

Laborator 1: Introducere în PL/SQL

1.1. Caracteristici generale:

- Construcțiile PL/SQL conțin structuri de control procedurale și comenzi descriptive SQL
- PL/SQL este un limbaj procedural structurat pe bloc, programele putând fi împărțite în blocuri logice
- Blocurile PL/SQL sunt procesate de motorul PL/SQL care poate fi rezident pe ORACLE SERVER sau pe un instrument de dezvoltare (ex.: Oracle Forms, Reports, JDeveloper etc.)
- Multe instrumente ORACLE au propriul motor PL/SQL (ex.: Oracle Forms, Reports, JDeveloper etc.)
- Tipurile de date din SQL pot fi folosite în PL/SQL
- Programarea în PL/SQL este modularizată – se utilizează blocurile care grupează instrucțiunile

1.2. Blocuri PL/SQL:

Orice unitate PL/SQL conține unul sau mai multe blocuri, complet separate sau imbricate.

Componentele unui bloc PL/SQL:

Un bloc PL/SQL este compus din până la 3 secțiuni: declarativă (opțională), executabilă (obligatorie) și de tratare a excepțiilor (opțională).

DECLARE (*Opțional*)
variabile, cursori, excepții
BEGIN (*Obligatoriu*)
comenzi SQL (asigură accesul la baza de date)
structuri de programare procedurală PL/SQL
EXCEPTION (*Opțional*)
acțiuni ce se execută când apare o eroare
END; (*Obligatoriu*)

Observații:

- comenzile SQL asigură accesul la baza de date
- operațiile efectuate cu variabilele PL/SQL în cadrul instrucțiunilor procedurale nu presupun accesarea bazei de date
- se folosește (;) după fiecare instrucțiune SQL sau instrucțiune de control PL/SQL
- blocul PL/SQL se termină cu (;)
- se folosește (/) pentru a lansa un bloc anonim în bufferul SQL
- o eroare în PL/SQL este tratată ca o excepție

Tipuri de blocuri PL/SQL:

Blocuri anonime:

1. Funcții stocate și funcții de aplicații
2. Proceduri stocate și proceduri de aplicații
3. Pachete
4. Declanșatoare (triggeri) pe baza de date / de aplicații

Blocurile anonime:

- sunt nedenumite
- nu sunt stocate în baza de date
- se declară inline, în locul în care se dorește execuția lor
- se execută în momentul rulării

Exemplu:

```
DECLARE
    v_variabila varchar2(5);
BEGIN
    SELECT coloana
    INTO v_variabila
    FROM tabela;
    EXCEPTION
        WHEN excepție THEN acțiune
END;
/
```

Blocuri anonime imbricate

- se pot imbrica mai multe blocuri;
- acestea se pot eticheta cu <<eticheta_bloc>> , iar variabilele din cadrul blocurilor se pot utiliza astfel: eticheta_bloc.variabila.

```
BEGIN
.....
<< eticheta_bloc >>
DECLARE
    .....
BEGIN
    .....
END eticheta_bloc;
END;
/
```

Proceduri, funcții:

- blocuri PL/SQL cu un nume;
- se pot stoca la nivel de ORACLE SERVER(proceduri/funcții stocate) sau la nivel de aplicație (DEVELOPER – Forms si Reports).

Exemple:

```
CREATE [OR REPLACE] PROCEDURE nume_procedura
IS
.....
BEGIN
.....
[EXCEPTION]
.....
END;
/
```

```
CREATE [OR REPLACE] FUNCTION nume_funcție
RETURN tip_data
IS
BEGIN
.....
RETURN valoare
[EXCEPTION]
.....
END;
/
```

Pachete de programe - grupează proceduri, funcții.

Declanșatori pe baza de date - blocuri PL/SQL asociate tabelor (de bază sau virtuale) și lansate automat în execuție când are loc o comandă de manipulare.

Declanșatori de aplicație - blocuri PL/SQL asociate unor evenimente din cadrul aplicației (de exemplu: deplasarea mouse-ului, apăsarea unui buton) și lansate în execuție automat.

3. Operatori în PL/SQL

Operator	Caracteristici
+, -, *, /, ** (op. exponențial)	Operatori aritmetici
AND, OR, NOT	Operatori logici
<, >, =, >=, <=, <>, !=	Operatori de comparație
BETWEEN ... AND ...	Operator de verificare a apartenenței la un interval
IN(listă)	Operator de verificare a apartenenței la o listă de valori
LIKE	Operator de comparare cu un șablon % - oricâte caractere; _ - un caracter;
IS NULL	Operator care verifică dacă o variabilă are valoarea NULL

	Operator de concatenare
@	Operator de conectare la distanță
& sau &&	Operatori pentru adresarea variabilelor de substituție
:=	Operator de atribuire

4. Funcții SQL disponibile în PL/SQL

- În cadrul instrucțiunilor descriptive sunt suportate toate tipurile de funcții SQL (inclusiv funcțiile de grup în cadrul instrucțiunii SELECT);
- Instrucțiunile PL/SQL:
 - ✓ Suportă funcții la nivel de înregistrare (single-row): numerice, caracter, data, de conversie etc.;
 - ✓ **NU** suportă funcții de grup (SUM, MIN, MAX, AVG, COUNT, STDDEV) sau funcția DECODE. De exemplu nu se pot utiliza construcții de forma: *IF DECODE(...) THEN ...* sau *IF AVG(...) THEN ...*

5. Conversii în blocurile PL/SQL

- PL/SQL convertește tipurile de date dinamic (de exemplu: o valoare numerică la o variabilă char);
- conversii implicite: caracter <-> numeric și caracter <-> data;
- conversii explicite: se utilizează funcțiile TO_DATE, TO_NUMBER, TO_CHAR.

