Laborator 2: VARIABILE

2.1. Declarare şi iniţializare

- declararea variabilelor se realizează în zona declarativă (delimitată prin DECLARE) a blocului (sau sub-blocului);
- iniţializarea se poate face la declarare sau în zona de execuţie (între BEGIN şi END);
- variabilele vor fi vizibile în restul blocului, respectiv şi în blocurile incluse în el, mai puţin în sub-blocurile în care numele lor este redefinit (ca în majoritatea limbajelor de programare structurate, semnificaţia unui nume definit de utilizator într-un bloc/sub-bloc este dată de cea mai apropiată declaraţie anterioară locului folosirii);
- toate variabilele PL/SQL au un tip de dată, restricţii şi un şir valid de valori;
- declararea şi iniţializarea se realizează astfel:

nume_variabila [CONSTANT] TIP_DATA [NOT NULL] [:= | DEFAULT expresie]

- constantele trebuie obligatoriu iniţializate, iar ulterior nu îşi vor putea schimba valoarea;
- variabilele NOT NULL trebuie obligatoriu iniţializate, iar ulterior nu vor putea primi valoarea NULL;
- se foloseşte următoarea **convenţie de notare**:

c_nume Constanta
v_nume Variabila
g_nume VarGlobala (variabilă globală definită în zona de specificaţii a pachetului de programe şi valabilă pentru toate subprogramele pachetului).

2.2. Tipuri de variabile

Variabile PL/SQL

- **Scalare**
- **Compozite**
- **Referinţă**
- **LOB (Large Objects): NCLOB, CLOB, BLOB, BFILE**
- **Obiect**

Variabile non-PL/SQL: variabile de mediu (BIND VARIABLES)

Variabile Scalare:
Tipurile scalare conţin valori simple (o variabila scalară poate conţine la un moment dat o singură valoare simplă) şi corespund în principal tipurilor pe care le pot avea coloanele tabelelor.

- ✓ **char (lung_max) - lungime fixă de max 32.767 bytes**
- ✓ **varchar2 (lung_max) – lungime variabilă de max 32.767 bytes**
- ✓ **long [şir de caractere de lungime variabilă 2GB]**
- ✓ **number (precizie,scală)**
- ✓ **boolean (true, false, null)**

- ✓ **date**
- ✓ **binary_integer şi pls_integer (numere întregi între -2147483647 şi 2147483647)**
- ✓ **binary_float şi binary_double (pentru numere reale în varianta Oracle 10g)**
- ✓ **timestamp (pentru fracţiuni de secundă)**

Exemple:

```
v_functie varchar2(9);
v_numar binary_integer:=0;
v_totalsal number(9,2):=0;
v_datainceput date:=sysdate+7;
c_taxa constant number(3,2):=8.25;
```

Afişarea variabilelor PL/SQL se realizează prin intermediul funcţiei PUT_LINE din pachetului DBMS_OUTPUT. Se poate utiliza operatorul de concatenare (||) pentru a afişa mai multe mesaje sau variabile pe aceeaşi linie.

```
DBMS_OUTPUT.PUT_LINE ('VALOAREA VARIABILEI ESTE:' ||variabila);
```

