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 şi 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 conţin o parte declarativă opţională, o parte executabilă obligatorie şi o parte de tratare de excepţii opţională.
Exista 2 tipuri de subprograme:

1. proceduri;
2. funcţii (trebuie să conţină cel puţin o comandă RETURN);

Subprogramele pot fi:

- **locale** (în cadrul altui bloc PL/SQL sau subprogram)
- **stocate** (create cu comanda CREATE) - odată create, procedurile şi funcţiile 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}
   [declaraţii locale]
BEGIN
   partea executabilă
[EXCEPTION
   partea de tratare a excepţiilor]
END [nume_procedură];
```

Lista de parametri conţine specificaţii 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 iniţială
  - ✓ de intrare / ieşire (IN OUT)
  - ✓ de ieşire (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;

Ştergerea unei proceduri se realizează prin comenzile:

DROP PROCEDURE nume_proc;

Informaţii despre procedurile deţinute de utilizatorul curent se pot obţine interogând vizualizarea USER_OBJECTS din dicţionarul 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 şi prin intermediul comenzii SHOW ERRORS.
Descrierea specificaţiei 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 poziţie

Exemplu:

subprogram(a tip_a, b_tip_b, c tip_c, d tip_d)

a) specificare prin poziţie: **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 şi poziţie: **subprog(var_a, var_b, d=>var_d, c=>var_c);**

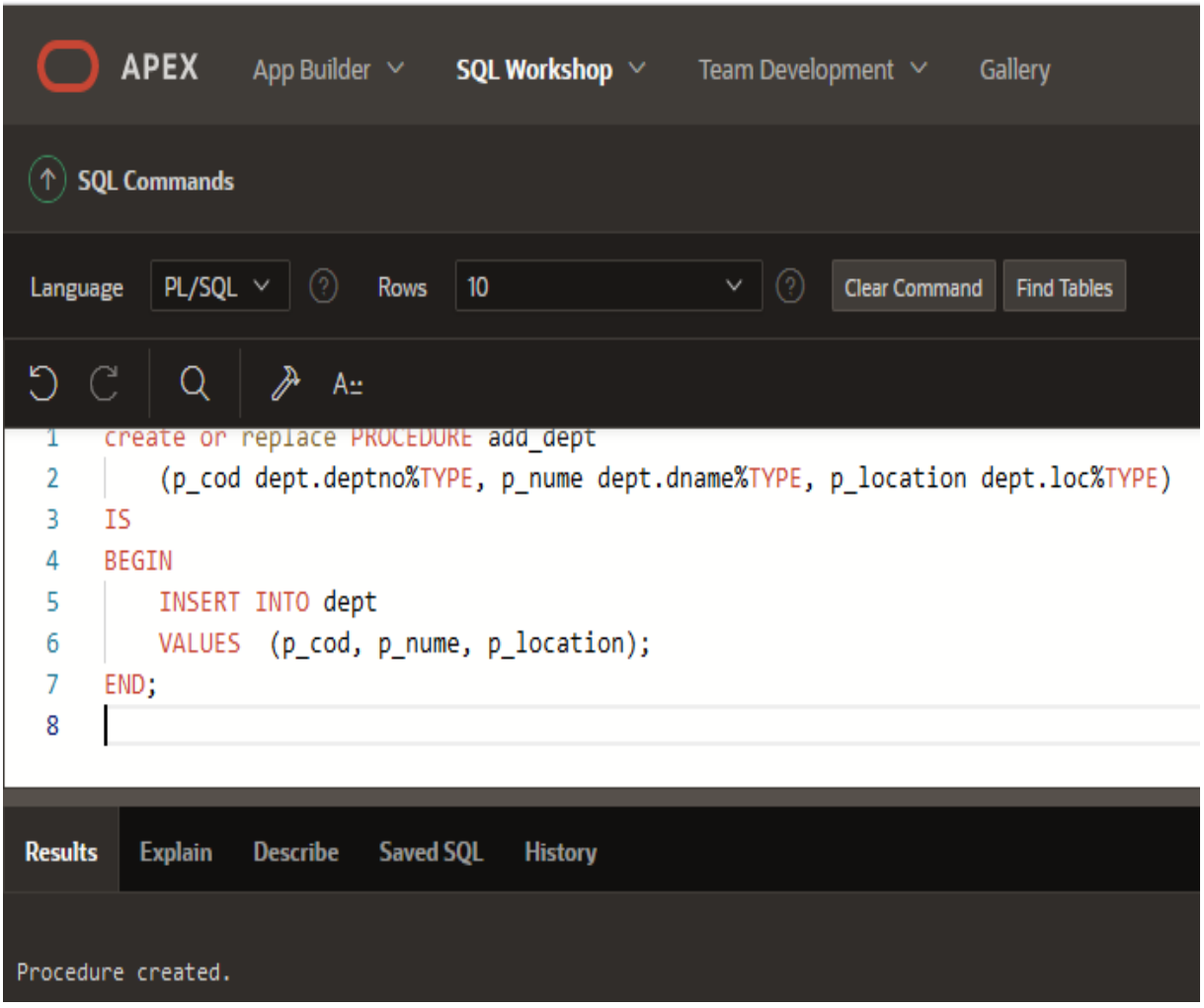
Probleme rezolvate

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. Invocaţi procedura în cadrul blocului.

