PL/SQL Basics:

Assignment

Harshit Kushmakar| 16896

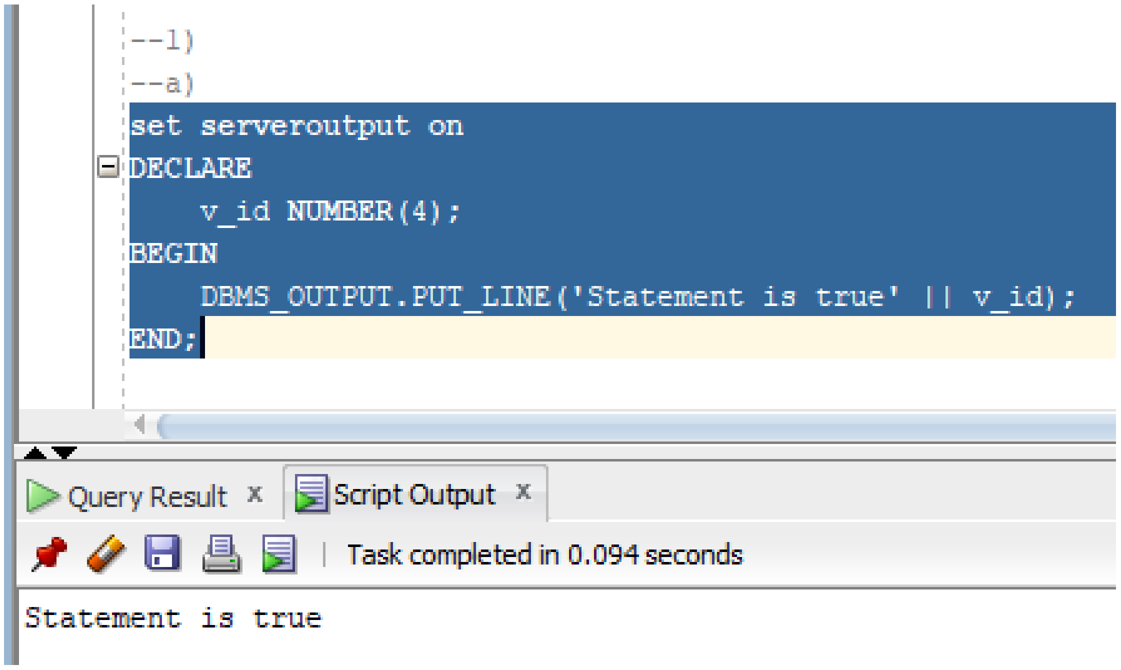
Practice 1:

Evaluate each of the following declarations.

Determine which of them are not legal and explain why?

a. DECLARE v_id NUMBER(4);