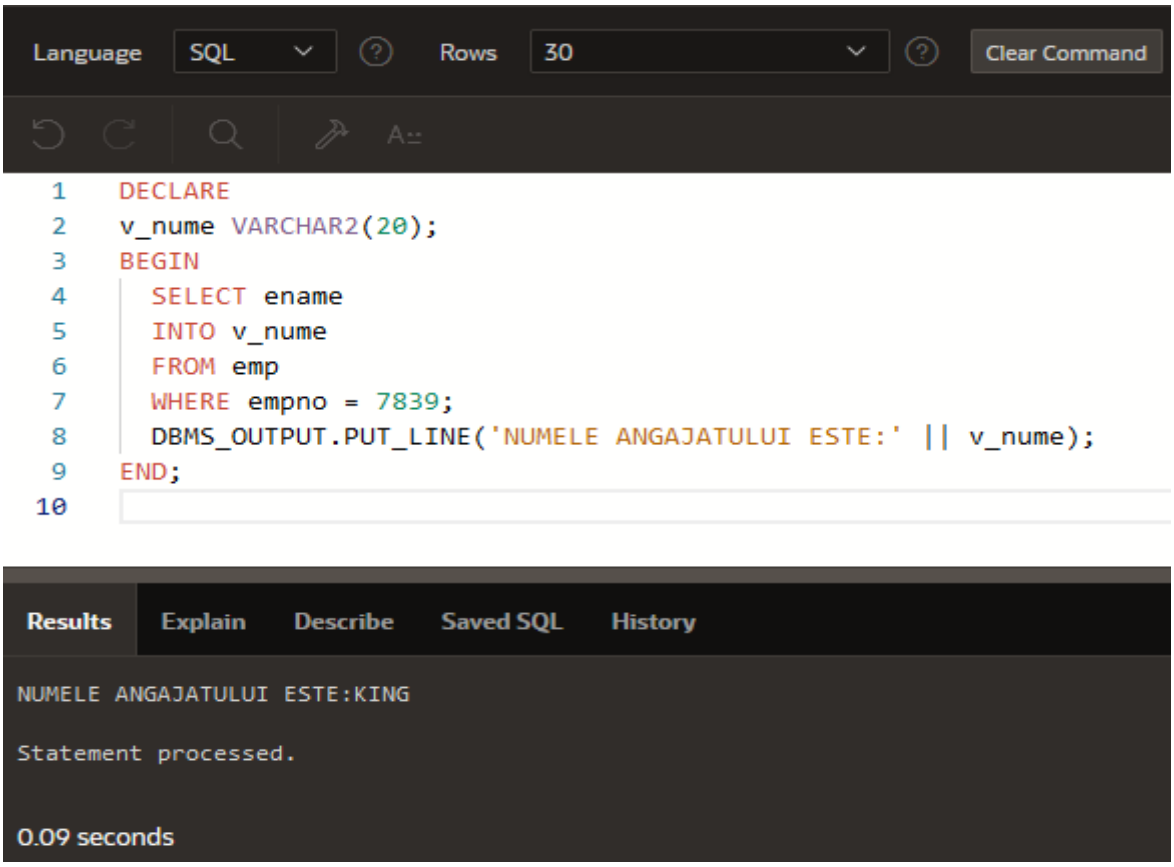
Initialiarea variabilelor cu valori din tabelele bazei de date

- ➢ Se utilizează comanda SELECT cu clauza INTO pentru popularea variabilelor PL/SQL cu valori ale atributelor din tabele;
- ➢ cererile SELECT din cadrul blocurilor PL/SQL trebuie să furnizeze o singură linie rezultat (în caz contrar se semnalează eroare).

Exemplu:

```
-- se afiseaza numele angajatului cu codul 7839

DECLARE
        v_nume VARCHAR2(20);
BEGIN
  SELECT ename
  INTO v_nume
  FROM emp
  WHERE empno = 7839;
  DBMS_OUTPUT.PUT_LINE('NUMELE ANGAJATULUI ESTE:' || v_nume);
END;
/
```



2.3. Atributul %TYPE

Atribuie unei variabile tipul altei variabile sau tipul de date specific unei coloane din tabelă.