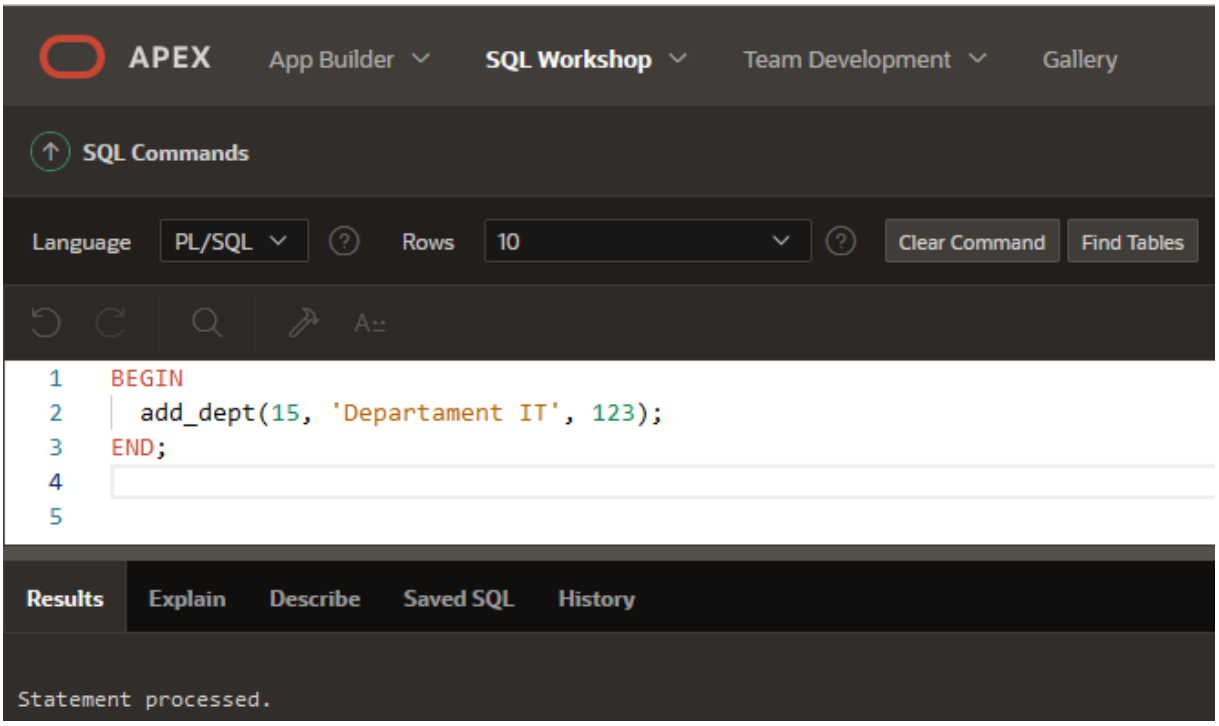
Solutie:
Pasul 1 - se creeaza procedura **add_dept** si se executa codul scris in aceasta:

```
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;
```

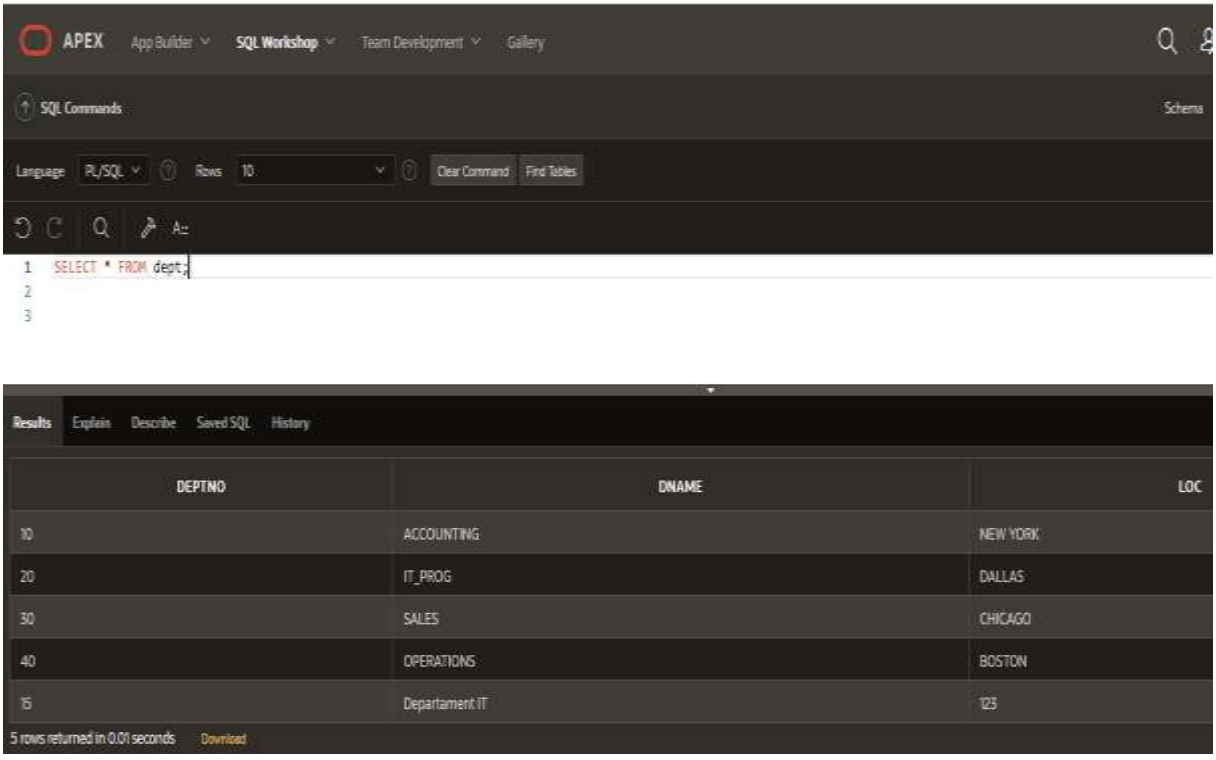


Pas 2 - se creeaza un bloc anonim care apeleaza procedura creata anterior si de observa efectul executiei:

