INTO nf FROM GradeReport WHERE Grade = 'F';


   IF (na > 0) THEN
      DBMS_OUTPUT.PUT_LINE('Number of A"s: ' || na);
   ELSE
      DBMS_OUTPUT.PUT_LINE('No student has achieved an A');
   END IF;


   IF (nb > 0) THEN
      DBMS_OUTPUT.PUT_LINE('Number of B"s: ' || nb);
   ELSE
      DBMS_OUTPUT.PUT_LINE('No student has achieved a B');
   END IF;


   IF (nc > 0) THEN
      DBMS_OUTPUT.PUT_LINE('Number of C"s: ' || nc);
```

```
    ELSE
        DBMS_OUTPUT.PUT_LINE('No student has achieved a C');
    END IF;

    IF (nd > 0) THEN
        DBMS_OUTPUT.PUT_LINE('Number of D"s: ' || nd);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No student has achieved a D');
    END IF;

    IF (nf > 0) THEN
        DBMS_OUTPUT.PUT_LINE('Number of F"s: ' || nf);
    ELSE
        DBMS_OUTPUT.PUT_LINE('No student has achieved an F');
    END IF;
END;
```

This revised program counts the occurrences of grades A, B, C, D, and F separately using individual SQL queries. Then, it outputs the statistics for each grade. If there are no students with a particular grade, it outputs a corresponding message indicating that no student has achieved that grade. Here's the screenshot of the output of this program:

PL/SQL ASSIGNMENT 2

1. Re-do question 4 of the PL/SQL Assignment 1 by using loop control. Hint: consider using VARRAY. For example,

```
DECLARE
        type gradeCount is varray(5) of integer;
        gdcnt gradeCount;
        type grade is varray(5) of char(1);
        gd grade;
        …
BEGIN
        gdcnt := gradeCount(0,0,0,0,0);
        gd := grade('A', 'B', 'C', 'D', 'F');
        …
END;
```

Answer: To revise the program using loop control and VARRAY, I iterated through the grades and dynamically counted the occurrences of each grade. Here's the revised program:

```
DECLARE
        TYPE gradeCount IS VARRAY(5) OF INTEGER;
        gdcnt gradeCount := gradeCount(0, 0, 0, 0, 0);
        TYPE grade IS VARRAY(5) OF CHAR(1);
        gd grade := grade('A', 'B', 'C', 'D', 'F');
        i INTEGER;
BEGIN
    FOR i IN 1..gd.COUNT LOOP
        SELECT COUNT(*) INTO gdcnt(i) FROM GradeReport WHERE Grade = gd(i);
    END LOOP;
    FOR i IN 1..gd.COUNT LOOP
        IF (gdcnt(i) > 0) THEN
            DBMS_OUTPUT.PUT_LINE('Number of ' || gd(i) || '''s: ' || gdcnt(i));
        ELSE
            DBMS_OUTPUT.PUT_LINE('No student has achieved a ' || gd(i));
        END IF;
    END LOOP;
END;
```

This revised program uses loop control to iterate through the grades specified in the VARRAY **gd**. Inside the loop, it dynamically counts the occurrences of each grade using SQL queries.

Then, another loop is used to output the statistics for each grade. If there are no students with a particular grade, it outputs a corresponding message indicating that no student has achieved that grade. Here's the screenshot of the output of the revised program:

2. Type the following anonymous block and execute it.

```
DECLARE
        sm binary_integer :=0;
        i binary_integer :=0;
BEGIN
        loop
                i := i + 1;
                if i > 10 then
                        exit;
                end if;
                sm := sm + i;
        end loop;
        dbms_output.put_line('sum= ' || sm || '.');
END;
```

Answer: Here's the screenshot of the output of the above anonymous block:

3. Revise the above program by using for loop and while loop respectively and execute them. The revised program should have the same result as the above program.

Answer: Here are the revised versions of the program using both **FOR** loop and **WHILE** loop:

Using FOR loop:

DECLARE

```
    sm BINARY_INTEGER := 0;
BEGIN
    FOR i IN 1..10 LOOP
        sm := sm + i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('sum= ' || sm || '.');
END;
```

Using WHILE loop:

