

## Laborator 8: SUBPROGRAME in PL/SQL (partea I - proceduri)

Un *subprogram este un bloc PL/SQL cu nume* (spre deosebire de blocurile anonime) care poate primi parametri și poate fi invocat dintr-un anumit mediu ( de exemplu, SQL\*Plus, Oracle Forms, Oracle Reports etc.)

Subprogramele sunt bazate pe structura de bloc PL/SQL. Similar, ele conțin o parte declarativă opțională, o parte executabilă obligatorie și o parte de tratare de excepții opțională.

Exista 2 tipuri de subprograme:

1. proceduri;
2. funcții (trebuie să conțină cel puțin o comandă RETURN);

Subprogramele pot fi:

- o **locale** (în cadrul altui bloc PL/SQL sau subprogram)
- o **stocate** (create cu comanda CREATE) - odată create, procedurile și funcțiile sunt stocate în baza de date, motiv pentru care se numesc subprograme stocate.

Sintaxa simplificată pentru crearea unei proceduri este următoarea:

```
[CREATE [OR REPLACE] ] PROCEDURE nume_procedură  
[ (lista_parametri) ]  
  {IS | AS}  
  [declarații locale]  
BEGIN  
  partea_executabilă  
[EXCEPTION  
  partea_de_tratare_a_excepțiilor]  
END [nume_procedură];
```

Lista de parametri conține specificații de parametri separate prin virgulă de forma:

**nume\_parametru mod\_parametru tip\_parametru;**

- mod\_parametru specifică dacă parametrul este:
  - ✓ de intrare (IN) – singurul care poate avea o valoare inițială
  - ✓ de intrare / ieșire (IN OUT)
  - ✓ de ieșire (OUT)
- mod\_parametru are valoarea implicită IN.

În cazul în care se modifică un obiect (vizualizare, tabel etc) de care depinde un subprogram, acesta este invalidat. Revalidarea se face fie prin recrearea subprogramului fie prin comanda:

**ALTER PROCEDURE nume\_proc COMPILE;**

Ștergerea unei proceduri se realizează prin comenzile:

**DROP PROCEDURE nume\_proc;**

Informații despre procedurile deținute de utilizatorul curent se pot obține interogând vizualizarea USER\_OBJECTS din dicționarul datelor.

```
SELECT OBJECT_NAME, OBJECT_TYPE, STATUS
FROM USER_OBJECTS
WHERE OBJECT_TYPE IN ('PROCEDURE');
```

Obs: STATUS – starea subprogramului (validă sau invalidă).

Codul complet al unui subprogram poate fi vizualizat folosind următoarea sintaxă:

```
SELECT TEXT
FROM USER_SOURCE
WHERE NAME = 'nume_subprogram'
ORDER BY LINE;
```

Eroarea apărută la compilarea unui subprogram poate fi vizualizată folosind următoarea sintaxă:

```
SELECT LINE, POSITION, TEXT
FROM USER_ERRORS
WHERE NAME = 'nume';
```

Erorile pot fi vizualizate și prin intermediul comenzii SHOW ERRORS.

Descrierea specificației unui subprogram se face prin comanda DESCRIBE.

Când este apelată o procedură **PL/SQL**, sistemul **Oracle** furnizează două metode pentru definirea parametrilor actuali:

- specificarea explicită prin nume
- specificarea prin poziție

Exemplu:

**subprogram(a tip\_a, b tip\_b, c tip\_c, d tip\_d)**

a) specificare prin poziție: **subprogram(var\_a, var\_b, var\_c, var\_d);**

b) specificare prin nume: **subprogram(b=>var\_b, c=>var\_c, d=>var\_d, a=>var\_a);**

c) specificare prin nume și poziție: **subprog(var\_a, var\_b, d=>var\_d, c=>var\_c);**