Ans:Valid

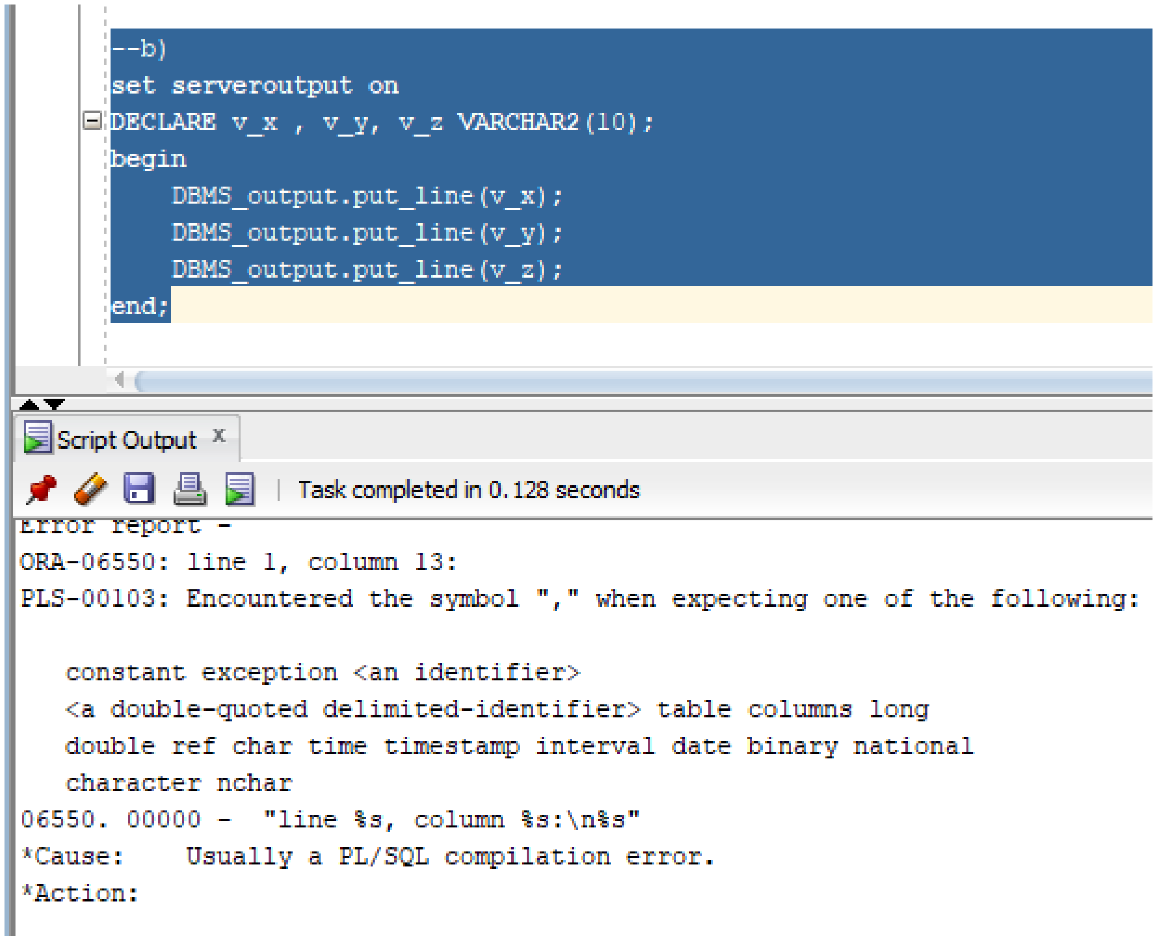


```
--1)
--a)
set serveroutput on
DECLARE
    v_id NUMBER(4);
BEGIN
    DBMS_OUTPUT.PUT_LINE('Statement is true' || v_id);
END;
```

```
Statement is true
```

b. DECLARE v_x , v_y, v_z VARCHAR2(10);

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



```
--b)
set serveroutput on
DECLARE v_x , v_y, v_z VARCHAR2(10);
begin
    DBMS_output.put_line(v_x);
    DBMS_output.put_line(v_y);
    DBMS_output.put_line(v_z);
end;
```

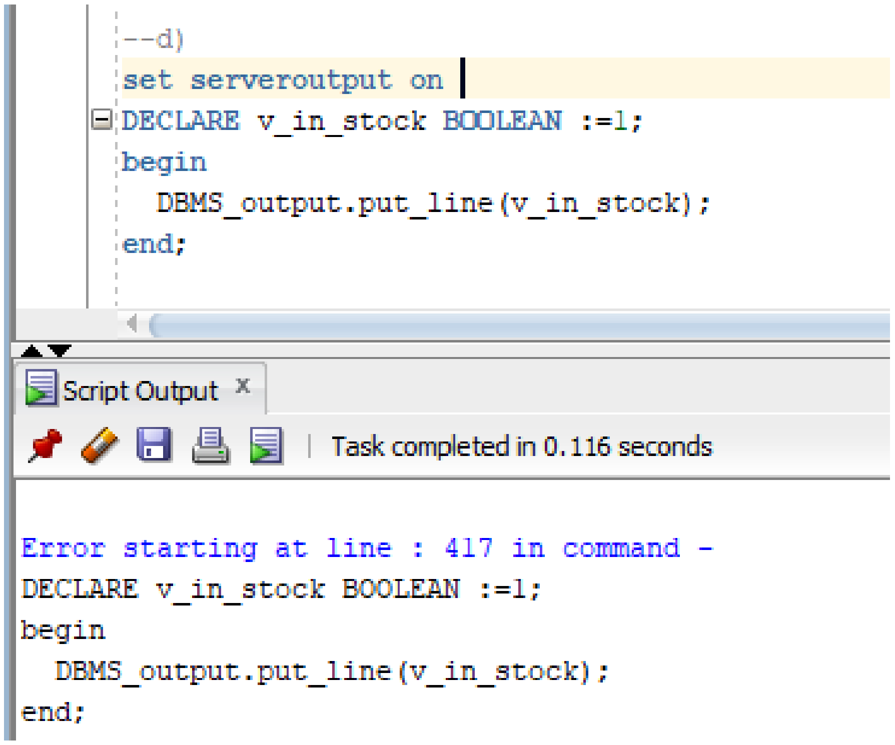
```
Error report -
ORA-06550: line 1, column 13:
PLS-00103: Encountered the symbol "," when expecting one of the following:

   constant exception <an identifier>
   <a double-quoted delimited-identifier> table columns long
   double ref char time timestamp interval date binary national
   character nchar
06550. 00000 -  "line %s, column %s:\n%s"
*Cause:    Usually a PL/SQL compilation error.
*Action:
```