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



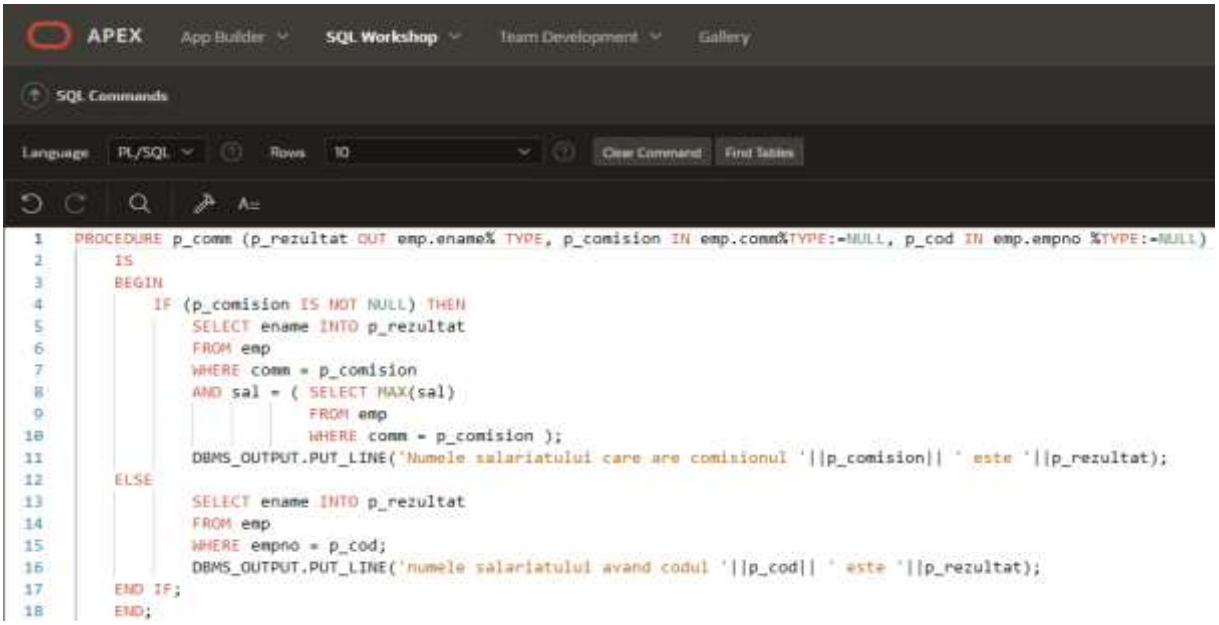
Pasul 3 - se afiseaza continutul tabelei DEPT pentru a verifica daca procedura a functionat corect:



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, iniţializat cu NULL;
