

## Laborator 2: VARIABLE

### 2.1. Declarare și inițializare

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

**nume\_variabila [CONSTANT] TIP\_DATA [NOT NULL] [:= |  
DEFAULT expresie]**

- constantele trebuie obligatoriu inițializate, iar ulterior nu își vor putea schimba valoarea;
- variabilele NOT NULL trebuie obligatoriu inițializate, iar ulterior nu vor putea primi valoarea NULL;
- se folosește următoarea **convenție de notare**:

**c\_nume Constanta**

**v\_nume Variabila**

**g\_nume VarGlobala** (variabilă globală definită în zona de specificații a pachetului de programe și valabilă pentru toate subprogramele pachetului).

### 2.2. Tipuri de variabile

Variabile PL/SQL

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

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

Variabile Scalare:

Tipurile scalare conțin valori simple (o variabila scalară poate conține la un moment dat o singură valoare simplă) și corespund în principal tipurilor pe care le pot avea coloanele tabelor.

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

- ✓ **date**
- ✓ **binary\_integer** și **pls\_integer** (numere întregi între -2147483647 și 2147483647)
- ✓ **binary\_float** și **binary\_double** (pentru numere reale în varianta Oracle 10g)
- ✓ **timestamp** (pentru fracțiuni 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;
```

Afișarea variabilelor PL/SQL se realizează prin intermediul funcției PUT\_LINE din pachetul DBMS\_OUTPUT. Se poate utiliza operatorul de concatenare ( || ) pentru a afișa mai multe mesaje sau variabile pe aceeași linie.

**DBMS\_OUTPUT.PUT\_LINE ('VALOAREA VARIABILEI ESTE:'  
||variabila);**

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

```
Language SQL Rows 30 Clear Command
1 DECLARE
2 v_ume VARCHAR2(20);
3 BEGIN
4 SELECT ename
5 INTO v_ume
6 FROM emp
7 WHERE empno = 7839;
8 DBMS_OUTPUT.PUT_LINE('NUMELE ANGAJATULUI ESTE:' || v_ume);
9 END;
10
```

```
Results Explain Describe Saved SQL History
NUMELE ANGAJATULUI ESTE:KING
Statement processed.
0.09 seconds
```

### 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_ume emp.ename%TYPE;  
    v_salariu emp.sal%TYPE;  
BEGIN  
    SELECT ename, sal  
    INTO v_ume, v_salariu  
    FROM emp  
    WHERE empno = 7839;  
    DBMS_OUTPUT.PUT_LINE('Numele angajatului este: ' || v_ume||'  
si are salariul = '||v_salariu);  
END;  
/
```

*Observație: Restricția NOT NULL a unei coloane nu se aplică și variabilei declarate prin folosirea atributului %TYPE.*

The screenshot displays the Oracle APEX SQL Workshop interface. At the top, the navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing a 'Language' dropdown set to 'SQL' and 'Rows' set to '20'. The main area contains a PL/SQL script:

```
1  -- afiseaza numele si salariul angajatului cu codul 7839.
2  DECLARE
3  v_nume emp.ename%TYPE;
4  v_salariu emp.sal%TYPE;
5  BEGIN
6      SELECT ename, sal
7      INTO v_nume, v_salariu
8      FROM emp
9      WHERE empno = 7839;
10     DBMS_OUTPUT.PUT_LINE('Numele angajatului este: ' || v_nume||' si are salariul = ' ||v_salariu);
11 END
12
```

Below the script, the 'Results' tab is selected, showing the output of the query:

```
Numele angajatului este: KING si are salariul = 5000
Statement processed.
0.02 seconds
```

## Probleme propuse spre rezolvare

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

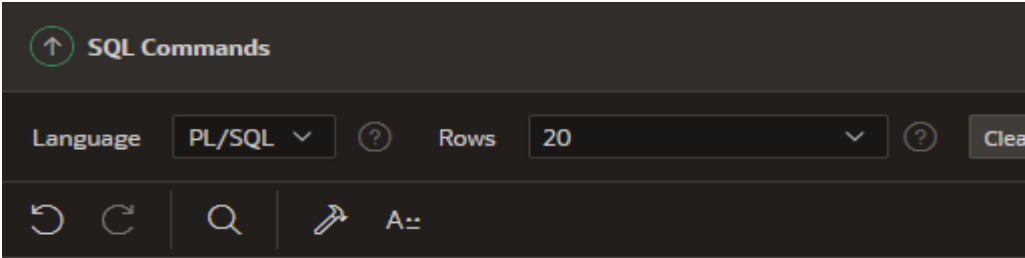
**DECLARE**

```
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;**



