**PL/SQL Basics:**

**Assignment**

**Harshit Kushmakar| 16896**

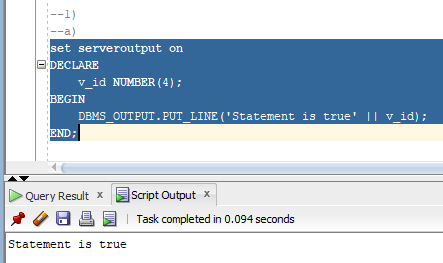
**Practice 1:**

**Evaluate each of the following declarations.**

**Determine which of them are not legal and explain why?**

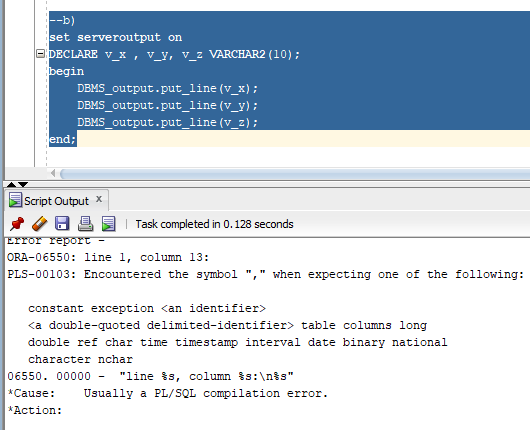
1. **DECLARE v\_id NUMBER(4);**

Ans:Valid

****

**b. DECLARE v\_x , v\_y, v\_z VARCHAR2(10);**

**Ans:** Not Valid - we can’t declare multiple variables at once.

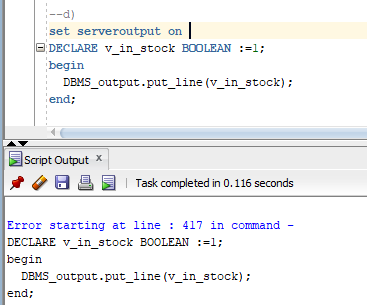
****

**c.DECLARE v\_birthdate DATE NOT NULL;**

ANS: Not Valid- we must assign a value.

**d. DECLARE v\_in\_stock BOOLEAN :=1;**

ANS: Not Valid - we have to write BOOLEAN := True;



**2) What will be the output of the following program?**

**DECLARE**

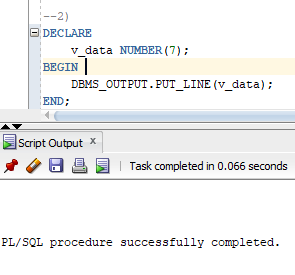
**v\_data NUMBER(7);**

**BEGIN**

**DBMS\_OUTPUT.PUT\_LINE(v\_data);**

**END;**

Ans: It will not give any output as we are not assigning any value in v\_data

****

**Practice 3:**

**DECLARE**

**v\_weight NUMBER (3):=600;**

**v\_message VARCHAR2(255):=‘Product 10012’;**

**BEGIN**

**DECLARE**

**v\_weight NUMBER(3) :=1;**

**v\_message VARCHAR2 (25):=‘Product 11001’;**

**v\_new\_locn VARCHAR2(25):=‘Europe’;**

**BEGIN**

**v\_weight := v\_weight +1;**

**v\_new\_locn:=‘Western ‘ ||v\_new\_locn;**

**-- Point 1**

**END;**

**v\_weight:=v\_weight + 1;**

**v\_message:=v\_message|| ‘ is in stock’;**

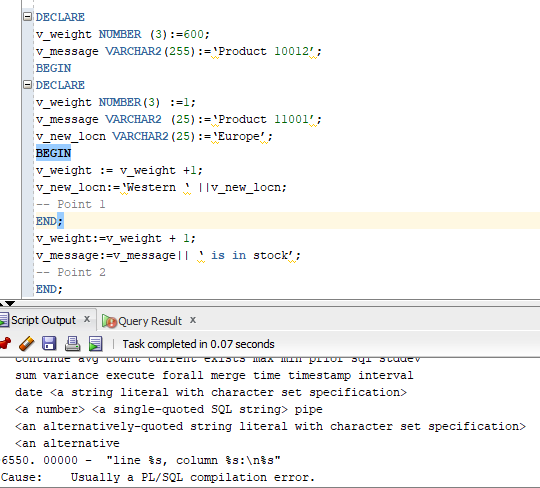
**-- Point 2**

**END;**

**2/3**

**Consider the above PL/SQL code. What will be the values of the**

**variables v\_weight, v\_message and v\_new\_locn at point 1 and point 2.**

****

**Practice 4:**

**Write a PL/SQL block that accepts values of two non zero numbers from user. The block performs the following operation ( first\_number/second\_number + second\_number). The result of the operation should be stored in a PL/SQL variable and also displayed on the screen.**

set serveroutput on

DECLARE

n1 number(2);

n2 number(2);

res number(2);