6. Comenzi SQL disponibile în PL/SQL

- PL/SQL permite folosirea comenzilor de manipulare a datelor (LMD):
 - ✓ SELECT
 - ✓ INSERT
 - ✓ UPDATE
 - ✓ DELETE
- PL/SQL permite folosirea comenzilor de control al tranzacțiilor:
 - ✓ COMMIT
 - ✓ ROLLBACK
 - ✓ SAVEPOINT

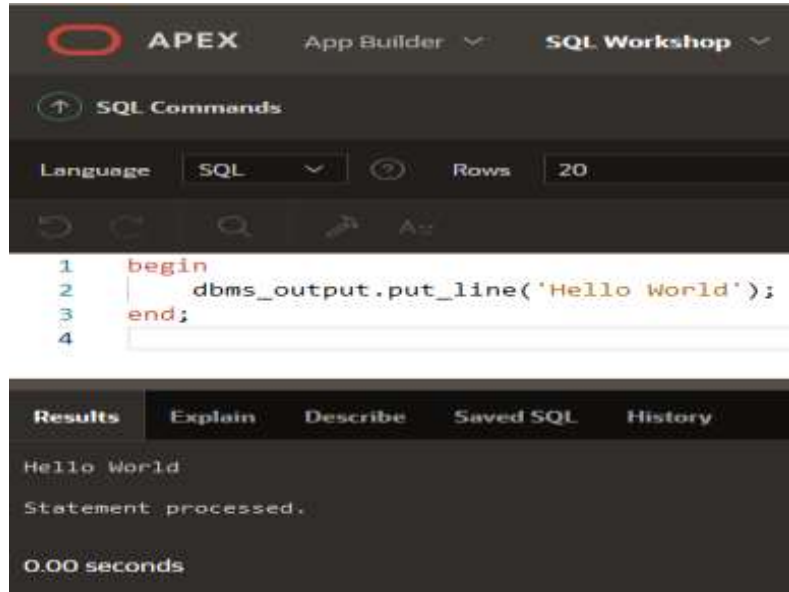
Notă: Un bloc PL/SQL nu este o tranzacție. Comenzile Commit / Rollback / Savepoint sunt independente de bloc, dar pot să apară în cadrul acestuia.

- PL/SQL **NU** acceptă comenzile de definire a datelor (LDD)
 - ✓ CREATE
 - ✓ ALTER
 - ✓ DROP
 - ✓ RENAME
 - ✓ TRUNCATE
- PL/SQL **NU** acceptă comenzile din cadrul limbajului pentru controlul datelor (Data Control Language - DCL)
 - ✓ GRANT
 - ✓ REVOKE

Probleme rezolvate

Specificați ce se va afișa la rularea următoarelor **programe (blocuri) PL/SQL**:

1)
begin
 dbms_output.put_line('Hello World');
end;



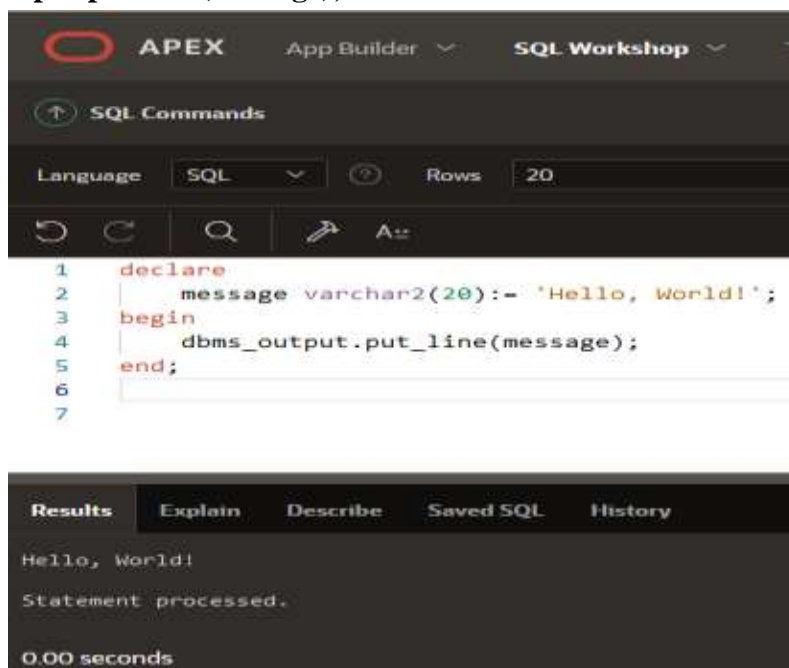
The screenshot shows the APEX SQL Workshop interface. The top bar includes the APEX logo, 'App Builder', and 'SQL Workshop'. Below this is the 'SQL Commands' section with a search icon and a 'Language' dropdown set to 'SQL'. The 'Rows' column is set to '20'. The main area contains a code editor with the following PL/SQL code:

```
1 begin
2     dbms_output.put_line('Hello World');
3 end;
```

Below the code editor is the 'Results' section, which is currently selected. It displays the output of the execution:

```
Hello World
Statement processed.
0.00 seconds
```

declare
 message varchar2(20):= 'Hello, World!';
begin
 dbms_output.put_line(message);
end;