## Probleme rezolvate

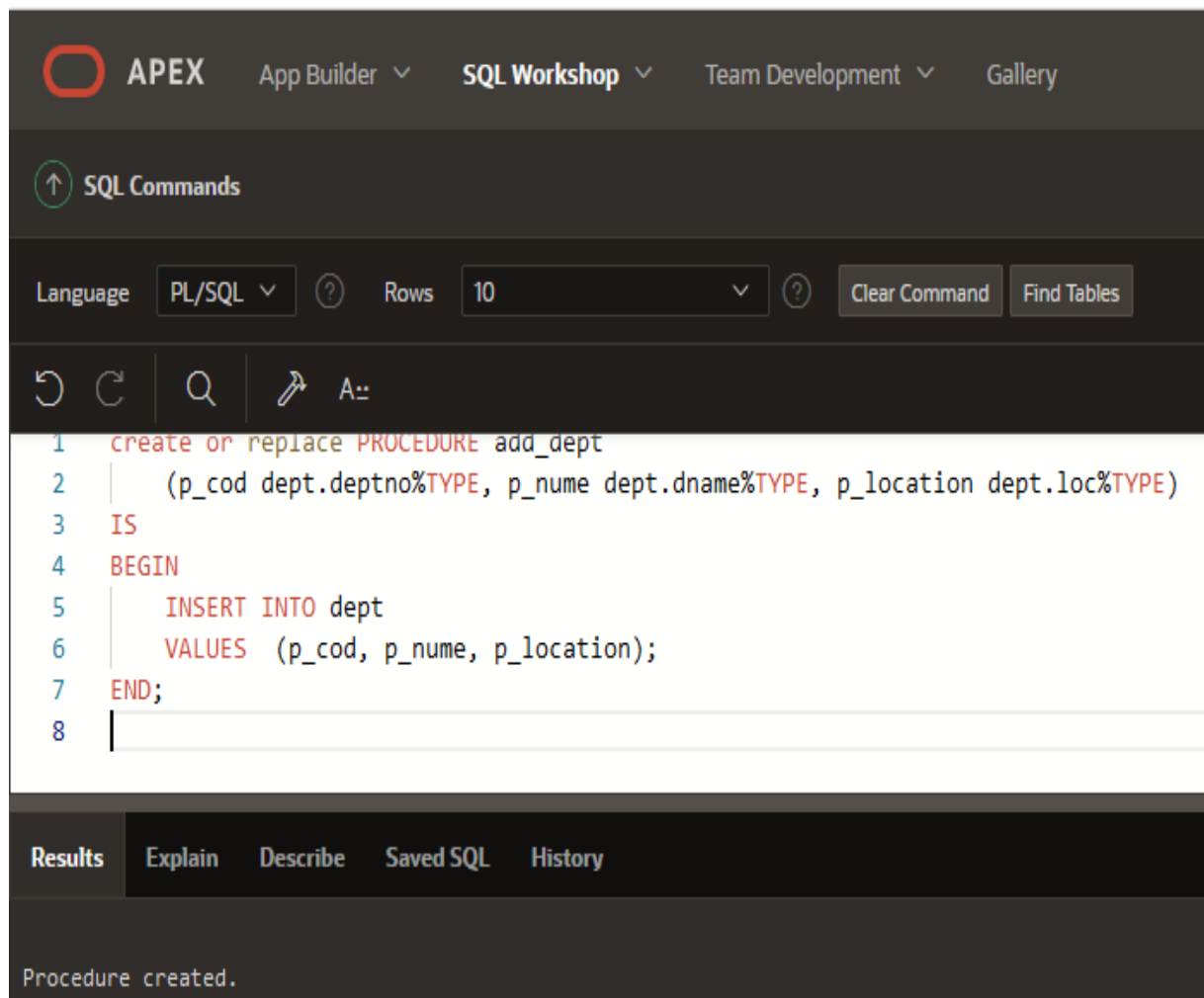
### a) Proceduri locale

1. Să se declare o procedură locală într-un bloc PL/SQL anonim prin care să se introducă în tabelul DEPT o nouă înregistrare precizând, prin intermediul parametrilor, valori pentru toate câmpurile. Invocați procedura în cadrul blocului.