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;



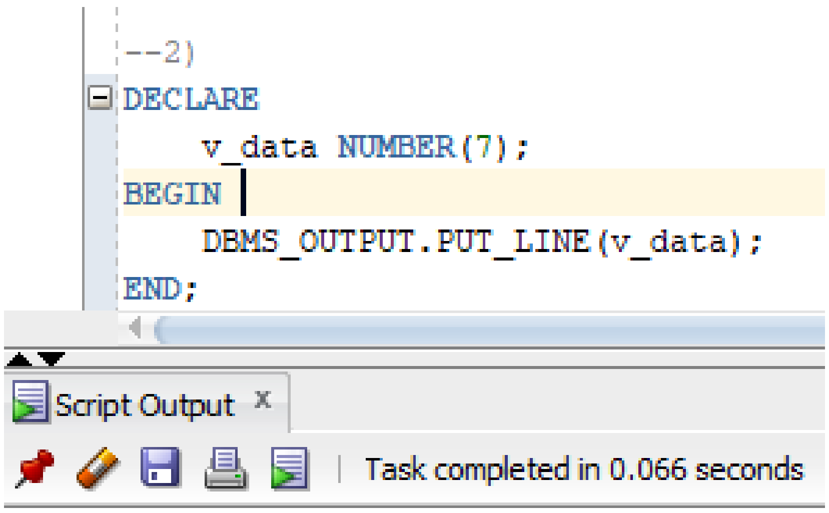
```
--d)
set serveroutput on
DECLARE v_in_stock BOOLEAN :=1;
begin
  DBMS_output.put_line(v_in_stock);
end;
```

```
Task completed in 0.116 seconds

Error starting at line : 417 in command -
DECLARE v_in_stock BOOLEAN :=1;
begin
  DBMS_output.put_line(v_in_stock);
end;
```

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



```
--2)
DECLARE
    v_data NUMBER(7);
BEGIN
    DBMS_OUTPUT.PUT_LINE(v_data);
END;
```

```
PL/SQL procedure successfully completed.
```

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';