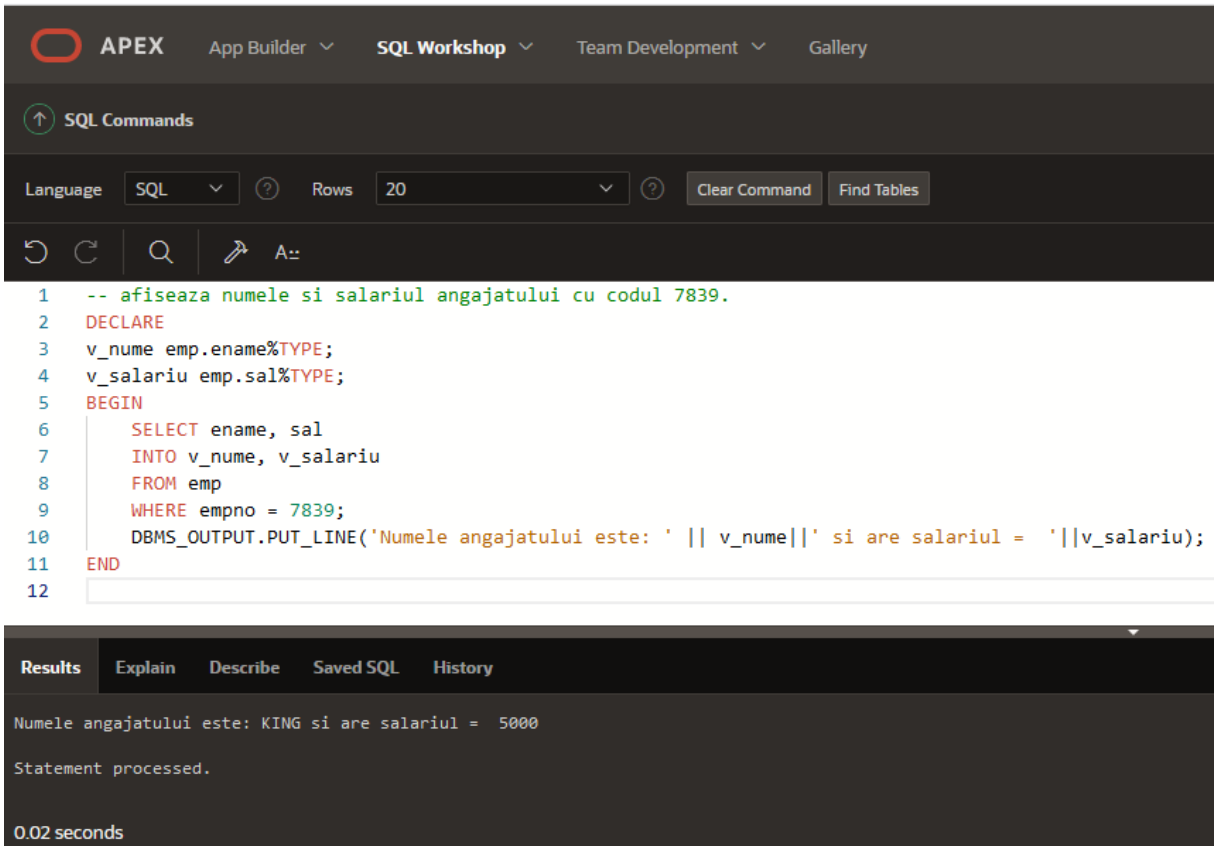
Declararea unei variabile cu %TYPE:

variabila tabelă.nume_coloană%TYPE;
sau
variabila1 tip_dată;
variabila2 variabila1%TYPE;

Exemplu:

```
-- afiseaza numele si salariul angajatului cu codul 7839.
DECLARE
        v_nume emp.ename%TYPE;
        v_salariu emp.sal%TYPE;
BEGIN
        SELECT ename, sal
        INTO v_nume, v_salariu
        FROM emp
        WHERE empno = 7839;
        DBMS_OUTPUT.PUT_LINE('Numele angajatului este: ' || v_nume||'
si are salariul =  '||v_salariu);
END;
/
```