Soluție:

Pasul 1 - se creează procedura **add\_dept** și se execută codul scris în aceasta:

```
create or replace PROCEDURE add_dept
    (p_cod dept.deptno%TYPE, p_nume dept.dname%TYPE, p_location
dept.loc%TYPE)
IS
BEGIN
    INSERT INTO dept
    VALUES (p_cod, p_nume, p_location);
END;
```



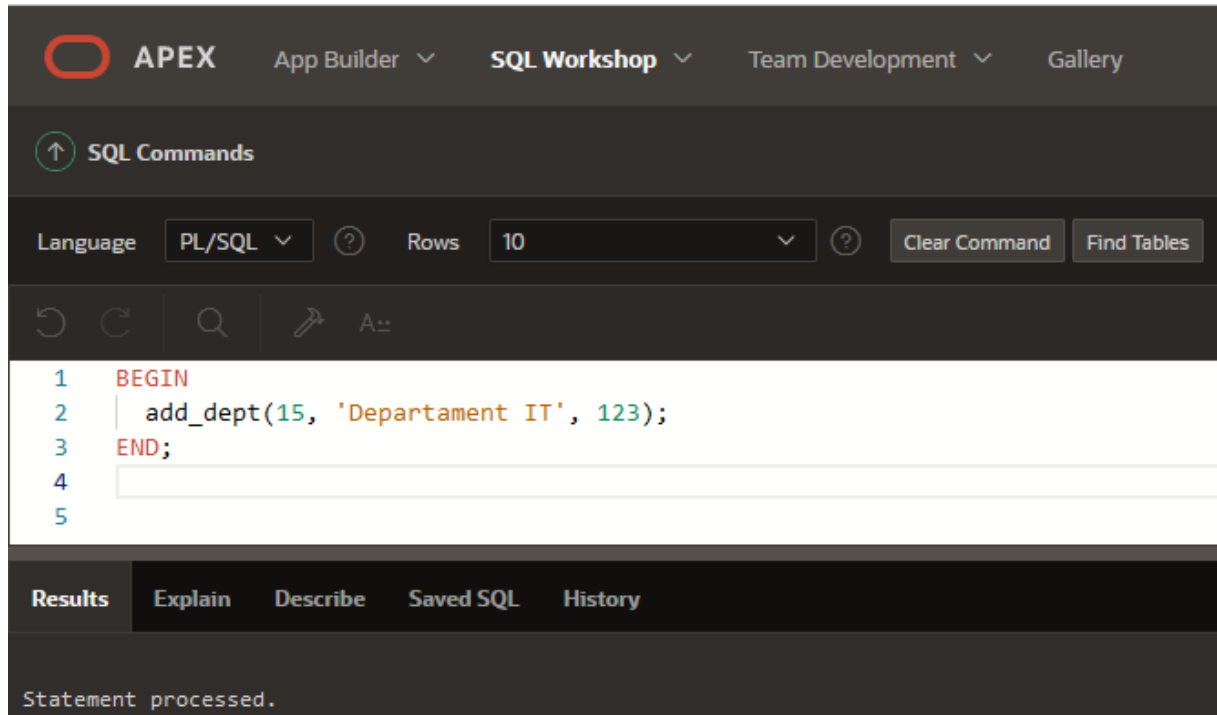
The screenshot displays the APEX SQL Workshop interface. At the top, there are navigation tabs for 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this, the 'SQL Commands' section is active, showing a 'Language' dropdown set to 'PL/SQL' and 'Rows' set to '10'. The main area contains the following PL/SQL code:

```
1 create or replace PROCEDURE add_dept
2 | (p_cod dept.deptno%TYPE, p_nume dept.dname%TYPE, p_location dept.loc%TYPE)
3 IS
4 BEGIN
5 |     INSERT INTO dept
6 |     VALUES (p_cod, p_nume, p_location);
7 END;
8 |
```

At the bottom, the 'Results' tab is selected, displaying the message 'Procedure created.'

Pas 2 - se creeaza un bloc anonim care apeleaza procedura creata anterior si de observa efectul executiei:

```
BEGIN  
  add_dept(15, 'Departament IT', 123);  
END;
```