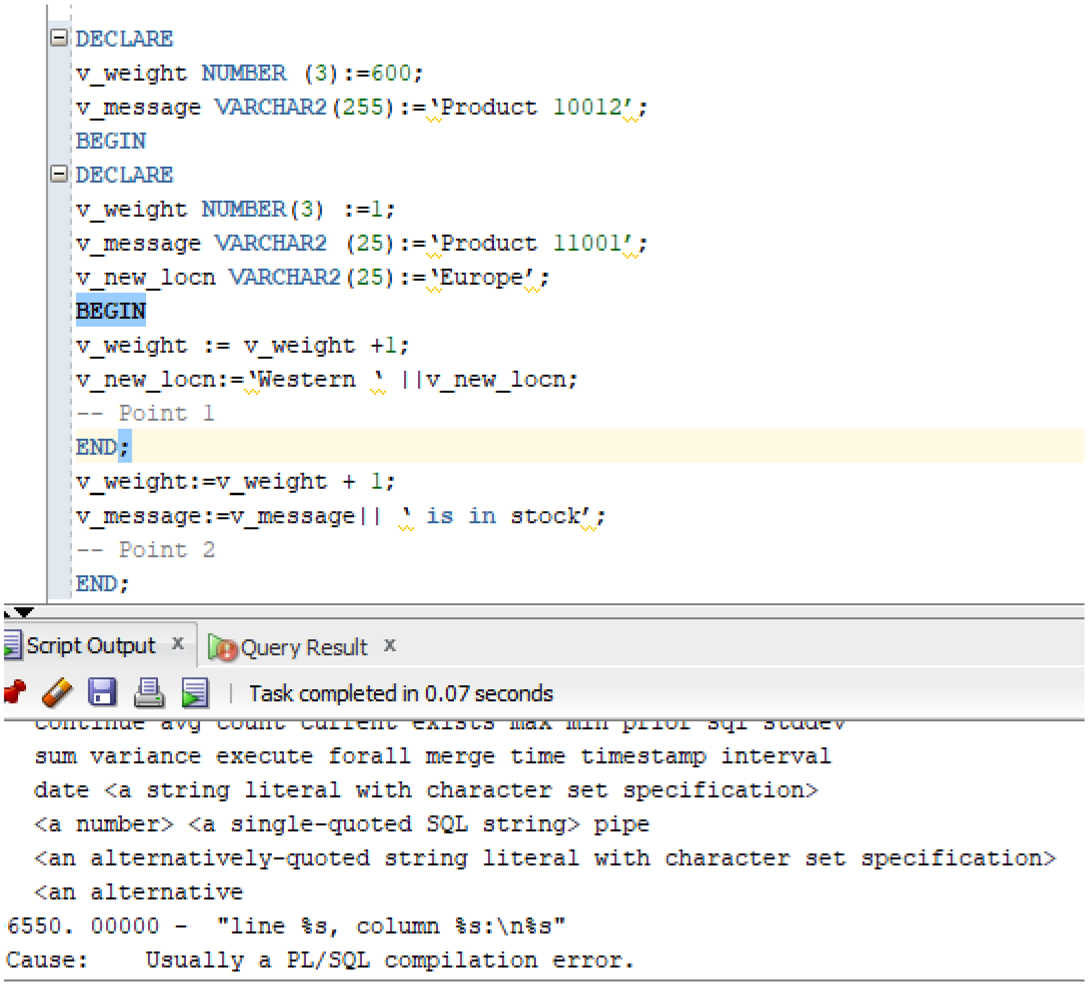
BEGIN

```
 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.



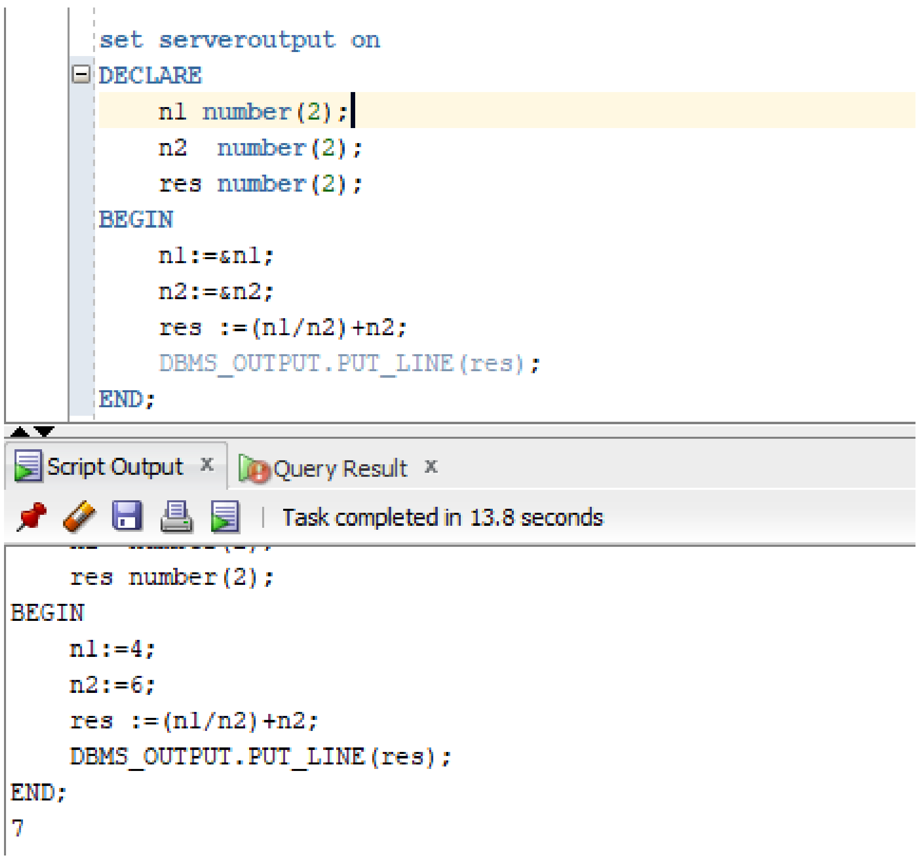
```
DECLARE
v_weight NUMBER (3):=600;
v_message VARCHAR2(255):='Product 10012';
BEGIN
DECLARE
v_weight NUMBER(3) :=1;
v_message VARCHAR2 (25):='Product 11001';
v_new_locn VARCHAR2(25):='Europe';
BEGIN
v_weight := v_weight +1;
v_new_locn:='Western ' ||v_new_locn;
-- Point 1
END;
v_weight:=v_weight + 1;
v_message:=v_message|| ' is in stock';
-- Point 2
END;
```