The screenshot shows the APEX SQL Workshop interface. The top bar includes the APEX logo, 'App Builder', and 'SQL Workshop'. Below this is the 'SQL Commands' section with a search icon and a 'Language' dropdown set to 'SQL'. The 'Rows' column is set to '20'. The main area contains a code editor with the following PL/SQL code:

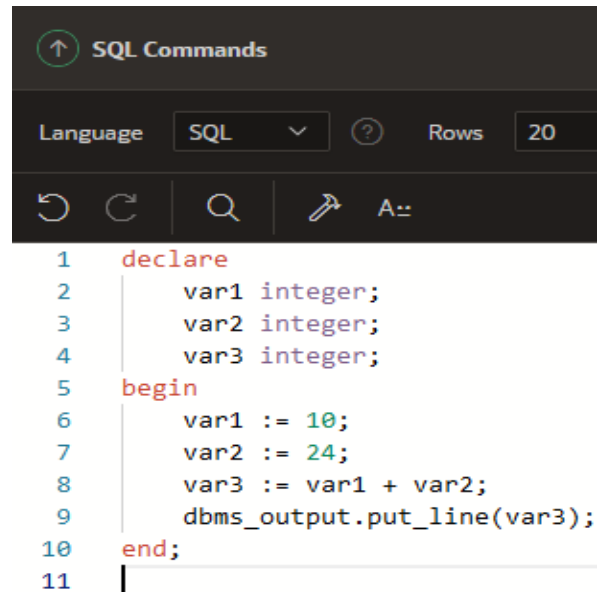
```
1 declare
2     message varchar2(20):= 'Hello, World!';
3 begin
4     dbms_output.put_line(message);
5 end;
```

Below the code editor is the 'Results' section, which is currently selected. It displays the output of the execution:

```
Hello, World!
Statement processed.
0.00 seconds
```

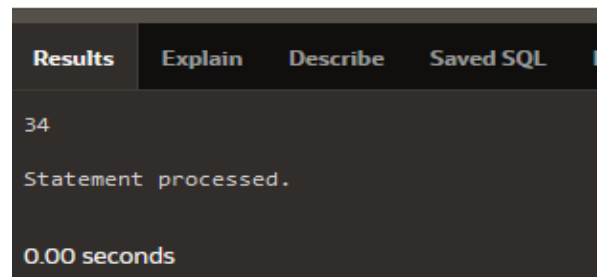
2) Program PL/SQL pentru adunarea a doua numere date.

```
declare
    var1 integer;
    var2 integer;
    var3 integer;
begin
    var1:=10;
    var2:=24;
    var3:=var1+var2;
    dbms_output.put_line(var3);
end;
```



The screenshot shows a dark-themed SQL editor window titled "SQL Commands". At the top, there is a "Language" dropdown menu set to "SQL" and a "Rows" counter set to "20". Below the menu are several icons: a refresh icon, a search icon, and a save icon. The main area of the window contains the following PL/SQL code:

```
1 declare
2     var1 integer;
3     var2 integer;
4     var3 integer;
5 begin
6     var1 := 10;
7     var2 := 24;
8     var3 := var1 + var2;
9     dbms_output.put_line(var3);
10 end;
11
```



The screenshot shows a dark-themed "Results" window. At the top, there are tabs for "Results", "Explain", "Describe", and "Saved SQL". The "Results" tab is active. The output shows the number "34" followed by the text "Statement processed." and "0.00 seconds".

```
34
Statement processed.
0.00 seconds
```

3) Program PL/SQL pentru interschimbarea a doua numere.

Avem doua solutii de interschimbare a doua numere

Solutia 1: cu o variabila temporara

```
declare
    a number;
    b number;
    temp number;
begin
    a:=5;
    b:=10;
    dbms_output.put_line('inainte de interschimbare:');
    dbms_output.put_line('a='||a||' b='||b);
    temp:=a;
    a:=b;
    b:=temp;
    dbms_output.put_line('dupa interschimbare:');
    dbms_output.put_line('a='||a||' b='||b);
end;
```

```
Language PL/SQL ? Rows 20 ? Clear
↶ ↷ 🔍 ↗ A::
1 declare
2     a number;
3     b number;
4     temp number;
5 begin
6     a:=5;
7     b:=10;
8     dbms_output.put_line('inainte de interschimbare:');
9     dbms_output.put_line('a='||a||' b='||b);
10    temp:=a;
11    a:=b;
12    b:=temp;
13    dbms_output.put_line('dupa interschimbare:');
14    dbms_output.put_line('a='||a||' b='||b);
15 end;
16
```

```
Results Explain Describe Saved SQL History
inainte de interschimbare:
a=5 b=10
dupa interschimbare:
a=10 b=5
Statement processed.
0.00 seconds
```

Solutia 2: Fara utilizarea unei variabile temporare

```
declare
    a number;
    b number;
begin
    a := 10;
    b := 5;
    dbms_output.put_line('inainte de interschimbare:');
    dbms_output.put_line('a = '||a||' b = '||b);
    a := a + b;
    b := a - b;
    a := a - b;
    dbms_output.put_line('dupa interschimbare:');
    dbms_output.put_line('a = '||a||' b = '||b);
end;
```