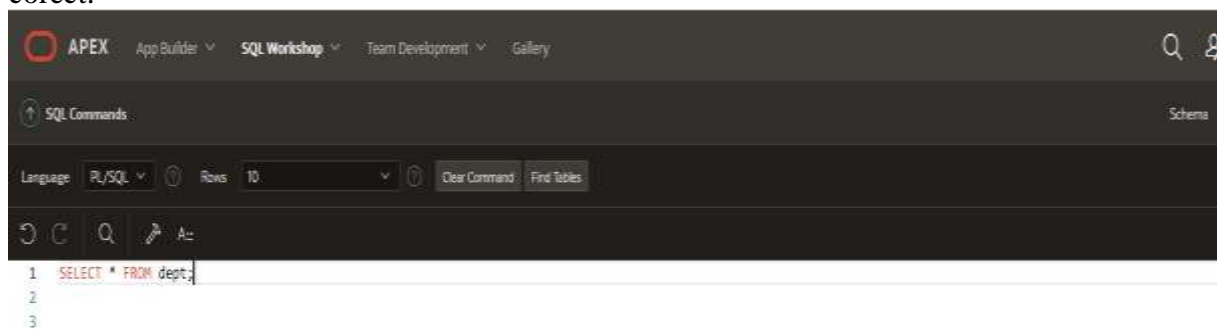


The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is the 'SQL Commands' section with a 'Language' dropdown set to 'PL/SQL', 'Rows' set to '10', and buttons for 'Clear Command' and 'Find Tables'. The main editor area contains the following PL/SQL code:

```
1 BEGIN  
2   add_dept(15, 'Departament IT', 123);  
3 END;  
4  
5
```

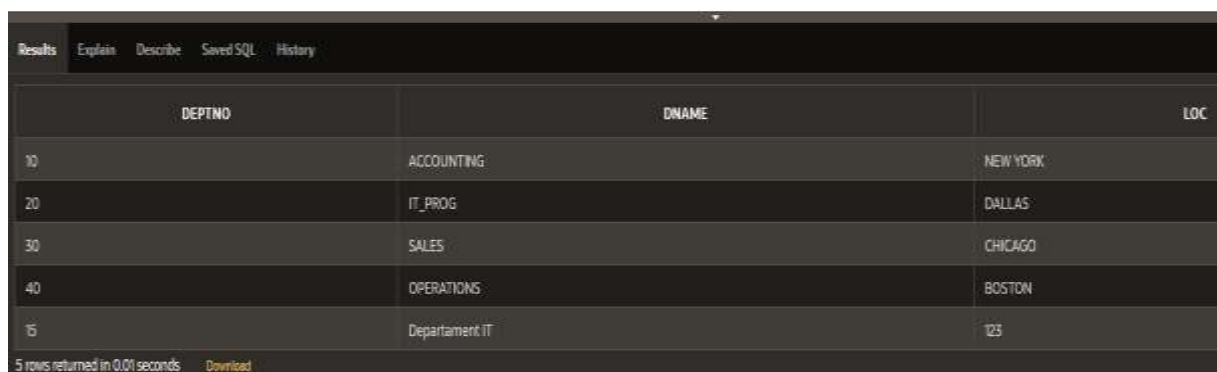
Below the editor are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, displaying the message 'Statement processed.'

Pasul 3 - se afiseaza continutul tabelii DEPT pentru a verifica daca procedura a functionat corect:



The screenshot shows the APEX SQL Workshop interface. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is the 'SQL Commands' section with a 'Language' dropdown set to 'PL/SQL', 'Rows' set to '10', and buttons for 'Clear Command' and 'Find Tables'. The main editor area contains the following SQL query:

```
1 SELECT * FROM dept;  
2  
3
```



The screenshot shows the 'Results' tab of the APEX SQL Workshop. It displays a table with the following data:

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	IT_PROG	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
15	Departament IT	123

At the bottom of the results area, it states '5 rows returned in 0.01 seconds' and includes a 'Download' button.