```
   continue avg count current exists max min prior sql stddev
   sum variance execute forall merge time timestamp interval
   date <a string literal with character set specification>
   <a number> <a single-quoted SQL string> pipe
   <an alternatively-quoted string literal with character set specification>
   <an alternative
6550. 00000 -  "line %s, column %s:\n%s"
Cause:    Usually a PL/SQL compilation error.
```

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;
```


```
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;
```

Task completed in 13.8 seconds

```
    res number(2);
BEGIN
    n1:=4;
    n2:=6;
    res :=(n1/n2)+n2;
    DBMS_OUTPUT.PUT_LINE(res);
END;
7
```

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;
```

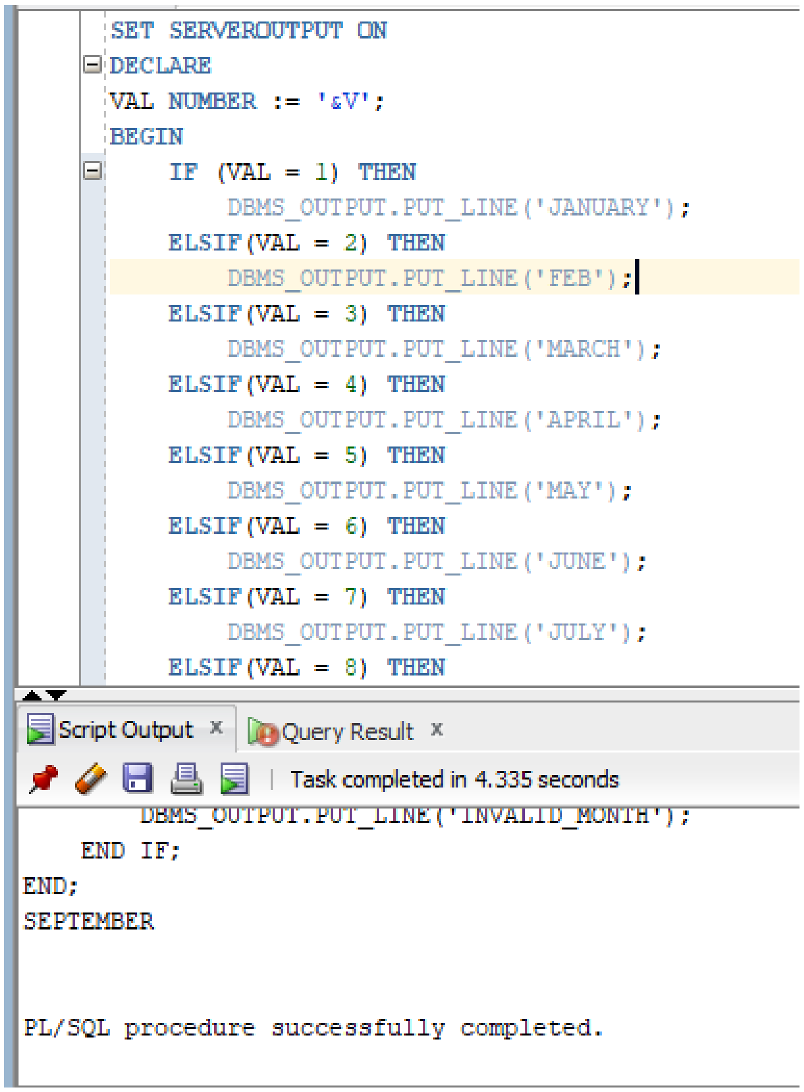
Practice 7:

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;
/
```


```
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
```