The screenshot shows the SQL Developer interface. At the top, there's a 'SQL Commands' header. Below it, the 'Language' is set to 'PL/SQL' and 'Rows' is set to '20'. The main area contains a PL/SQL script with the following code:

```
1 declare
2     a number;
3     b number;
4 begin
5     a := 10;
6     b := 5;
7     dbms_output.put_line('inainte de interschimbare:');
8     dbms_output.put_line('a = '||a||' b = '||b);
9     a := a + b;
10    b := a - b;
11    a := a - b;
12    dbms_output.put_line('dupa interschimbare:');
13    dbms_output.put_line('a = '||a||' b = '||b);
14 end;
15
```

Below the script, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the output of the script:

```
inainte de interschimbare:
a = 10 b = 5
dupa interschimbare:
a = 5 b = 10

Statement processed.

0.05 seconds
```

4) Program PL/SQL pentru a afla care este cel mai mare număr dintre 3(trei) valori date.

declare

a number := 10;

b number := 12;

c number := 5;

begin

dbms_output.put_line('a = '||a||' b = '||b||' c = '||c);

if a > b AND a > c then

dbms_output.put_line('a este cel mai mare');

else

if b > a AND b > c then

dbms_output.put_line('b este cel mai mare');


```

        else
            dbms_output.put_line('c cel mai mare');
        end if;
    end if;
end;

```

```

1  declare
2      a number := 10;
3      b number := 12;
4      c number := 5;
5  begin
6      dbms_output.put_line('a = ' || a || ' b = ' || b || ' c = ' || c);
7      if a > b AND a > c then
8          dbms_output.put_line('a este cel mai mare');
9      else
10         if b > a AND b > c then
11             dbms_output.put_line('b este cel mai mare');
12         else
13             dbms_output.put_line('c cel mai mare');
14         end if;
15     end if;
16 end;
17

```

```

Results Explain Describe Saved SQL History
a = 10 b = 12 c = 5
b este cel mai mare

Statement processed.

0.00 seconds

```

5) Program PL/SQL pentru verificarea primalitatii unui numar.

Un **numar este prim** daca este divizibil cu 1 si cu el insusi. De exemplu, numerele 2, 3, 5, 7, s.a.m.d. In schimb numerele 4, 6, 8, s.a.m.d. nu sunt numere prime.

```

declare
    n number;
    i number;
    flag number;
begin

```

```

i:=2;
flag:=1;
n:=7; -- exemplu pentru un numar prim
-- pentru un numar neprim se poate inlocui instructiunea de mai sus cu urmatoarea n:=8
for i in 2..n/2
loop
    if mod(n,i)=0
    then
        flag:=0;
        exit;
    end if;
end loop;
if flag=1
then
    dbms_output.put_line('numar prim');
else
    dbms_output.put_line('numar neprim');
end if;
end;

```

The screenshot shows a SQL IDE interface. At the top, there's a 'SQL Commands' header. Below it, the 'Language' is set to 'PL/SQL', and 'Rows' is set to '20'. There are buttons for 'Clear Command' and 'Find Tables'. The main area contains the PL/SQL code from the previous block, with line numbers 1 through 22. The code is color-coded: keywords in red, identifiers in blue, and literals in green. Below the code editor, there's a 'Results' tab selected, showing the output 'numar prim' and the message 'Statement processed.'.

```

1 declare
2     n number;
3     i number;
4     flag number;
5 begin
6     i:=2;
7     flag:=1;
8     n:=7; -- exemplu pentru un numar prim
9     -- pentru un numar neprim se poate inlocui instructiunea de mai sus cu urmatoarea n:=8
10 for i in 2..n/2
11     loop
12         if mod(n,i) = 0 then
13             flag:=0;
14             exit;
15         end if;
16     end loop;
17     if flag = 1 then
18         dbms_output.put_line('numar prim');
19     else
20         dbms_output.put_line('numar neprim');
21     end if;
22 end;