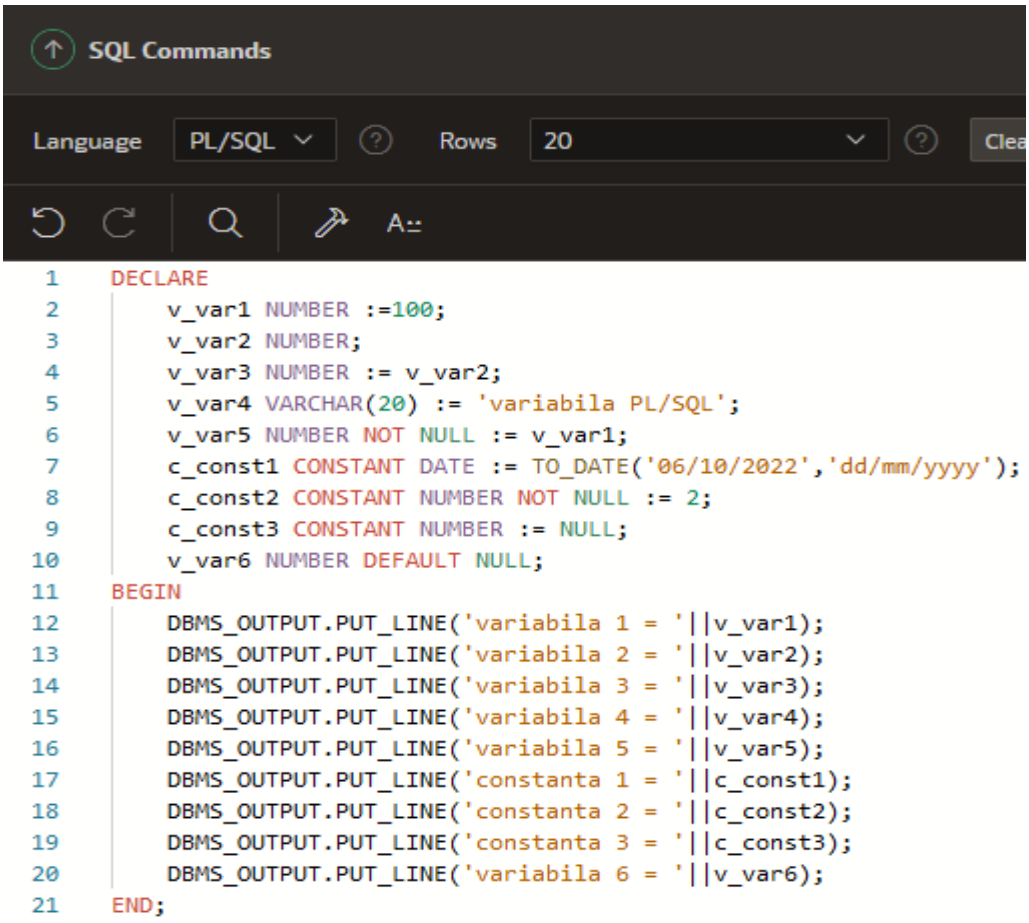
Observaţie: Restricţia NOT NULL a unei coloane nu se aplică şi variabilei declarate prin folosirea atributului %TYPE.



Probleme propuse spre rezolvare

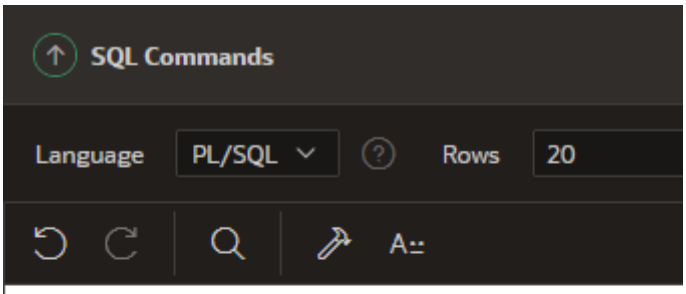
1. Specificaţi ce se va afişa la rularea următorului bloc PL/SQL:

```
DECLARE
   v_var1 NUMBER :=100;
   v_var2 NUMBER;
   v_var3 NUMBER := v_var2;
   v_var4 VARCHAR(20) := 'variabila PL/SQL';
   v_var5 NUMBER NOT NULL := v_var1;
   c_const1 CONSTANT DATE := TO_DATE('06/10/2022','dd/mm/yyyy');
   c_const2 CONSTANT NUMBER NOT NULL := 2;
   c_const3 CONSTANT NUMBER := NULL;
   v_var6 NUMBER DEFAULT NULL;
BEGIN
   DBMS_OUTPUT.PUT_LINE('variabila 1 = '||v_var1);
   DBMS_OUTPUT.PUT_LINE('variabila 2 = '||v_var2);
   DBMS_OUTPUT.PUT_LINE('variabila 3 = '||v_var3);
   DBMS_OUTPUT.PUT_LINE('variabila 4 = '||v_var4);
   DBMS_OUTPUT.PUT_LINE('variabila 5 = '||v_var5);
   DBMS_OUTPUT.PUT_LINE('constanta 1 = '||c_const1);
   DBMS_OUTPUT.PUT_LINE('constanta 2 = '||c_const2);
   DBMS_OUTPUT.PUT_LINE('constanta 3 = '||c_const3);
   DBMS_OUTPUT.PUT_LINE('variabila 6 = '||v_var6);
END;
```