2. Să se declare o procedură locală care are parametrii următori:

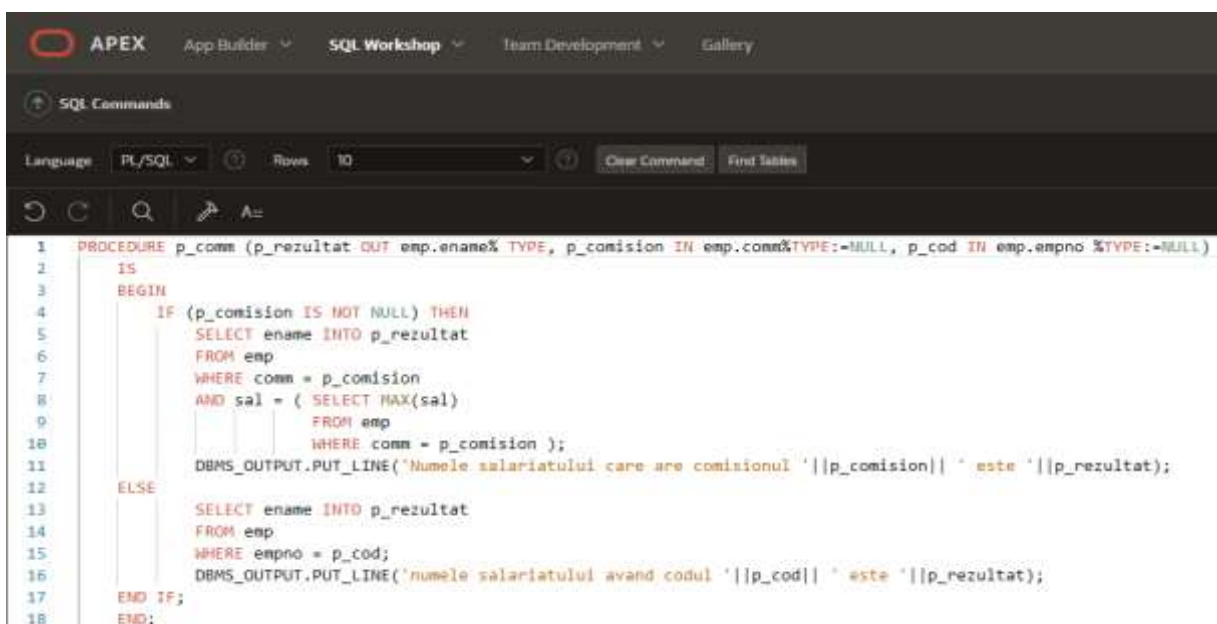
- p\_rezultat (parametru de tip OUT) de tipul coloanei ename din tabelul EMP;
- p\_comision (parametru de tip IN) de tipul coloanei commdin EMP, inițializat cu NULL;
- p\_cod (parametru de tip IN) de tipul coloanei emp\_id din EMP, inițializat cu NULL.

Dacă p\_comision nu este NULL atunci în p\_rezultat se va memora numele salariatului care are salariul maxim printre salariații având comisionul respectiv. În caz contrar, în p\_rezultat se va memora numele salariatului al cărui cod are valoarea dată la apelarea procedurii.

Soluție:

Pasul 1 - se creeaza procedura **add\_dept** si se executa codul scris in aceasta:

```
PROCEDURE p_comm (p_rezultat OUT emp.ename% TYPE,
                  p_comision IN  emp.comm%TYPE:=NULL,
                  p_cod      IN  emp.empno %TYPE:=NULL)
IS
BEGIN
    IF (p_comision IS NOT NULL) THEN
        SELECT ename INTO p_rezultat
        FROM emp
        WHERE comm = p_comision
        AND sal = (SELECT MAX(sal)
                  FROM emp
                  WHERE comm= p_comision);
        DBMS_OUTPUT.PUT_LINE('Numele salariatului care are comisionul
'||p_comision|| ' este '||p_rezultat);
    ELSE
        SELECT ename INTO p_rezultat
        FROM emp
        WHERE empno = p_cod;
        DBMS_OUTPUT.PUT_LINE('numele salariatului avand codul '||p_cod|| '
este '||p_rezultat);
    END IF;
END;
```