BEGIN

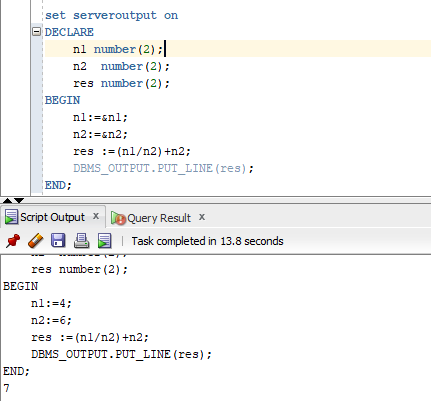
n1:=&n1;

n2:=&n2;

res :=(n1/n2)+n2;

DBMS\_OUTPUT.PUT\_LINE(res);

END;



**Q5:**

**Declare two SQL\* plus variables named MAX\_SALARY and**

**MIN\_SALARY that are of data type NUMBER.**

**Write a PL/SQL block that accepts deptno value from a user, selects**

**the maximum salary and minimum salary paid in the department,**

**from the EMP table and stores the corresponding values in**

**MAX\_SALARY and MIN\_SALARY respectively.**

**Use appropriate SQL \* plus command to see the modified values of**

**MAX\_SALARY and MIN\_SALARY**

**set serveroutput on**

DECLARE

max\_sal number;

min\_sal number;

BEGIN

SELECT max(salary) into max\_sal from employee1 where depid=4;

SELECT min(salary) into min\_sal from employee1 where depid=4;

DBMS\_OUTPUT.Put\_line(max\_sal || ' '|| min\_sal);

END;

**Practice 6:**

**Write a PL/SQL block that accepts employee number from a user and**

**retrieves the salary for the employee from the EMP table. It**

**determines the salary class as per the following criteria and displays**

**the salary and salary class on the screen**

**Criteria for deciding salary class:**

**•If the salary is less than 2500, then it comes under the class ‘LOW’**

**•If the salary is greater than or equal to 2500 and less than 5000,**

**then it comes under class ‘MEDIUM’.**

**•If the salary is greater than or equal to 5000, then it comes under**

**class ‘HIGH’**

set serveroutput on

declare

max\_sal number;

min\_sal number;

begin

select max(salary) into max\_sal from employee\_Kushmakar where depid=4;

select min(salary) into min\_sal from employee\_Kushmakar where depid=4;

DBMS\_OUTPUT.Put\_line(max\_sal || ' '|| min\_sal);

end;

**Practice 7:**

**Write a PL/SQL block that accepts an integer value between 1 and 12 from a user and displays the name of the corresponding month. If the number input is not in the range +1 to +12 then the block should display the message “Invalid Month” on the screen**

**3/3.**

SET SERVEROUTPUT ON

DECLARE

VAL NUMBER := '&V';

BEGIN

IF (VAL = 1) THEN

DBMS\_OUTPUT.PUT\_LINE('JANUARY');

ELSIF(VAL = 2) THEN

DBMS\_OUTPUT.PUT\_LINE('FEB');

ELSIF(VAL = 3) THEN

DBMS\_OUTPUT.PUT\_LINE('MARCH');

ELSIF(VAL = 4) THEN

DBMS\_OUTPUT.PUT\_LINE('APRIL');

ELSIF(VAL = 5) THEN

DBMS\_OUTPUT.PUT\_LINE('MAY');

ELSIF(VAL = 6) THEN

DBMS\_OUTPUT.PUT\_LINE('JUNE');

ELSIF(VAL = 7) THEN

DBMS\_OUTPUT.PUT\_LINE('JULY');

ELSIF(VAL = 8) THEN

DBMS\_OUTPUT.PUT\_LINE('AUGUST');

ELSIF(VAL = 9) THEN

DBMS\_OUTPUT.PUT\_LINE('SEPTEMBER');

ELSIF(VAL = 10) THEN

DBMS\_OUTPUT.PUT\_LINE('OCTOBER');

ELSIF(VAL = 11) THEN

DBMS\_OUTPUT.PUT\_LINE('NOV');

ELSIF(VAL = 12) THEN

DBMS\_OUTPUT.PUT\_LINE('DEC');

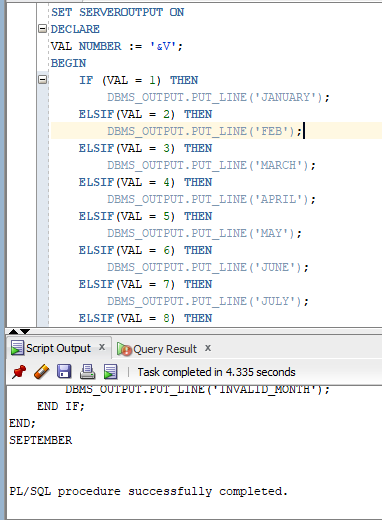
ELSE

DBMS\_OUTPUT.PUT\_LINE('INVALID\_MONTH');