2. Specificaţi ce se va afişa la rularea următorului bloc PL/SQL (care conţine blocuri imbricate, ilustrând domeniul de vizibilitate al unor variabile care au acelaşi nume):

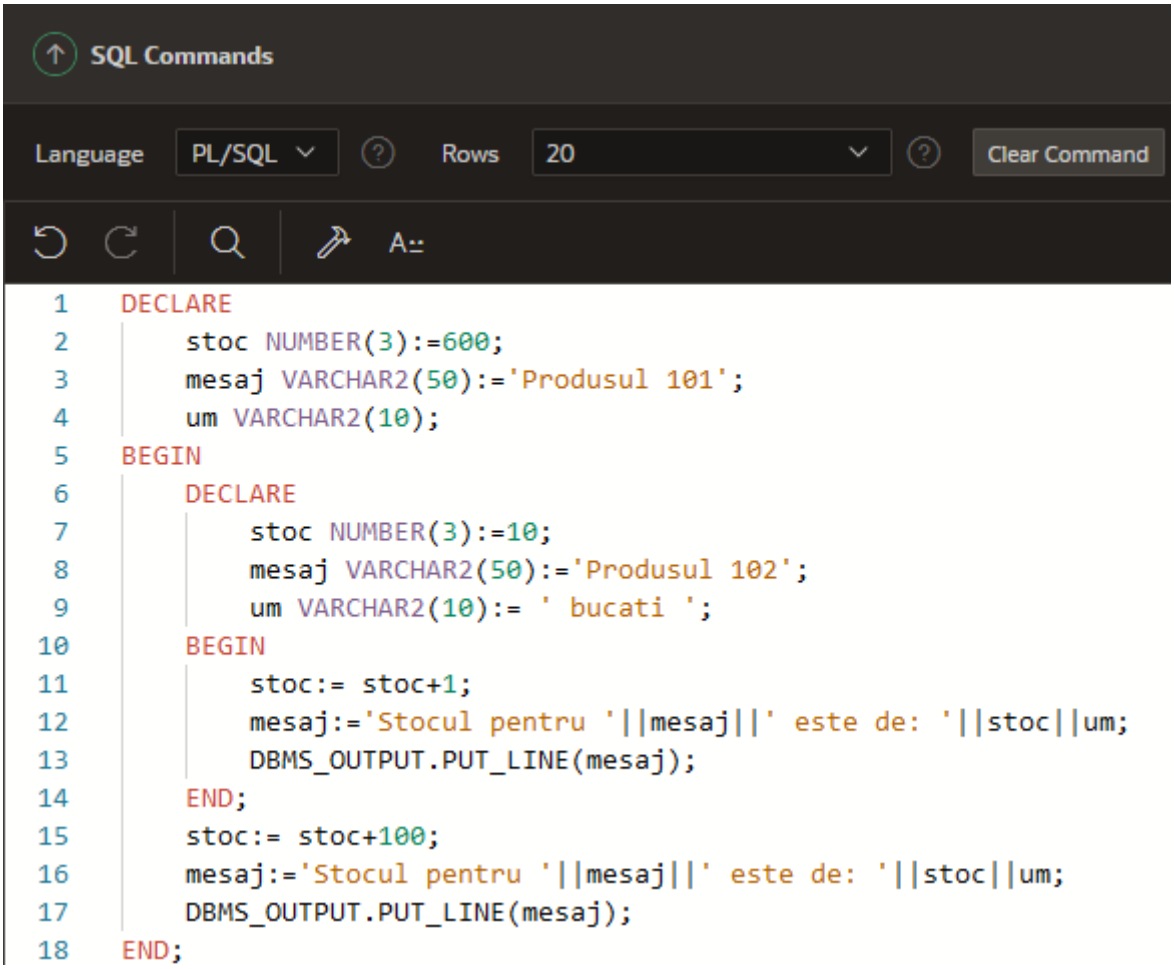
```
DECLARE
        var NUMBER;
BEGIN
        var := 1;
        DBMS_OUTPUT.PUT_LINE(var);
        <<bloc1>>
        DECLARE
                var NUMBER;
        BEGIN
                var :=2;
                DBMS_OUTPUT.PUT_LINE(var);
        END bloc1;
        DBMS_OUTPUT.PUT_LINE(var);
        <<bloc2>>
        DECLARE
                var NUMBER;
        BEGIN
                var :=3;
                DBMS_OUTPUT.PUT_LINE(var);
        <<bloc3>>
        DECLARE
                var NUMBER;
                BEGIN
                        var :=4;
                        DBMS_OUTPUT.PUT_LINE(var);
                        DBMS_OUTPUT.PUT_LINE(bloc2.var);
        END bloc3;
        DBMS_OUTPUT.PUT_LINE(var);
        END bloc2;
        DBMS_OUTPUT.PUT_LINE(var);
END;
```




```
DECLARE
    var NUMBER;
BEGIN
    var := 1;
    DBMS_OUTPUT.PUT_LINE(var);
    <<bloc1>>
    DECLARE
        var NUMBER;
    BEGIN
        var :=2;
        DBMS_OUTPUT.PUT_LINE(var);
    END bloc1;
    DBMS_OUTPUT.PUT_LINE(var);
    <<bloc2>>
    DECLARE
        var NUMBER;
    BEGIN
        var :=3;
        DBMS_OUTPUT.PUT_LINE(var);
        <<bloc3>>
        DECLARE
            var NUMBER;
        BEGIN
            var :=4;
            DBMS_OUTPUT.PUT_LINE(var);
            DBMS_OUTPUT.PUT_LINE(bloc2.var);
        END bloc3;
        DBMS_OUTPUT.PUT_LINE(var);
    END bloc2;
    DBMS_OUTPUT.PUT_LINE(var);
END;
```