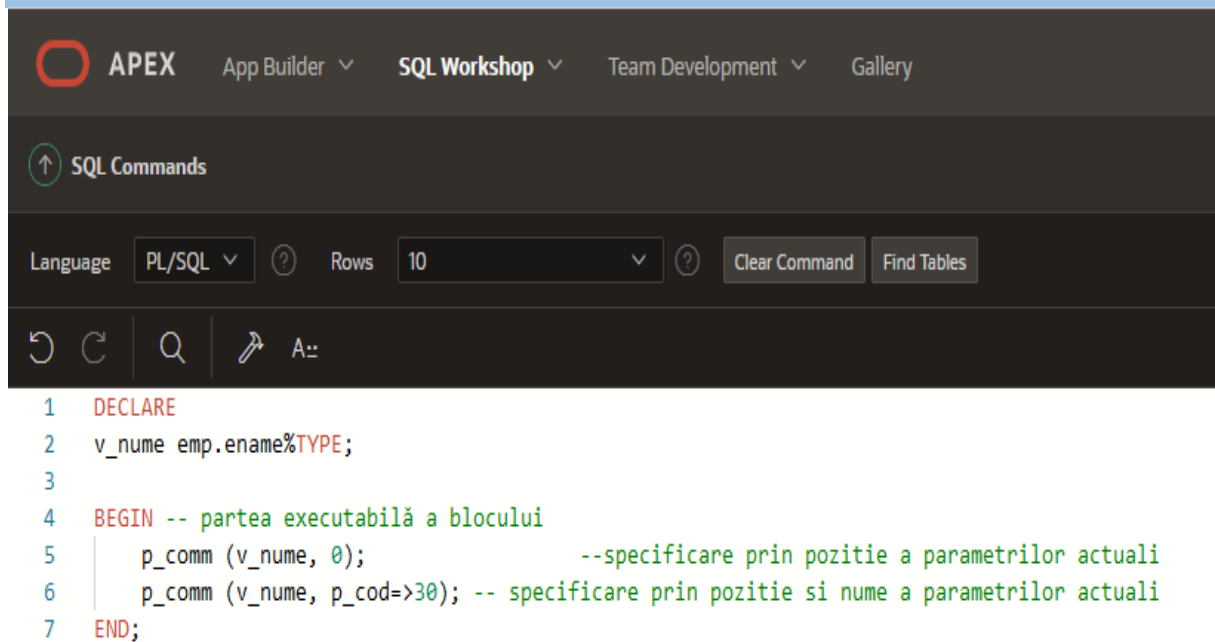


The screenshot shows the APEX SQL Workshop interface. At the top, there are navigation tabs for 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below the tabs, there is a 'SQL Commands' section with a language dropdown set to 'PL/SQL' and a 'Rows' dropdown set to '10'. The main area displays the PL/SQL code for the p\_comm procedure, which is identical to the code provided in the previous block. The code is highlighted in a light blue color, and the line numbers are visible on the left side of the editor.

Pas 2 – se creeaza un bloc anonim care apeleaza procedura creata anterior si de observa efectul executiei:

```
DECLARE
v_nume emp.ename%TYPE;

BEGIN -- partea executabilă a blocului
    p_comm (v_nume, 0);          --specificare prin pozitie a parametrilor actuali
    p_comm (v_nume, p_cod=>30); -- specificare prin pozitie si nume a parametrilor actuali
END;
```



The screenshot shows the APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop, Team Development, and Gallery. Below the tabs is a section for SQL Commands. The interface includes a Language dropdown set to PL/SQL, a Rows dropdown set to 10, and buttons for Clear Command and Find Tables. The main area displays the PL/SQL code from the previous block, with line numbers 1 through 7. The code is: 1 DECLARE, 2 v\_nume emp.ename%TYPE;, 3, 4 BEGIN -- partea executabilă a blocului, 5 p\_comm (v\_nume, 0); --specificare prin pozitie a parametrilor actuali, 6 p\_comm (v\_nume, p\_cod=>30); -- specificare prin pozitie si nume a parametrilor actuali, 7 END;.

3. Să se creeze o procedură stocată fără parametri care afișează un mesaj “Programare PL/SQL”, ziua de astăzi în formatul DD-MONTH-YYYY și ora curentă, precum și ziua de ieri în formatul DD-MON-YYYY.

Soluție:

```
CREATE OR REPLACE PROCEDURE p_today IS
    azi DATE := SYSDATE;
    ieri azi%TYPE;
BEGIN
    DBMS_OUTPUT.PUT_LINE('Programare PL/SQL') ;
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(azi, 'dd-month-yyyy hh24:mi:ss'));
    ieri := azi - 1;
    DBMS_OUTPUT.PUT_LINE(TO_CHAR(ieri, 'dd-mon-yyyy'));
END;
```

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables

```
1 CREATE OR REPLACE PROCEDURE P_today IS
2     azi DATE := SYSDATE;
3     ieri azi%TYPE;
4 BEGIN
5     DBMS_OUTPUT.PUT_LINE('Programare PL/SQL') ;
6     DBMS_OUTPUT.PUT_LINE(TO_CHAR(azi, 'dd-month-yyyy hh24:mi:ss'));
7     ieri := azi - 1;
8     DBMS_OUTPUT.PUT_LINE(TO_CHAR(ier, 'dd-mon-yyyy'));
9 END;
```

Results Explain Describe Saved SQL History

Procedure created.