END IF;

END;

/



**8(a): Write a PL/SQL block that accepts a positive number from a user and displays its factorial on the screen.**

DECLARE

FACT NUMBER := 1;

N NUMBER := &n;

BEGIN

WHILE N>0 LOOP

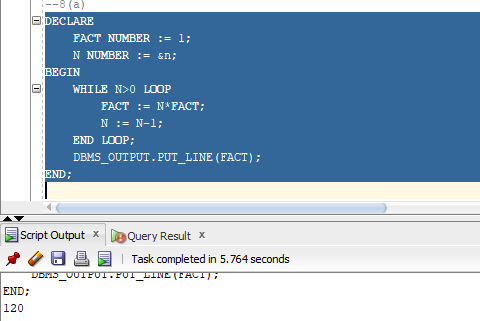
FACT := N\*FACT;

N := N-1;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(FACT);

END;



**8(b): Write a PL/SQL block that accepts a positive number ‘n’ from a user and displays a Fibonacci series of ‘n’ numbers.**

declare

first number := 0;

second number := 1;

temp number;

n number := &n;

i number;

begin

dbms\_output.put\_line('Series:');

dbms\_output.put\_line(first);

dbms\_output.put\_line(second);

for i in 2..n loop

temp:=first+second;

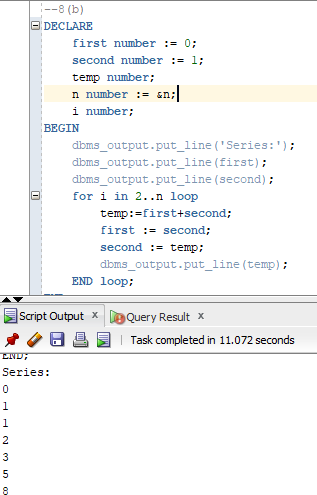
first := second;

second := temp;

dbms\_output.put\_line(temp);

end loop;

end;



**8c):Write a PL/SQL block that accepts a positive number ‘n’ from a user and displays a Fibonacci series whose last number is the largest integer lesser than or equal to ‘n’.**

DECLARE

I NUMBER;

LAST NUMBER := '&LAST';

I1 NUMBER := 0;

I2 NUMBER := 1;

FIB NUMBER;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(I1);

DBMS\_OUTPUT.PUT\_LINE(I2);

FOR I IN 2..LAST LOOP

FIB := I1 + I2;

IF FIB > LAST THEN

EXIT;

END IF;

DBMS\_OUTPUT.PUT\_LINE(FIB);

I1 := I2;

I2 := FIB;

END LOOP;

END;

/

Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application

Description automatically generated

**8d) Write a PL/SQL block that accepts a positive number ‘n’ and displays whether that number is a Prime number or not.**

set serveroutput on

declare

count\_factor number := 0;

enter\_num number;

begin

enter\_num := &enter\_num;

for i in 1..enter\_num loop

if(mod(enter\_num,i)=0)

then count\_factor:=count\_factor+1;

end if;

end loop;

if(count\_factor=2) then dbms\_output.put\_line('Is Prime');

else dbms\_output.put\_line('Is Not Prime');

end if;

end;

Graphical user interface, text, application, email

Description automatically generated

**8e) Write a PL/SQL block that accepts a positive number ‘n’ and displays all the prime numbers lesser than the given number ‘n’.**

set serveroutput on

declare

enter\_num number;

count\_num number;

j number;

begin

enter\_num := &enter\_num;

for i in 2..enter\_num-1 loop

count\_num := 0;

for j in 1..i loop

if mod(i,j)=0 then count\_num:=count\_num+1;

end if;

end loop;

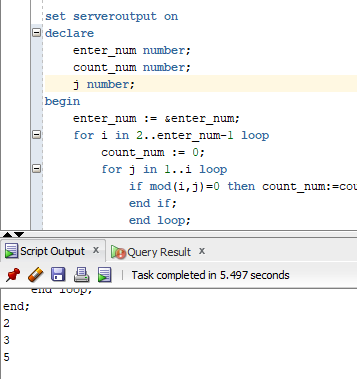
if count\_num<=2 then

dbms\_output.put\_line(i);

end if;

end loop;

end;



**Practice 9:**

**Write a PL/SQL block that accepts employee number from a user. Declare a PL/SQL record or a composite variable to store the employee number, employee name, department number and the department name of the employee. Retrieve the values of these columns for the employee from the EMP and DEPT tables and display the employee name and the corresponding department name on the screen.**

**Practice 10:**

**Write a PL/SQL block to add a department row in the DEPT table with the following values for columns**

**a. The block retrieves the maximum value of deptno from the dept table and adds 1 to it to generate the value of deptno.**

**b. Dname value should be ‘Education’**

**c. Loc value should be NULL**