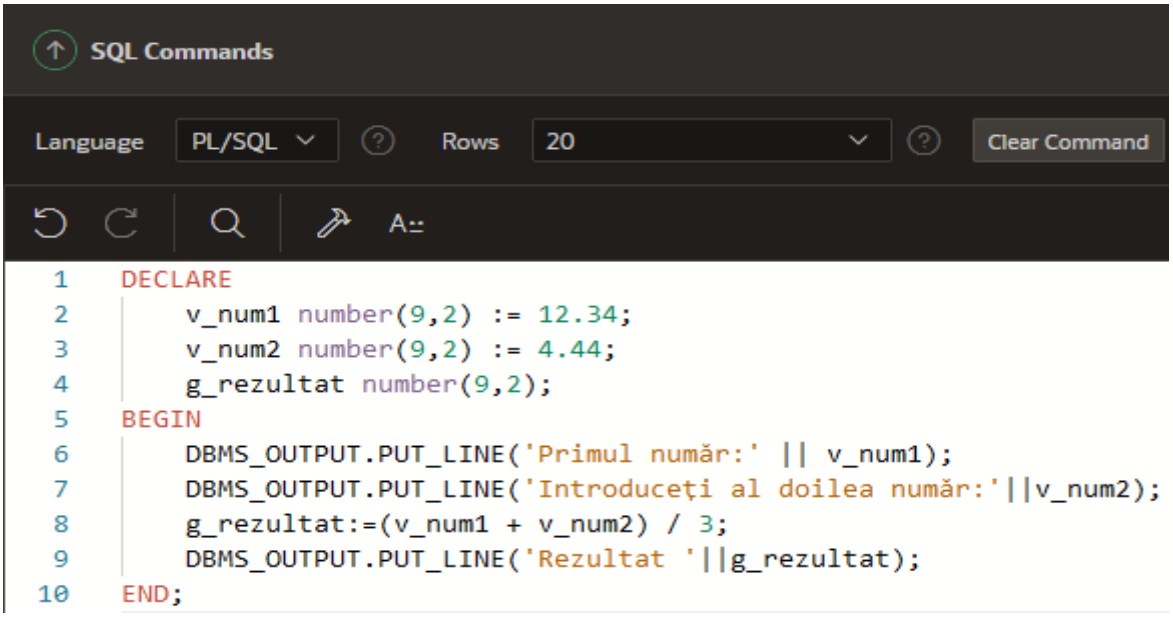
3. Specificați ce se va afișa la rularea următorului bloc PL/SQL:

```
DECLARE
        stoc NUMBER(3):=600;
        mesaj VARCHAR2(50):='Produsul 101';
        um VARCHAR2(10);
BEGIN
        DECLARE
                stoc NUMBER(3):=10;
                mesaj VARCHAR2(50):='Produsul 102';
                um VARCHAR2(10):= ' bucati ';
        BEGIN
                stoc:= stoc+1;
                mesaj:='Stocul pentru '||mesaj||' este de: '||stoc||um;
                DBMS_OUTPUT.PUT_LINE(mesaj);
        END;
        stoc:= stoc+100;
        mesaj:='Stocul pentru '||mesaj||' este de: '||stoc||um;
        DBMS_OUTPUT.PUT_LINE(mesaj);
END;
```