- p_cod (parametru de tip IN) de tipul coloanei emp_id din EMP, iniţializat cu NULL.
Dacă p_comision nu este NULL atunci în p_rezultat se va memora numele salariatului care are salariul maxim printre salariaţii având comisionul respectiv. În caz contrar, în p_rezultat se va memora numele salariatului al cărui cod are valoarea dată la apelarea procedurii.

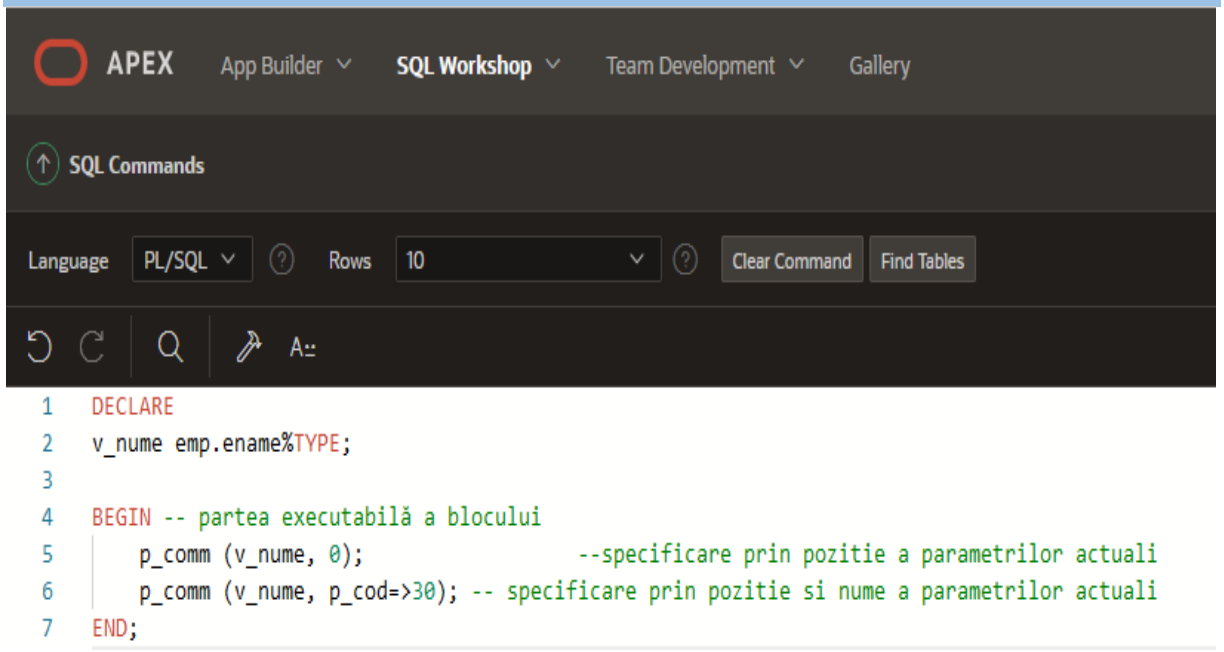
Solutie:
Pasul 1 - se creeaza procedura **add_dept** si se executa codul scris in aceasta:

```
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;
```



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

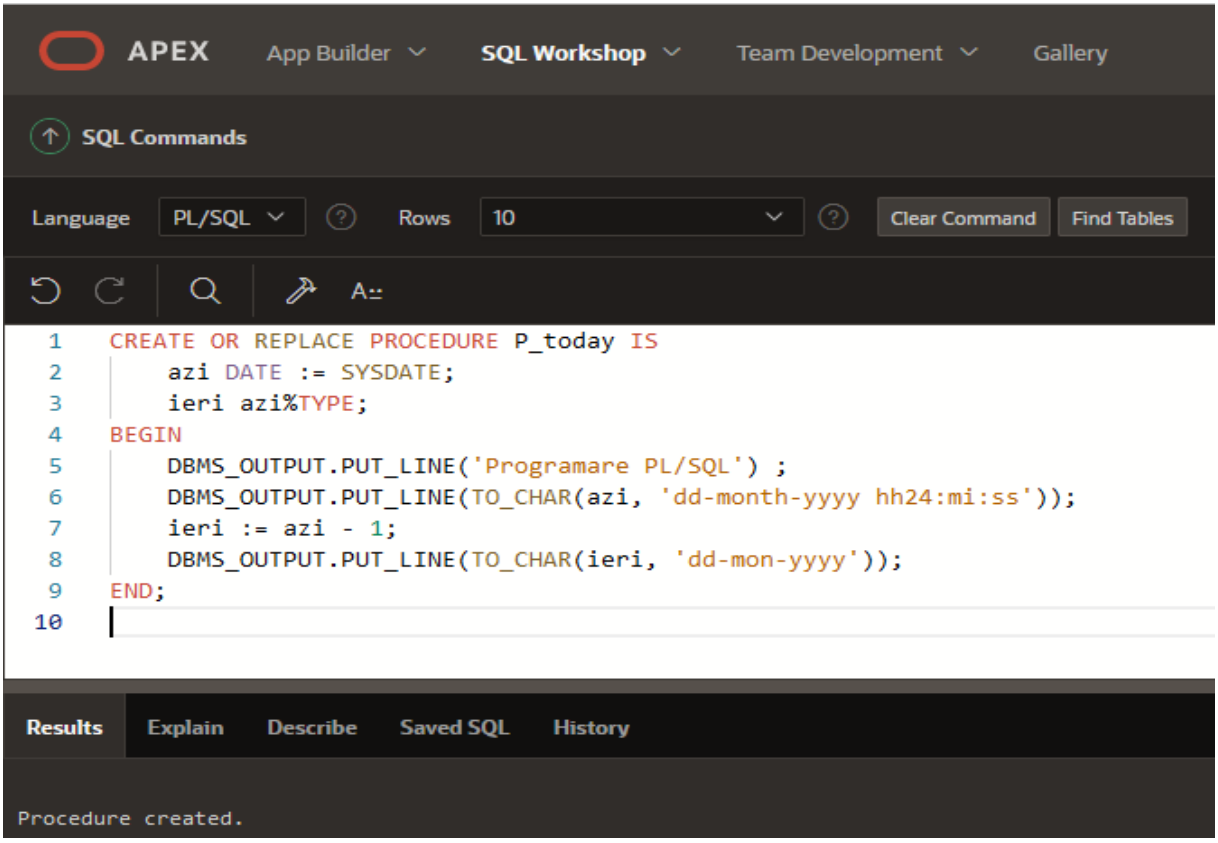
```
DECLARE