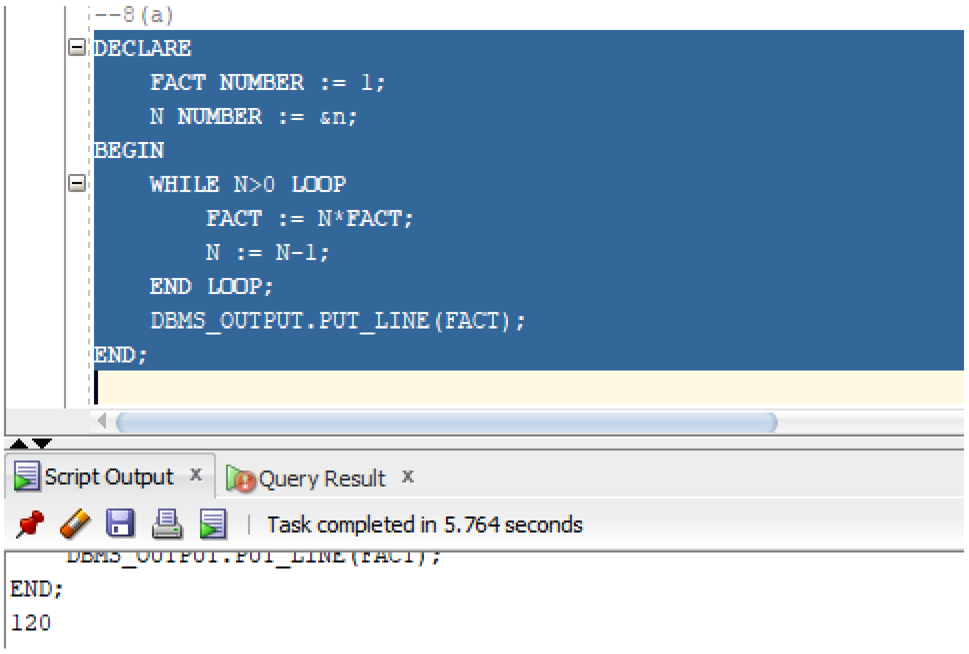
Task completed in 4.335 seconds

```
        DBMS_OUTPUT.PUT_LINE('INVALID_MONTH');
    END IF;
END;
SEPTEMBER


PL/SQL procedure successfully completed.
```

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(a)
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;
```

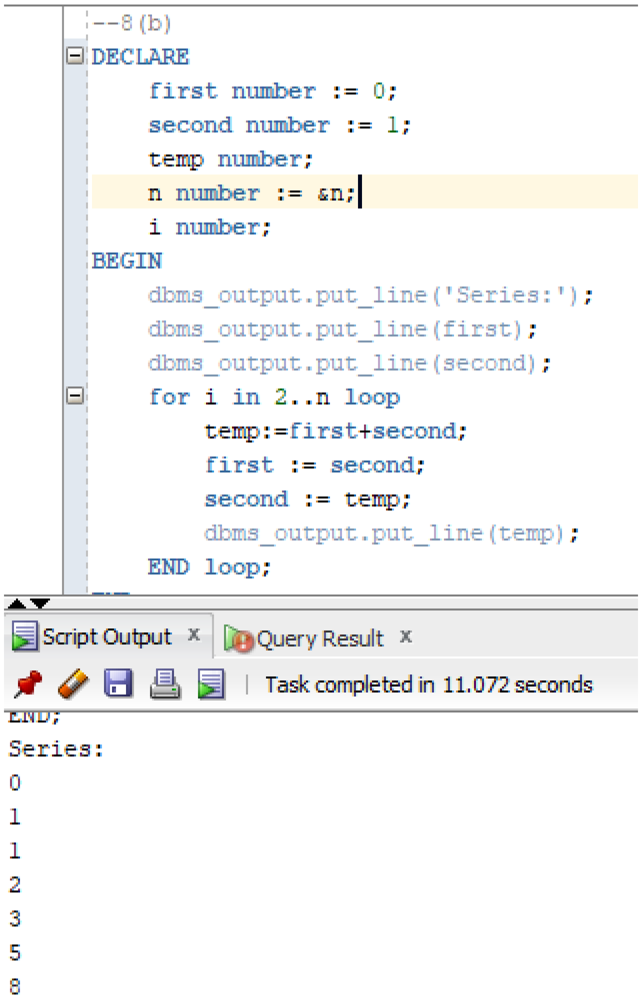
Task completed in 5.764 seconds

```
    DBMS_OUTPUT.PUT_LINE(FACT);
END;
120
```

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;
```



```
--8(b)
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;
```