4. Să se calculeze suma a două numere, iar rezultatul să se dividă cu 3. Numerele se initializează prin codul PL/SQL.

```
DECLARE
   v_num1 number(9,2) := 12.34;
   v_num2 number(9,2) := 4.44;
   g_rezultat number(9,2);
BEGIN
   DBMS_OUTPUT.PUT_LINE('Primul număr:' || v_num1);
   DBMS_OUTPUT.PUT_LINE('Introduceţi al doilea număr:'||v_num2);
   g_rezultat:=(v_num1 + v_num2) / 3;
   DBMS_OUTPUT.PUT_LINE('Rezultat '||g_rezultat);
END;
```



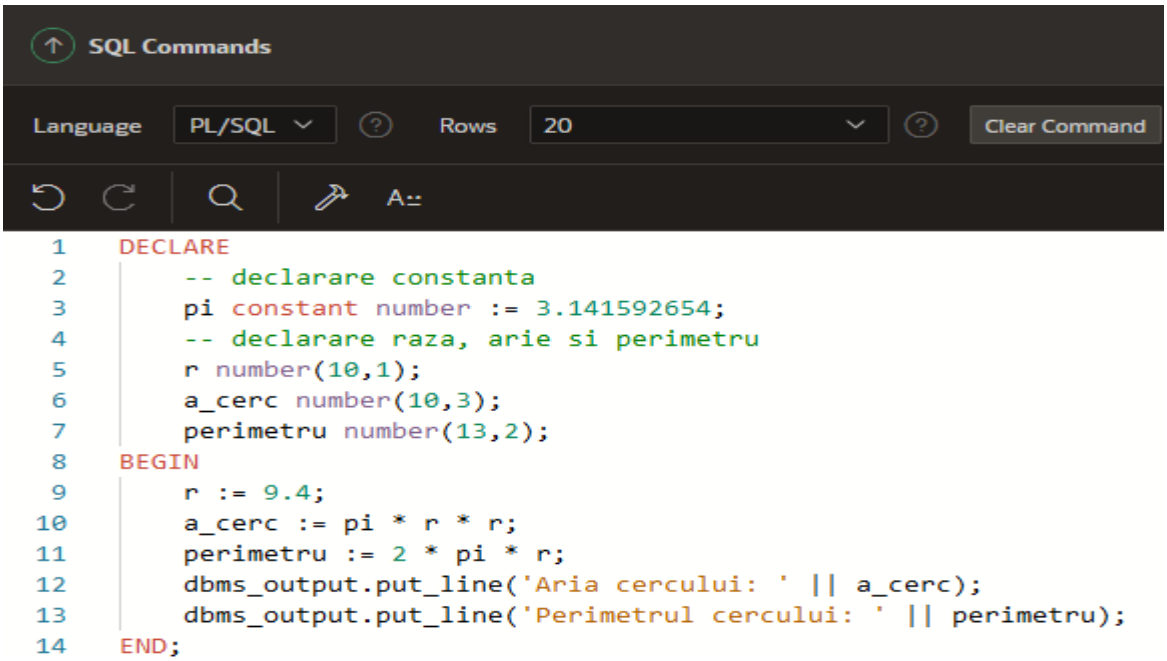
5. Să se afişeze salariul mărit cu un procent. Salariul şi procentul se precizează in codul PL/SQL.

```
DECLARE
   v_sal NUMBER(9,4);
   v_procent NUMBER(9,4);
BEGIN
   v_sal := 10000.00;
   v_procent := 12.50;
   dbms_output.put_line('Salariul este = '||v_sal);
   dbms_output.put_line('Procentul de marire = '||v_procent);
   dbms_output.put_line(to_char(nvl(v_sal, 0) * (1 + nvl(v_procent, 0) / 100)));
END;
```


```
1   DECLARE
2       v_sal NUMBER(9,4);
3       v_procent NUMBER(9,4);
4   BEGIN
5       v_sal := 10000.00;
6       v_procent := 12.50;
7       dbms_output.put_line('Salariul este = '||v_sal);
8       dbms_output.put_line('Procentul de marire = '||v_procent);
9       dbms_output.put_line(to_char(nvl(v_sal,0)*(1+nvl(v_procent,0)/100)));
10  END;
```