v_nume emp.ename%TYPE;

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;
```



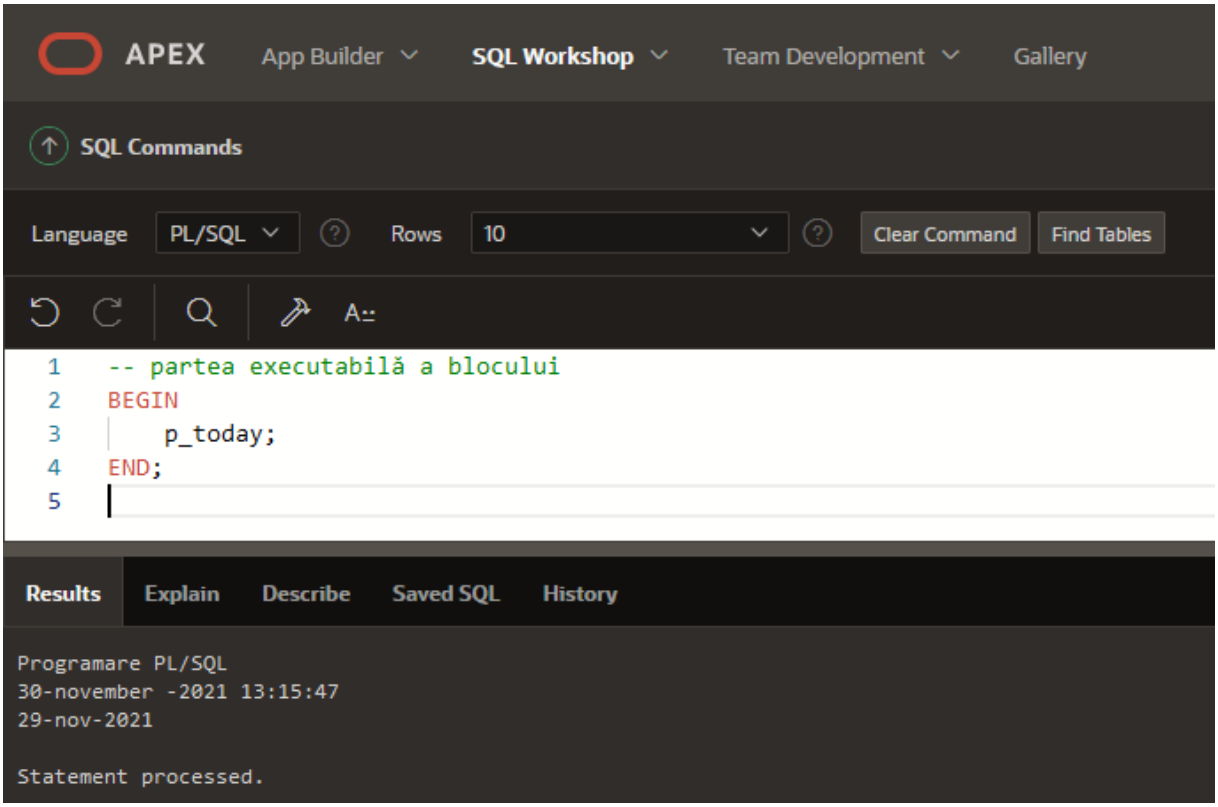
3. Să se creeze o procedură stocată fără parametri care afişează un mesaj “Programare PL/SQL”, ziua de astăzi în formatul DD-MONTH-YYYY şi ora curentă, precum şi ziua de ieri în formatul DD-MON-YYYY.