Script Output × Query Result ×

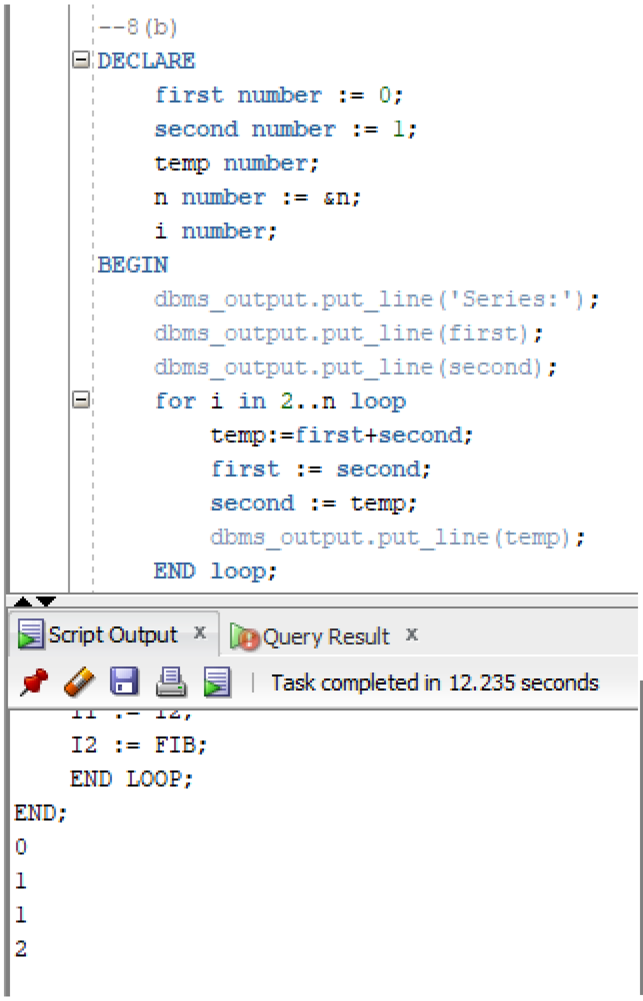
Task completed in 11.072 seconds

```
END;
Series:
0
1
1
2
3
5
8
```

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;
/
```

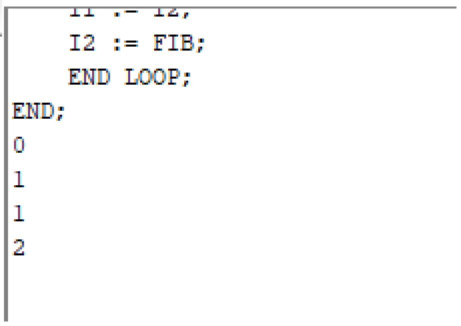


```
--8 (b)
DECLARE
    first number := 0;
    second number := 1;
    temp number;
    n number := &n;
    i number;
BEGIN
    dbms_output.put_line('Series:');
    dbms_output.put_line(first);
    dbms_output.put_line(second);
    for i in 2..n loop
        temp:=first+second;
        first := second;
        second := temp;
        dbms_output.put_line(temp);
    END loop;
```

Script Output × Query Result ×

Task completed in 12.235 seconds