```

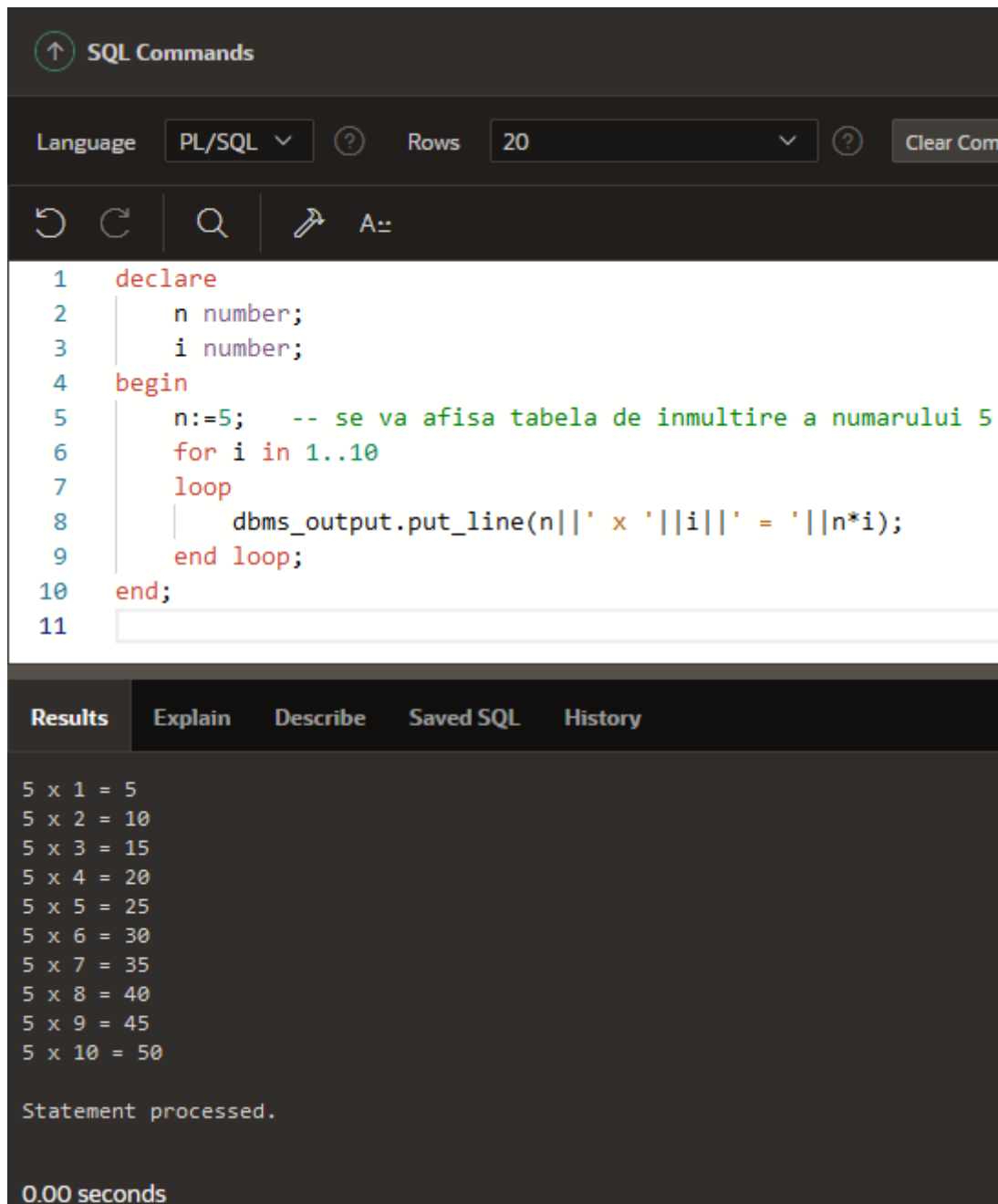
Results Explain Describe Saved SQL History

numar prim

Statement processed.

6) Program PL/SQL pentru afișarea tabelii de înmulțire a unui număr dat.

```
declare
    n number;
    i number;
begin
    n:=5; -- se va afisa tabela de inmultire a numarului 5
    for i in 1..10
    loop
        dbms_output.put_line(n||' x '||i||' = '||n*i);
    end loop;
end;
```



```
SQL Commands

Language PL/SQL Rows 20 Clear Com

1 declare
2     n number;
3     i number;
4 begin
5     n:=5; -- se va afisa tabela de inmultire a numarului 5
6     for i in 1..10
7     loop
8         dbms_output.put_line(n||' x '||i||' = '||n*i);
9     end loop;
10 end;
11

Results Explain Describe Saved SQL History

5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

Statement processed.

0.00 seconds
```

7) Program PL/SQL pentru calculul produsului tuturor numerelor mai mici decât numărul n dat.

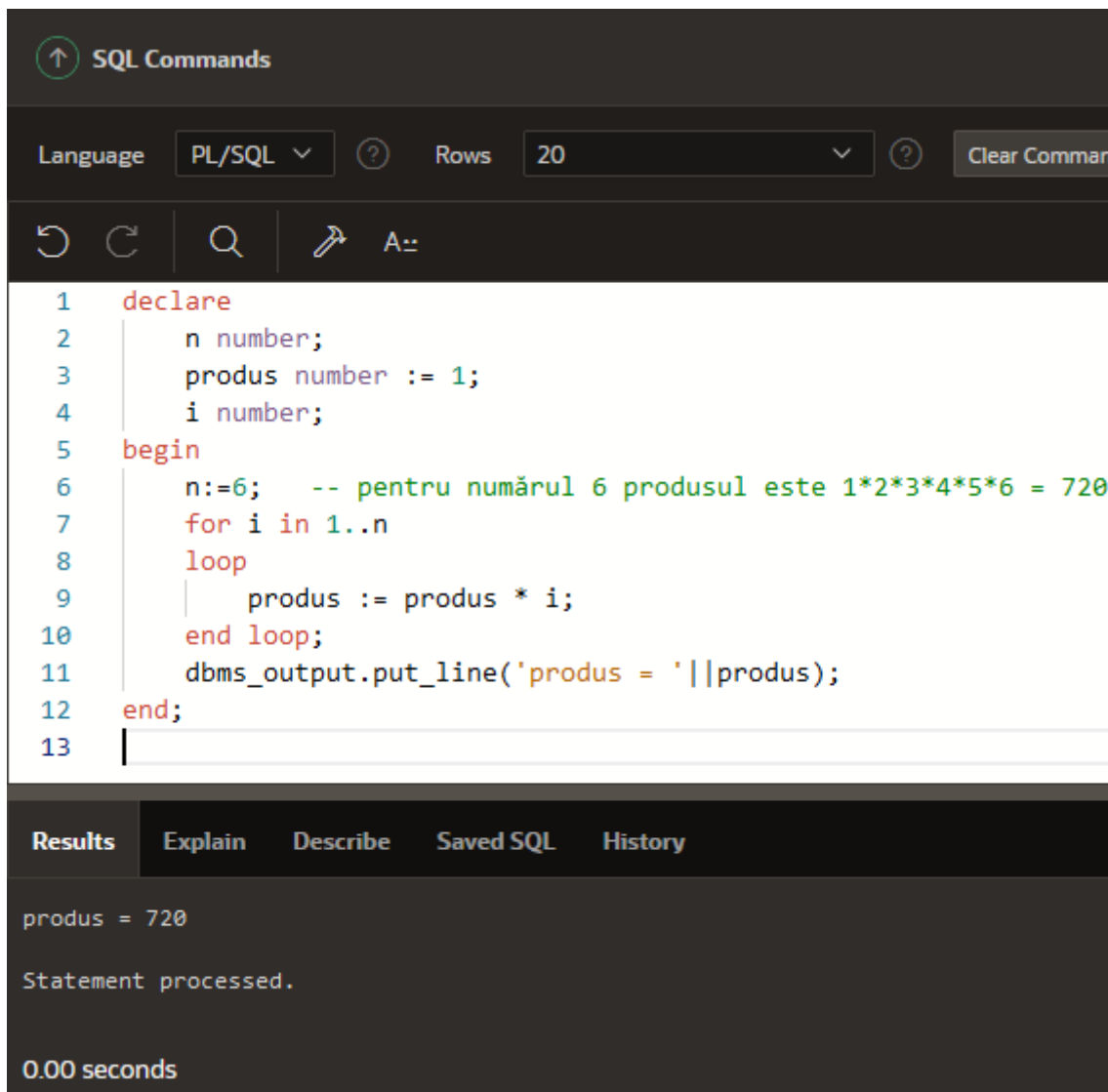
declare

```
n number;  
produs number := 1;  
i number;
```