```
1 DECLARE
2     v_var1 NUMBER :=100;
3     v_var2 NUMBER;
4     v_var3 NUMBER := v_var2;
5     v_var4 VARCHAR(20) := 'variabila PL/SQL';
6     v_var5 NUMBER NOT NULL := v_var1;
7     c_const1 CONSTANT DATE := TO_DATE('06/10/2022','dd/mm/yyyy');
8     c_const2 CONSTANT NUMBER NOT NULL := 2;
9     c_const3 CONSTANT NUMBER := NULL;
10    v_var6 NUMBER DEFAULT NULL;
11 BEGIN
12     DBMS_OUTPUT.PUT_LINE('variabila 1 = '||v_var1);
13     DBMS_OUTPUT.PUT_LINE('variabila 2 = '||v_var2);
14     DBMS_OUTPUT.PUT_LINE('variabila 3 = '||v_var3);
15     DBMS_OUTPUT.PUT_LINE('variabila 4 = '||v_var4);
16     DBMS_OUTPUT.PUT_LINE('variabila 5 = '||v_var5);
17     DBMS_OUTPUT.PUT_LINE('constanta 1 = '||c_const1);
18     DBMS_OUTPUT.PUT_LINE('constanta 2 = '||c_const2);
19     DBMS_OUTPUT.PUT_LINE('constanta 3 = '||c_const3);
20     DBMS_OUTPUT.PUT_LINE('variabila 6 = '||v_var6);
21 END;
```

2. Specificați ce se va afișa la rularea următorului bloc PL/SQL (care conține blocuri imbricate, ilustrând domeniul de vizibilitate al unor variabile care au același 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;
```

```
SQL Commands

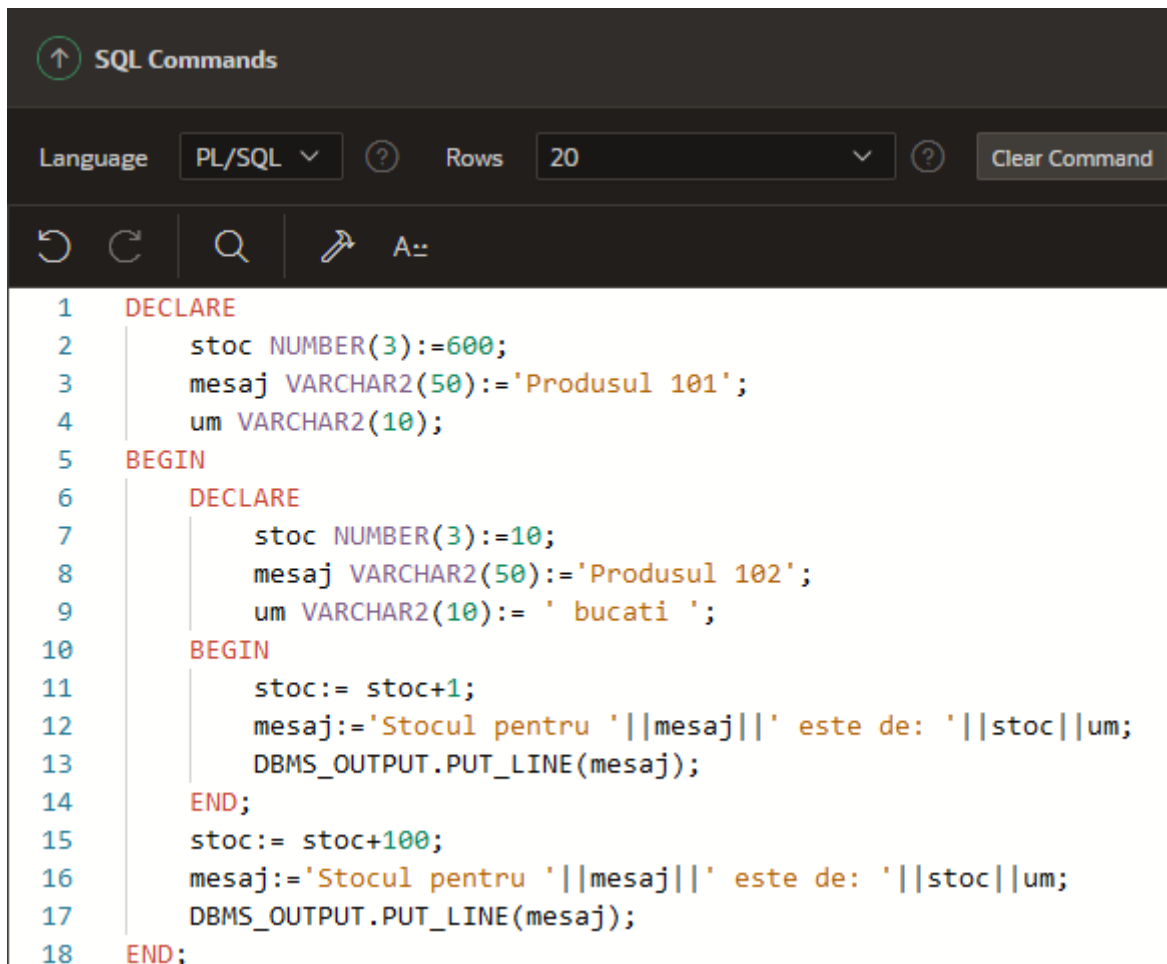
Language PL/SQL ? Rows 20

↶ ↷ 🔍 ↵ A::

1 DECLARE
2   var NUMBER;
3 BEGIN
4   var := 1;
5   DBMS_OUTPUT.PUT_LINE(var);
6   <<bloc1>>
7   DECLARE
8     var NUMBER;
9   BEGIN
10    var :=2;
11    DBMS_OUTPUT.PUT_LINE(var);
12  END bloc1;
13  DBMS_OUTPUT.PUT_LINE(var);
14  <<bloc2>>
15  DECLARE
16    var NUMBER;
17  BEGIN
18    var :=3;
19    DBMS_OUTPUT.PUT_LINE(var);
20    <<bloc3>>
21    DECLARE
22      var NUMBER;
23    BEGIN
24      var :=4;
25      DBMS_OUTPUT.PUT_LINE(var);
26      DBMS_OUTPUT.PUT_LINE(bloc2.var);
27    END bloc3;
28    DBMS_OUTPUT.PUT_LINE(var);
29  END bloc2;
30  DBMS_OUTPUT.PUT_LINE(var);
31 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;
```