**BEGIN** -- partea executabilă a blocului  
    **p\_today**;  
**END**;

APEX App Builder SQL Workshop Team Development Gallery

SQL Commands

Language PL/SQL Rows 10 Clear Command Find Tables

```
1 -- partea executabilă a blocului
2 BEGIN
3     p_today;
4 END;
```

Results Explain Describe Saved SQL History

Programare PL/SQL  
30-november -2021 13:15:47  
29-nov-2021

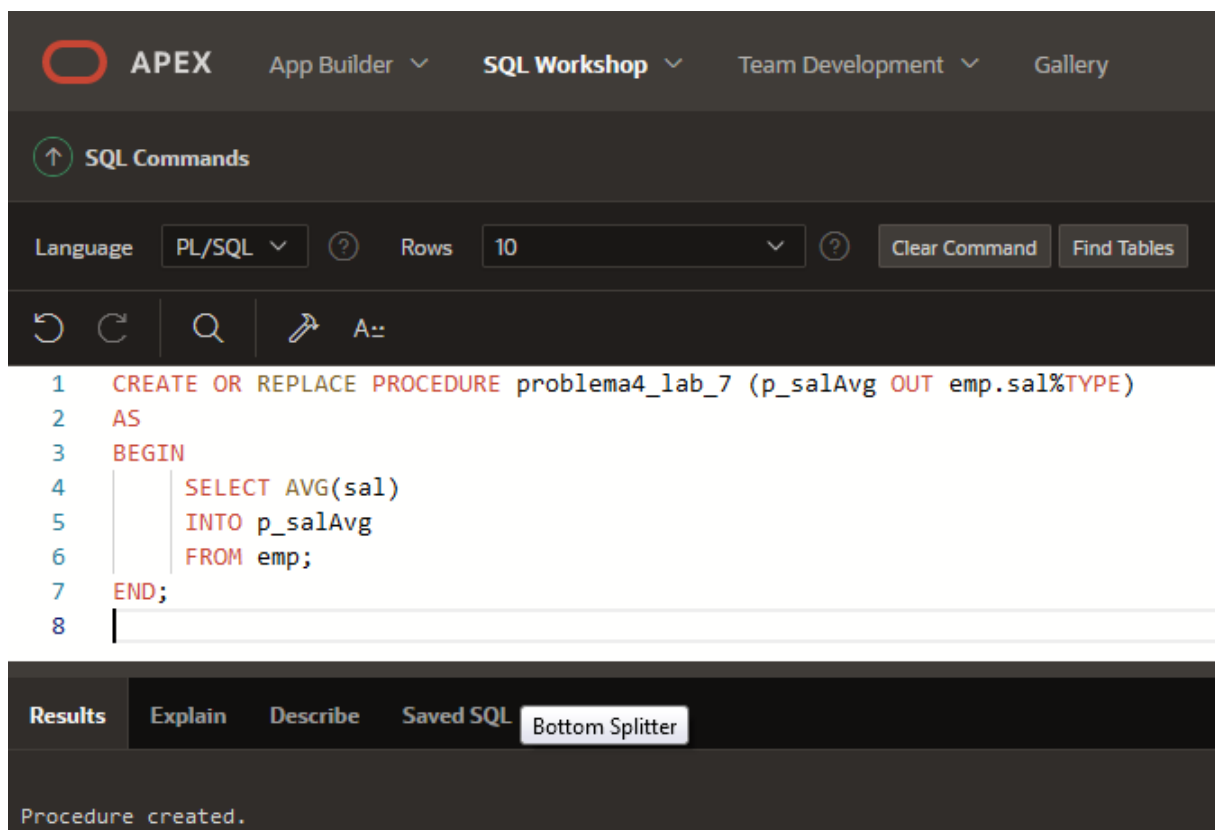
Statement processed.