Solutie:

```
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;
```



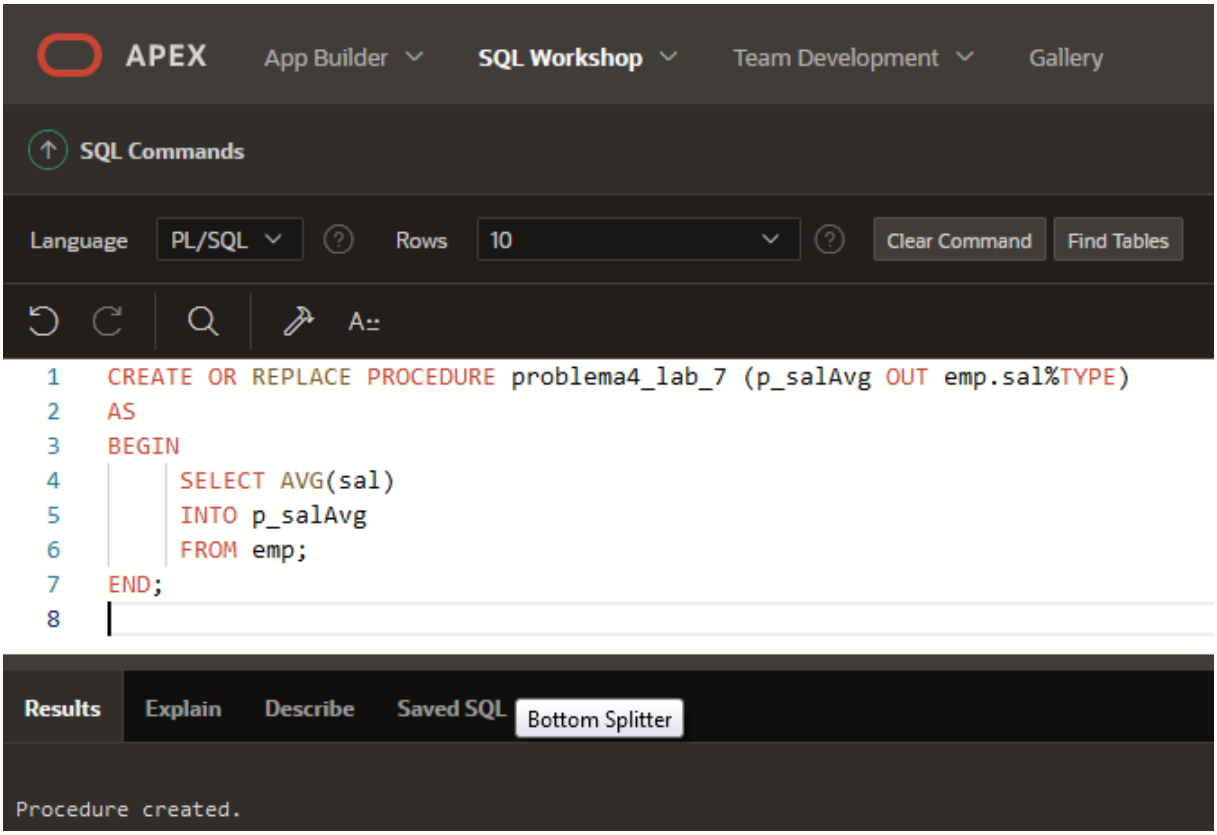

```
BEGIN -- partea executabilă a blocului
        p_today;
END;
```



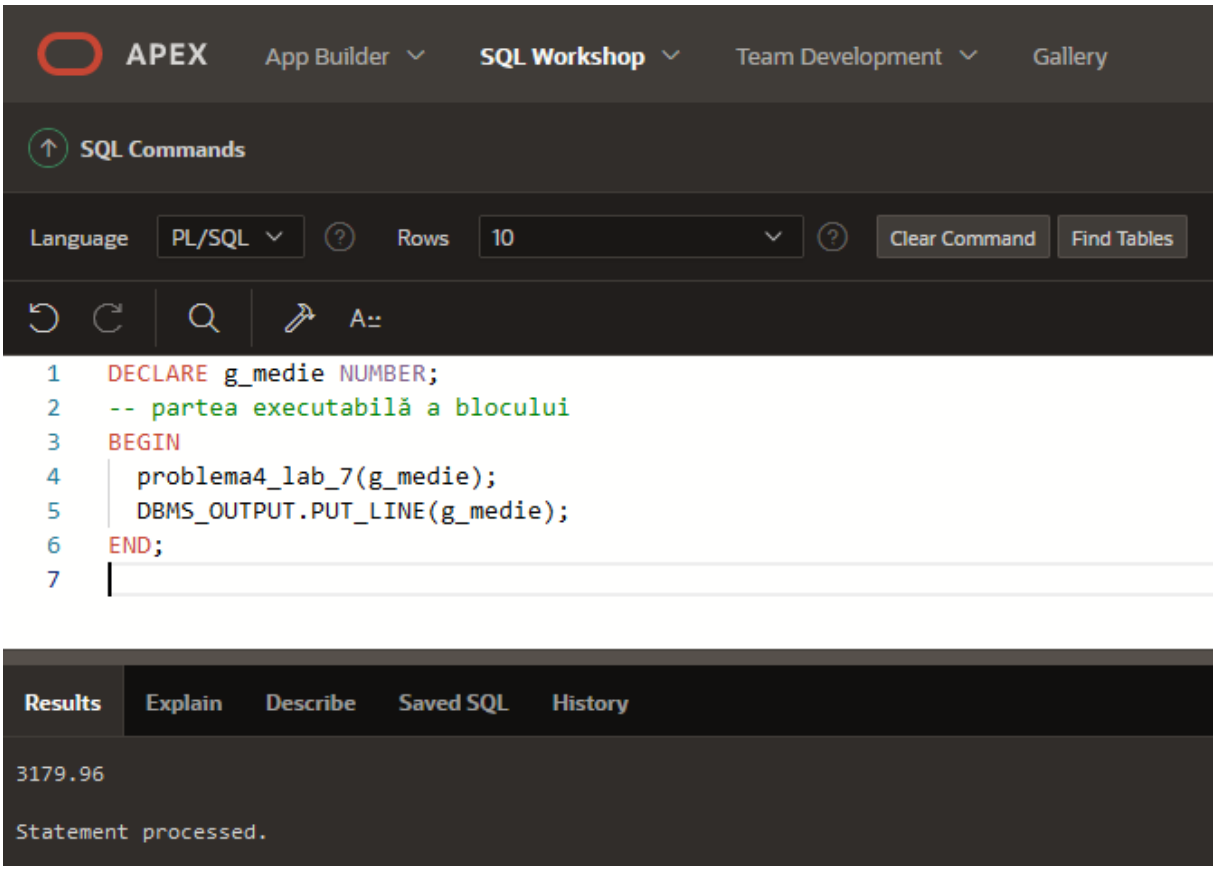
4. a) Să se creeze o procedură stocată care calculează salariul mediu al angajaţilor, returnându-l prin intermediul unui parametru de tip OUT.
b) Să se apeleze procedura regăsind valoarea medie a salariilor într-o variabilă gazdă. Afişaţi valoarea variabilei.

Solutie:

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




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



5. a) Creaţi o procedură numită GET_EMP_pnu care întoarce salariul şi denumirea job-ului pentru un angajat al cărui cod este transmis ca parametru.

b) Executaţi procedura utilizând câte o variabilă gazdă pentru cei doi parametri de tip OUT.

c) testaţi procedura atât pentru coduri existente cât şi pentru coduri inexistente din tabelul EMP_pnu. Ce se întâmplă atunci când o invocăm pentru un cod inexistent ?

6. a) Analizaţi subprogramele create anterior în vizualizarea din dicţionarul datelor USER_OBJECTS.

b) Regăsiţi codul unuia dintre subprogramele create anterior în vizualizarea USER_SOURCE.

c) Aflaţi tipul parametrilor uneia dintre procedurile create anterior utilizând comanda DESCRIBE.

Probleme propuse spre rezolvare

1. Să se creeze o procedură stocată care măreşte salariile angajaţilor care nu au comision şi 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 ieşire) de tip NUMBER;
- p_job (parametru de intrare) de tip job din jobs_pnu, iniţializat cu NULL;
- p_titlu (parametru de intrare) de tip job_title din jobs_pnu, iniţializat cu NULL.

Dacă job nu este NULL atunci în rezultat se va memora numărul de angajaţi având codul job-ului specificat ca parametru. În caz contrar, în rezultat se va memora numărul de angajaţi 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