6. Să se afişeze aria unui cerc, stiind raza cercului.

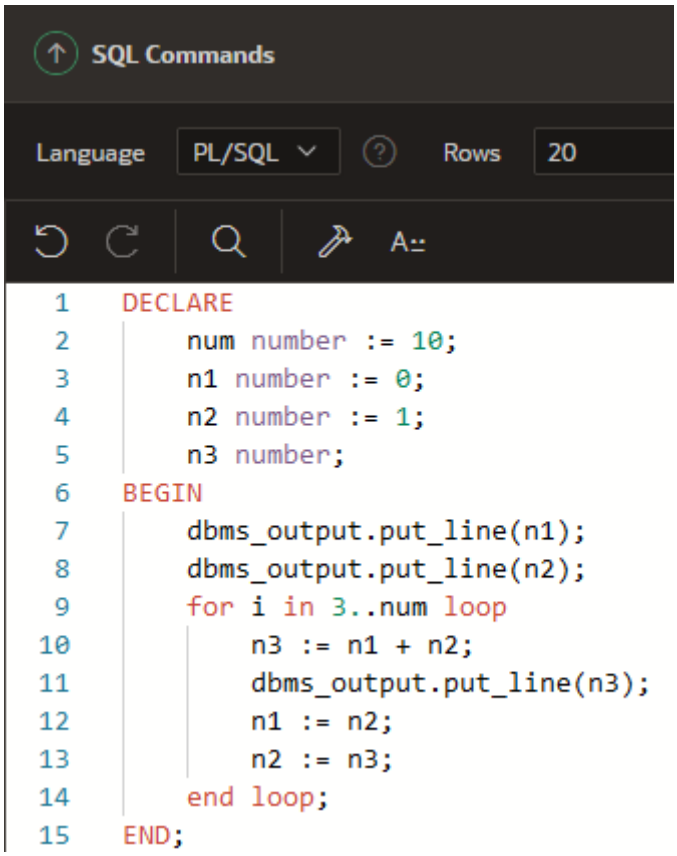
```
DECLARE
   -- declarare constanta
   pi constant number := 3.141592654;
   -- declarare raza, arie si perimetru
   r number(10,1);
   a_cerc number(10,3);
        perimetru number(13,2);
BEGIN
        r := 9.4;
        a_cerc := pi * r * r;
        perimetru := 2 * pi * r;
        dbms_output.put_line('Aria cercului: ' || a_cerc);
        dbms_output.put_line('Perimetrul cercului: ' || perimetru);
END;
```




```
1   DECLARE
2       -- declarare constanta
3       pi constant number := 3.141592654;
4       -- declarare raza, arie si perimetru
5       r number(10,1);
6       a_cerc number(10,3);
7       perimetru number(13,2);
8   BEGIN
9       r := 9.4;
10      a_cerc := pi * r * r;
11      perimetru := 2 * pi * r;
12      dbms_output.put_line('Aria cercului: ' || a_cerc);
13      dbms_output.put_line('Perimetrul cercului: ' || perimetru);
14  END;
```

7. Să se afişeze primii 10 termeni din sirul lui Fibonacci.

```
DECLARE
        num number := 10;
        n1 number := 0;
        n2 number := 1;
        n3 number;
BEGIN
        dbms_output.put_line(n1);
        dbms_output.put_line(n2);
        for i in 3..num loop
                n3 := n1 + n2;
                dbms_output.put_line(n3);
                n1 := n2;
                n2 := n3;
        end loop;
END;
```



Bibliografie web:
https://www.softwaretestinghelp.com/pl-sql-data-types-variables-constants-literals/
https://www.bullraider.com/database/pl-sql/pl-sql-examples
https://www.oracletutorial.com/plsql-tutorial/