4. a) Să se creeze o procedură stocată care calculează salariul mediu al angajaților, returnându-l prin intermediul unui parametru de tip OUT.

b) Să se apeleze procedura regăsind valoarea medie a salariilor într-o variabilă gazdă. Afișați valoarea variabilei.

Soluție:

```
CREATE OR REPLACE PROCEDURE problema4_pnu (p_salAvg OUT
emp.sal%TYPE)
AS
BEGIN
    SELECT AVG(sal)
    INTO p_salAvg
    FROM emp;
END;
```



The screenshot displays the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs for 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. Below this is a 'SQL Commands' section with a search icon and a list of commands. The 'Language' is set to 'PL/SQL' and 'Rows' is set to '10'. There are buttons for 'Clear Command' and 'Find Tables'. The main area shows the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE problema4_lab_7 (p_salAvg OUT emp.sal%TYPE)
2 AS
3 BEGIN
4     SELECT AVG(sal)
5     INTO p_salAvg
6     FROM emp;
7 END;
8
```

Below the code editor, there are tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'Bottom Splitter'. The 'Results' tab is active, showing the message 'Procedure created.'

```
DECLARE g_medie NUMBER;
BEGIN -- partea executabilă a blocului
    problema4_lab_7 (g_medie);
    DBMS_OUTPUT.PUT_LINE(g_medie);
END;
```

The screenshot shows the Oracle APEX SQL Workshop interface. At the top, there are navigation tabs: APEX, App Builder, SQL Workshop, Team Development, and Gallery. Below the tabs, there is a section for SQL Commands. The Language is set to PL/SQL, and the number of rows to display is 10. There are buttons for 'Clear Command' and 'Find Tables'. Below the command area, there are icons for undo, redo, search, and a command prompt. The main area contains the following PL/SQL code:

```
1 DECLARE g_medie NUMBER;
2 -- partea executabilă a blocului
3 BEGIN
4     problema4_lab_7(g_medie);
5     DBMS_OUTPUT.PUT_LINE(g_medie);
6 END;
7
```

Below the code, there is a 'Results' tab with sub-tabs for 'Results', 'Explain', 'Describe', 'Saved SQL', and 'History'. The 'Results' tab is active, showing the output '3179.96' and the message 'Statement processed.'

5. a) Creați o procedură numită GET\_EMP\_pnu care întoarce salariul și denumirea job-ului pentru un angajat al cărui cod este transmis ca parametru.
  - b) Executați procedura utilizând câte o variabilă gazdă pentru cei doi parametri de tip OUT.
  - c) testați procedura atât pentru coduri existente cât și pentru coduri inexistente din tabelul EMP\_pnu. Ce se întâmplă atunci când o invocăm pentru un cod inexistent ?
  
6. a) Analizați subprogramele create anterior în vizualizarea din dicționarul datelor USER\_OBJECTS.
  - b) Regăsiți codul unuia dintre subprogramele create anterior în vizualizarea USER\_SOURCE.
  - c) Aflați tipul parametrilor unuia dintre procedurile create anterior utilizând comanda DESCRIBE.

## Probleme propuse spre rezolvare

1. Să se creeze o procedură stocată care mărește salariile angajaților care nu au comision și au media mai mică decât cea a departamentului în care lucrează cu o valoare transmisă ca parametru.

2. Să se declare o procedură locală care are următorii parametri:

- p\_rezultat (parametru de ieșire) de tip NUMBER;

- p\_job (parametru de intrare) de tip job din jobs\_pnu, inițializat cu NULL;

- p\_titlu (parametru de intrare) de tip job\_title din jobs\_pnu, inițializat cu NULL.

Dacă job nu este NULL atunci în rezultat se va memora numărul de angajați având codul job-ului specificat ca parametru. În caz contrar, în rezultat se va memora numărul de angajați care au titlul job-ului dat de al treilea parametru din procedura.

### **Bibliografie web:**

<https://www.w3resource.com/>

<https://www.bullraider.com/database/pl-sql/pl-sql-examples>

<https://www.oracletutorial.com/plsql-tutorial/plsql-procedure/>

<https://docs.oracle.com/database/121/LNPLS/subprograms.htm#LNPLS00825>