begin

```
n:=6; -- pentru numărul 6 produsul este 1*2*3*4*5*6 = 720  
for i in 1..n  
loop  
    produs := produs * i;  
end loop;  
dbms_output.put_line('produs = '||produs);
```

end;



The screenshot shows an SQL IDE interface. At the top, there's a title bar 'SQL Commands'. Below it, a toolbar contains 'Language' set to 'PL/SQL', 'Rows' set to '20', and a 'Clear Command' button. The main area displays the PL/SQL code from the previous blocks, with line numbers 1 through 13. The code is color-coded: keywords in red, variables in blue, and comments in green. Below the code editor, there's a 'Results' tab selected, showing the output 'produs = 720' and the message 'Statement processed.' at the bottom, along with the execution time '0.00 seconds'.

```
1 declare  
2     n number;  
3     produs number := 1;  
4     i number;  
5 begin  
6     n:=6; -- pentru numărul 6 produsul este 1*2*3*4*5*6 = 720  
7     for i in 1..n  
8     loop  
9         produs := produs * i;  
10    end loop;  
11    dbms_output.put_line('produs = '||produs);  
12 end;  
13
```

Results Explain Describe Saved SQL History

produs = 720


Statement processed.

0.00 seconds

8) Program PL/SQL pentru afișarea următoarei figuri:

```
*  
**  
***  
****  
*****
```

```
declare  
    n number:=5;  
    i number;  
    j number;  
begin  
    for i in 1..n  
    loop  
        for j in 1..i  
        loop  
            dbms_output.put('*');  
        end loop;  
        dbms_output.new_line;  
    end loop;  
end;
```



The screenshot shows the SQL Developer interface. The top panel is titled "SQL Commands" and shows the language set to "PL/SQL" and "Rows" set to "20". The main editor contains the following PL/SQL code:

```
1 declare  
2     n number:=5;  
3     i number;  
4     j number;  
5 begin  
6     for i in 1..n  
7     loop  
8         for j in 1..i  
9         loop  
10            dbms_output.put('*');  
11            end loop;  
12            dbms_output.new_line;  
13        end loop;  
14    end;  
15
```

The bottom panel shows the "Results" tab with the following output:

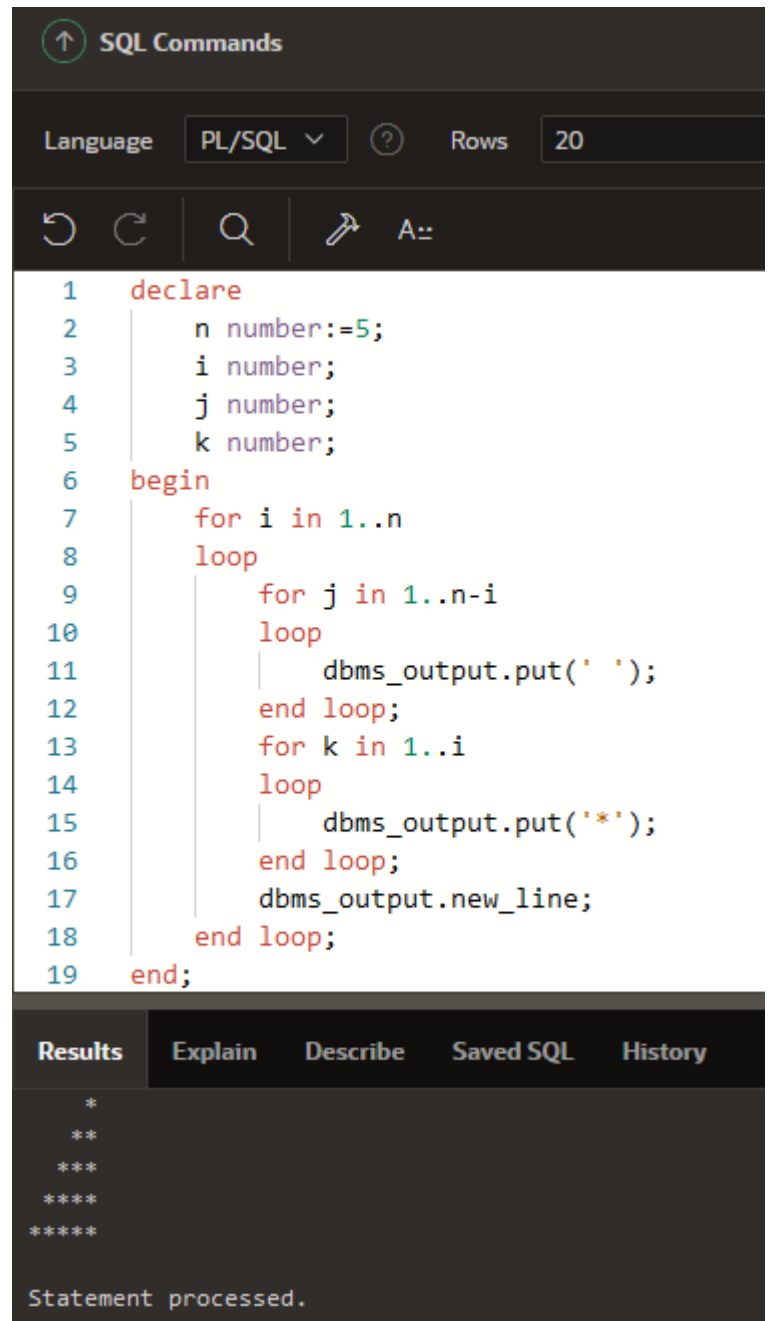
```
*  
**  
***  
****  
*****
```

Below the output, it states "Statement processed." and "0.00 seconds".

9) Program PL/SQL pentru afișarea următoarei figuri:

```
*  
**  
***  
****  
*****
```

```
declare  
    n number:=5;  
    i number;  
    j number;  
    k number;  
begin  
    for i in 1..n  
    loop  
        for j in 1..n-i  
        loop  
            dbms_output.put(' ');  
        end loop;  
        for k in 1..i  
        loop  
            dbms_output.put('*');  
        end loop;  
        dbms_output.new_line;  
    end loop;  
end;
```



The screenshot shows the SQL Developer interface. At the top, the title bar reads "SQL Commands". Below it, the "Language" dropdown is set to "PL/SQL", and the "Rows" field is set to "20". The main editor area contains the following PL/SQL code:

```
1 declare  
2     n number:=5;  
3     i number;  
4     j number;  
5     k number;  
6 begin  
7     for i in 1..n  
8     loop  
9         for j in 1..n-i  
10        loop  
11            dbms_output.put(' ');  
12        end loop;  
13        for k in 1..i  
14        loop  
15            dbms_output.put('*');  
16        end loop;  
17        dbms_output.new_line;  
18    end loop;  
19 end;
```

Below the editor, there are tabs for "Results", "Explain", "Describe", "Saved SQL", and "History". The "Results" tab is active, displaying the output of the PL/SQL code:

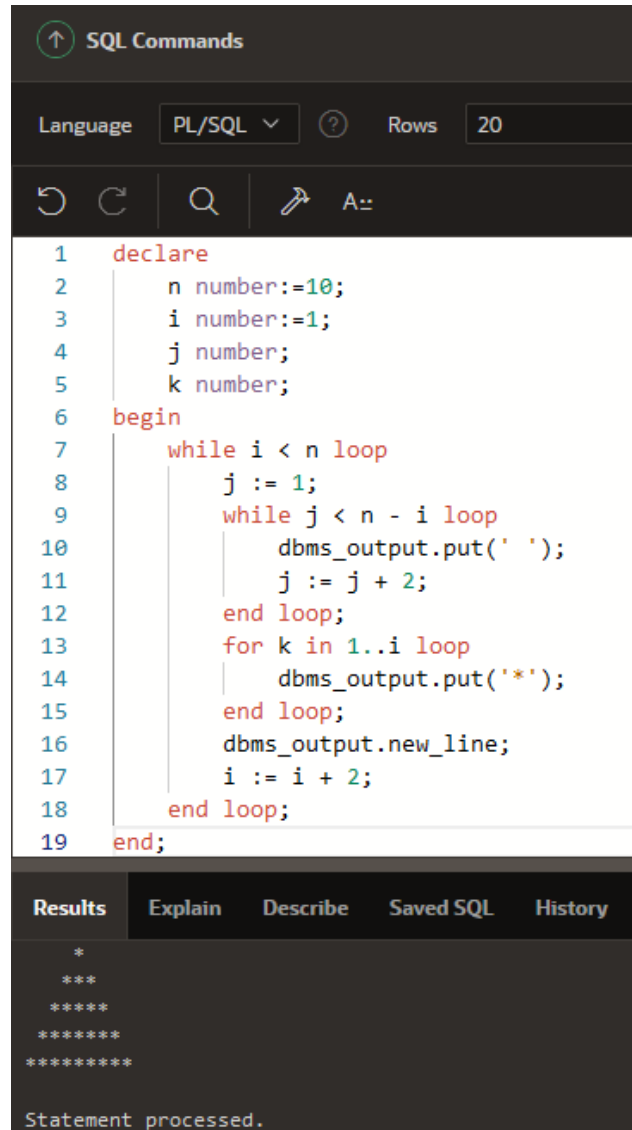
```
*  
**  
***  
****  
*****
```

At the bottom of the Results tab, the text "Statement processed." is visible.