```
    I2 := FIB;
    END LOOP;
END;
0
1
1
2
```

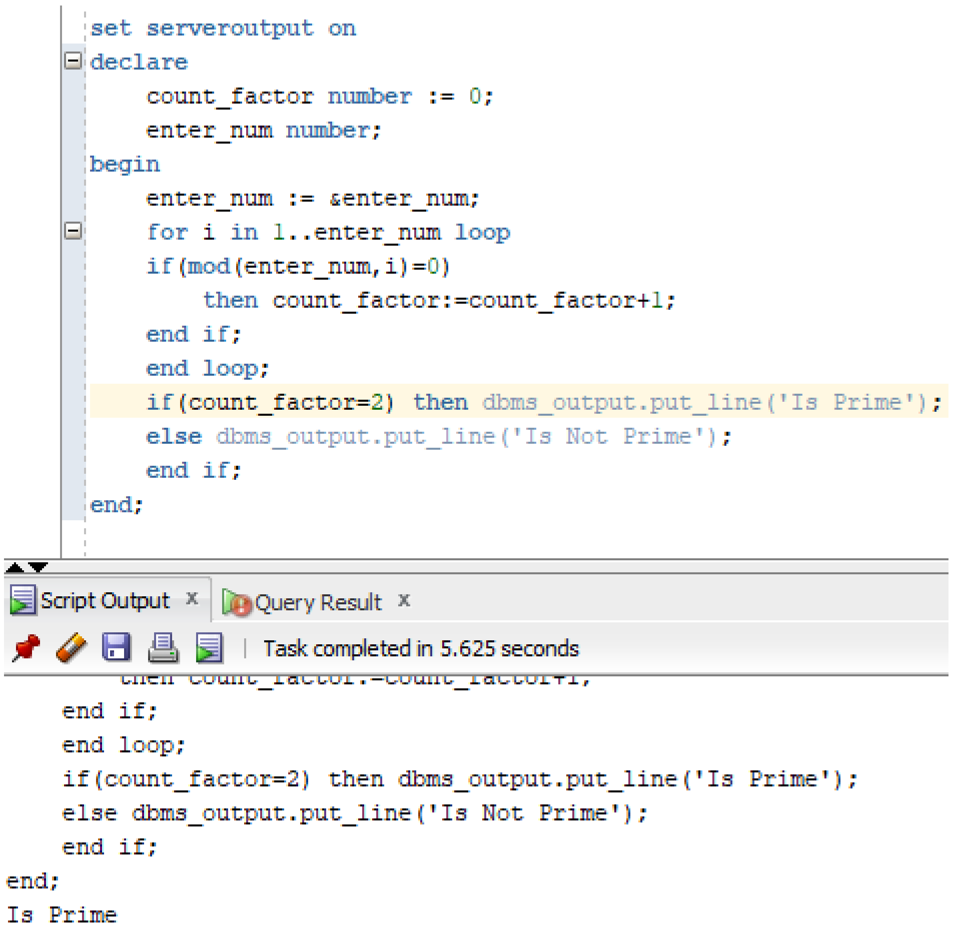


```
    I2 := FIB;
    END LOOP;
END;
0
1
1
2
```

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;
```



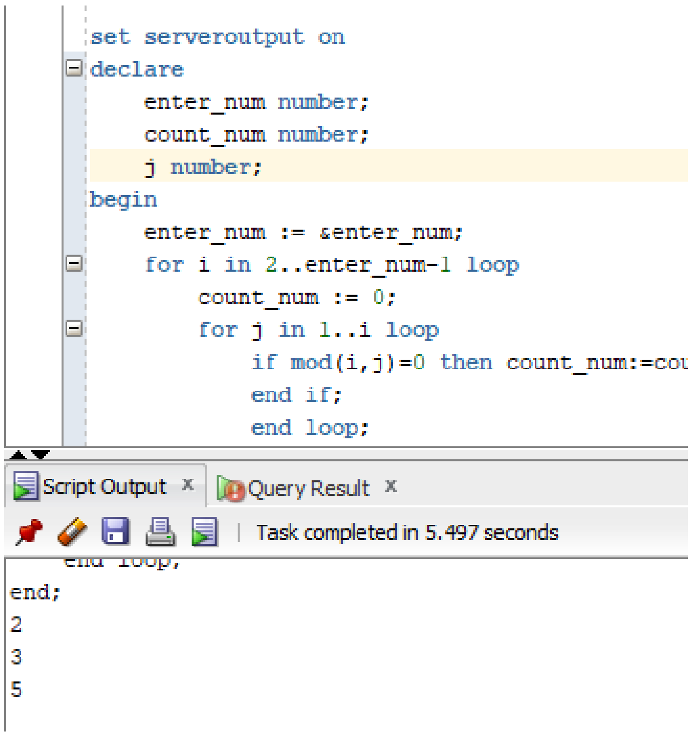
```
set serveroutput on
declare
    count_factor number := 0;
    enter_num number;
begin
    enter_num := &enter_num;
    for i in 1..enter_num loop
    if(mod(enter_num,i)=0)
        then count_factor:=count_factor+1;
    end if;
    end loop;
    if(count_factor=2) then dbms_output.put_line('Is Prime');
    else dbms_output.put_line('Is Not Prime');
    end if;
end;
```

Task completed in 5.625 seconds

```
        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;
Is Prime
```

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;
```



```
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:=cou
            end if;
            end loop;
```

```
Task completed in 5.497 seconds
    end loop;
end;
2
3
5
```

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