The screenshot shows a SQL Commands window with a dark theme. At the top, there is a title bar with an upward arrow and the text "SQL Commands". Below the title bar, there is a toolbar with a "Language" dropdown set to "PL/SQL", a "Rows" dropdown set to "20", and a "Clear Command" button. Below the toolbar, there are icons for undo, redo, search, and a cursor. The main area of the window displays the PL/SQL code from the previous block, with line numbers 1 through 18 on the left side. The code is color-coded: keywords are in red, identifiers and literals are in green, and operators and punctuation are in blue.

```
1 DECLARE
2     stoc NUMBER(3):=600;
3     mesaj VARCHAR2(50):='Produsul 101';
4     um VARCHAR2(10);
5 BEGIN
6     DECLARE
7         stoc NUMBER(3):=10;
8         mesaj VARCHAR2(50):='Produsul 102';
9         um VARCHAR2(10):= ' bucati ';
10    BEGIN
11        stoc:= stoc+1;
12        mesaj:='Stocul pentru '||mesaj||' este de: '||stoc||um;
13        DBMS_OUTPUT.PUT_LINE(mesaj);
14    END;
15    stoc:= stoc+100;
16    mesaj:='Stocul pentru '||mesaj||' este de: '||stoc||um;
17    DBMS_OUTPUT.PUT_LINE(mesaj);
18 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**

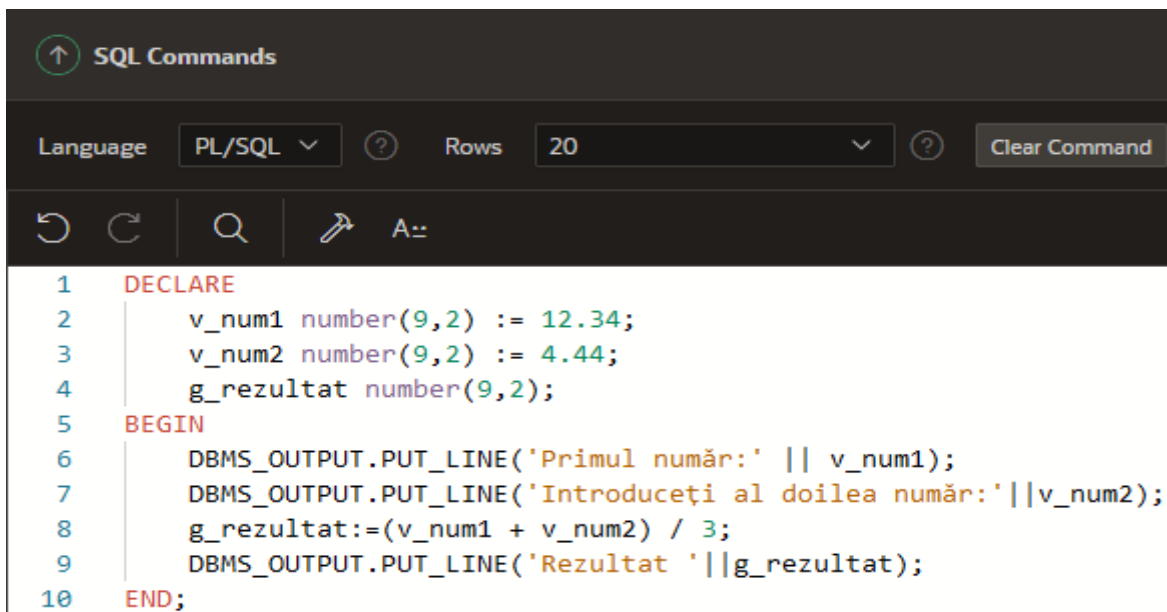
**DBMS\_OUTPUT.PUT\_LINE('Primul număr:' || v\_num1);**

**DBMS\_OUTPUT.PUT\_LINE('Introduceți al doilea număr:' || v\_num2);**

**g\_rezultat:=(v\_num1 + v\_num2) / 3;**

**DBMS\_OUTPUT.PUT\_LINE('Rezultat ' || g\_rezultat);**

**END;**



The screenshot shows the SQL Developer interface with the following code in the editor:

```
1 DECLARE
2     v_num1 number(9,2) := 12.34;
3     v_num2 number(9,2) := 4.44;
4     g_rezultat number(9,2);
5 BEGIN
6     DBMS_OUTPUT.PUT_LINE('Primul număr:' || v_num1);
7     DBMS_OUTPUT.PUT_LINE('Introduceți al doilea număr:' || v_num2);
8     g_rezultat:=(v_num1 + v_num2) / 3;
9     DBMS_OUTPUT.PUT_LINE('Rezultat ' || g_rezultat);
10 END;
```

5. Să se afișeze salariul mărit cu un procent. Salariul și procentul se precizează în 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;**

```

SQL Commands
Language PL/SQL Rows 20 Clear Command Find Tables
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 afișeze aria unui cerc, știind 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;**

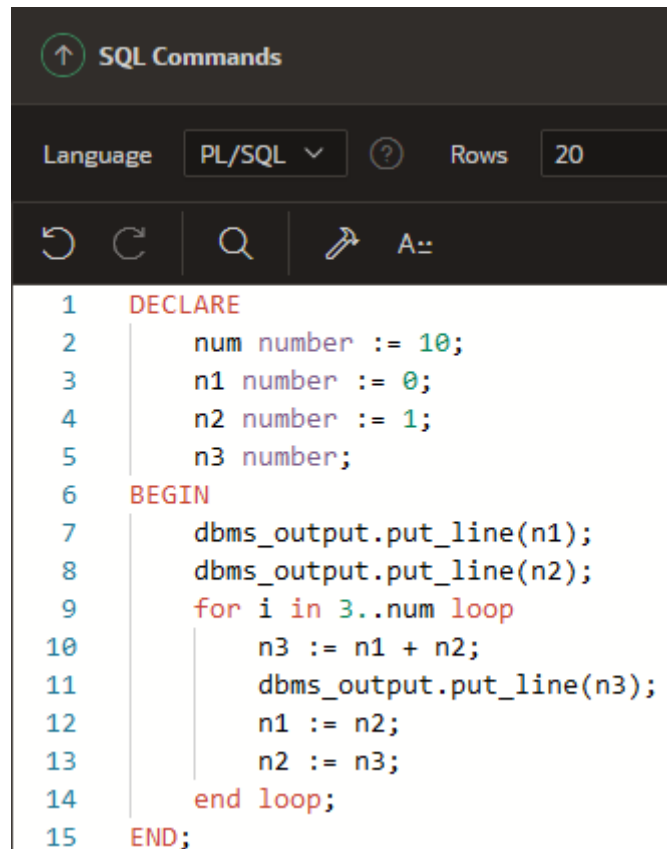
```

SQL Commands
Language PL/SQL Rows 20 Clear Command
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 afișeze 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;
```



The screenshot shows a dark-themed SQL Command window. At the top, it says "SQL Commands" with an upward arrow icon. Below that, there are controls for "Language" (set to "PL/SQL"), a help icon, and "Rows" (set to "20"). A toolbar contains icons for refresh, redo, search, and a keyboard shortcut "A::". The main area displays the following PL/SQL code:

```
1 DECLARE
2     num number := 10;
3     n1 number := 0;
4     n2 number := 1;
5     n3 number;
6 BEGIN
7     dbms_output.put_line(n1);
8     dbms_output.put_line(n2);
9     for i in 3..num loop
10        n3 := n1 + n2;
11        dbms_output.put_line(n3);
12        n1 := n2;
13        n2 := n3;
14    end loop;
15 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/>