10) Program PL/SQL pentru afișarea următoarei figuri:

```
*
***
*****
*****
*****
```

```
declare
    n number:=10;
    i number:=1;
    j number;
    k number;
begin
    while i<n loop
        j:=1;
        while j<n-i loop
            dbms_output.put(' ');
            j:=j+2;
        end loop;
        for k in 1..i
        loop
            dbms_output.put('*');
        end loop;
        dbms_output.new_line;
        i:=i+2;
    end loop;
end;
```



The screenshot shows the SQL Developer interface. The top bar indicates the language is PL/SQL and the number of rows is 20. The main editor displays the PL/SQL code from the previous block, with line numbers 1 through 19. Below the editor, the 'Results' tab is active, showing the output of the program: a pattern of asterisks forming a right-angled triangle. The output is: *

```
***
*****
*****
*****
```

At the bottom of the results window, the text 'Statement processed.' is visible.

11) Program PL/SQL pentru afișarea următoarei figuri:

```

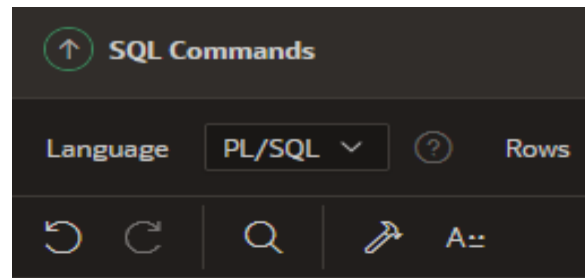
*
***
*****
*****
*****
*****
*****
***
*

```

```

declare
    n number:=10;
    i number:=1;
    j number;
    k number;
begin
    while i < n loop
        j := 1;
        while j < n - i loop
            dbms_output.put(' ');
            j := j + 2;
        end loop;
        for k in 1..i loop
            dbms_output.put('*');
        end loop;
        dbms_output.new_line;
        i := i + 2;
    end loop;
    i := 7;
    while i >= 1 loop
        j := 9;
        while j > i loop
            dbms_output.put(' ');
            j := j - 2;
        end loop;
        for k in 1..i loop
            dbms_output.put('*');
        end loop;
        dbms_output.new_line;
        i := i - 2;
    end loop;
end;

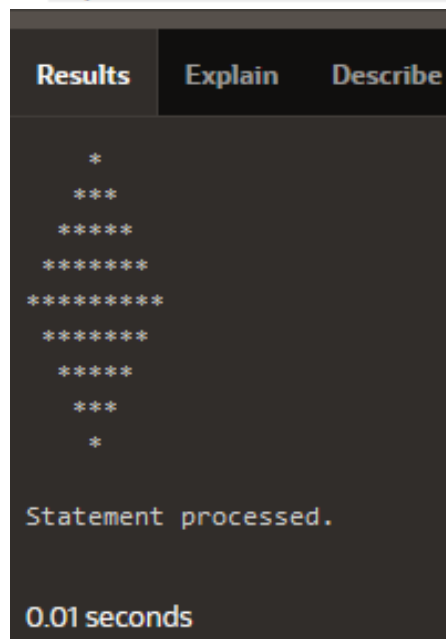
```



```

1 declare
2     n number:=10;
3     i number:=1;
4     j number;
5     k number;
6 begin
7     while i < n loop
8         j := 1;
9         while j < n - i loop
10            dbms_output.put(' ');
11            j := j + 2;
12        end loop;
13        for k in 1..i loop
14            dbms_output.put('*');
15        end loop;
16        dbms_output.new_line;
17        i := i + 2;
18    end loop;
19    i := 7;
20    while i >= 1 loop
21        j := 9;
22        while j > i loop
23            dbms_output.put(' ');
24            j := j - 2;
25        end loop;
26        for k in 1..i loop
27            dbms_output.put('*');
28        end loop;
29        dbms_output.new_line;
30        i := i - 2;
31    end loop;
32 end;

```



Probleme propuse spre rezolvare

Scrieți programe PL/SQL (blocuri anonime) pentru următoarele cerințe:

- 1) Sa se calculeze diferența a doua numere întregi. De exemplu: primul număr este 87 si cel de-al doilea număr este 34.
- 2) Sa se afișeze care este cel mai mic număr dintre 2(doua) numere întregi date. De exemplu: primul număr este 123, iar al doilea număr este 11.
- 3) Sa se verifice daca un număr întreg dat este sau număr par. De exemplu numărul 34.
- 4) Sa se afișeze care este cel mai mic număr dintre 3(trei) numere întregi date. De exemplu: primul număr este 323, al doilea număr 131, iar al treilea număr este 1234.
- 5) Sa se afișeze tabela de adunare a unui număr dat. De exemplu, pentru numărul 8.
- 6) Sa calculeze suma primelor n numere întregi mai mici decât n. De exemplu, pentru numărul $n=6$, suma este egala cu $1+2+3+4+5+6 = 21$
- 7) Sa se afișeze următoarea figură:

```
*****  
*****  
***  
**  
*
```

- 8) Sa se afișeze următoarea figură:

```
*****  
  ****  
   ***  
    **  
     *
```

- 9) Sa se afișeze următoarea figură:

```
*****  
  *****  
   *****  
    ***  
     *
```

- 10) Sa se afișeze următoarea figură:

```
*****  
  *****  
   *****  
    ***  
     *  
      *  
     ***  
    *****  
   *****
```