```
DECLARE
    sm BINARY_INTEGER := 0;
    i BINARY_INTEGER := 0;
BEGIN
    WHILE i < 10 LOOP
        i := i + 1;
        sm := sm + i;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('sum= ' || sm || '.');
END;
```

PL/SQL Assignment 3

1. Write a stored procedure named as *update_superSSN*. The procedure takes two input parameters: *empSSN* (as an employee's SSN) and *new_superSSN* (as a new supervisor's SSN). The procedure will update the supervisor's SSN to *new_superSSN* for the given empSSN. The template for defining such a procedure is given as below:

create procedure update_superSSN(empSSN in employee.SSN%type, new_superSSN in employee.superssn%type) as

... //procedure code

Answer:

CREATE OR REPLACE **PROCEDURE** update_superSSN(empSSN **IN** employee.SSN%TYPE, new_superSSN **IN** employee.super_ssn%TYPE) **AS**

BEGIN

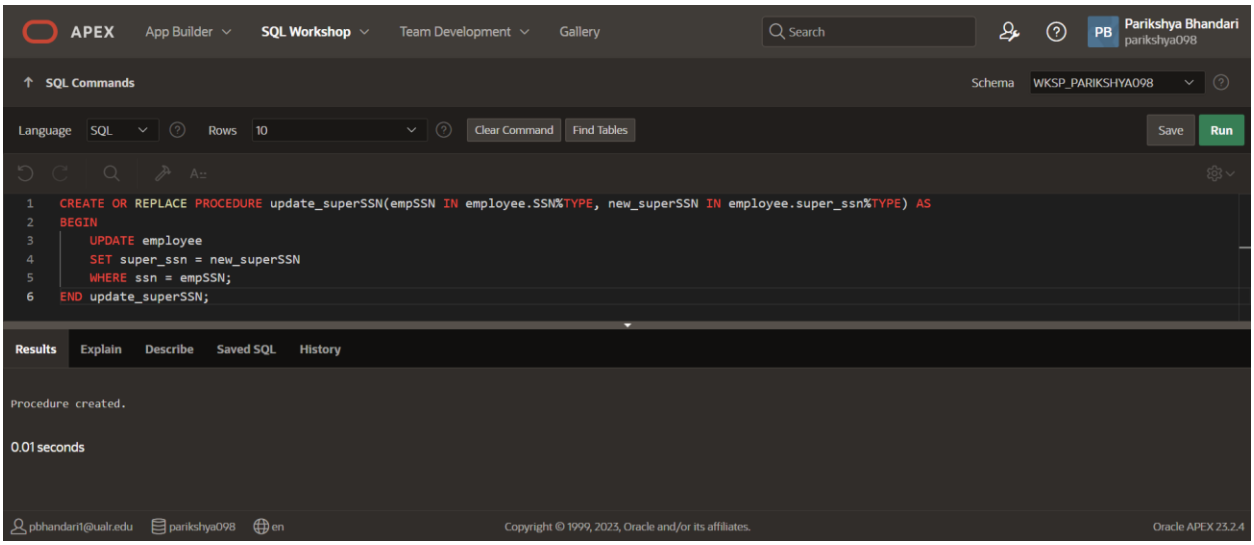
UPDATE employee

SET super_ssn = new_superSSN

WHERE ssn = empSSN;

END update_superSSN;

The above given stored procedure will update an employee's supervisor SSN (**super_ssn**) with a new value (**new_superSSN**) for a specific employee identified by their SSN (**empSSN**).



2. Write a stored function named as NumOfEmployee_ByDept. The function has one input parameter deptname (i.e., department name) and returns the number of employees working in that department.

Once the stored function is compiled without any errors, you can run the following SQL query:
Select dname, NumOfEmployee_byDept(dname)
From department;

Answer:

CREATE OR REPLACE **FUNCTION** NumOfEmployee_ByDept(deptname **IN** department.Dname%TYPE) **RETURN** NUMBER **AS**

num_employees NUMBER(5);

BEGIN

SELECT COUNT(*)

INTO num_employees

FROM employee e

JOIN department d **ON** e.dno = d.Dnumber

WHERE d.Dname = deptname;

RETURN num_employees;

END NumOfEmployee_ByDept;

The above given stored function will return the number of employees working in a specific department, identified by the department name (**deptname**). After compiling the function without any errors, I used the provided SQL query to retrieve the number of employees for each department by department name.