#### 

[Data, denumire fișier (aplicați stilul Titlu 4 pt. adăugare în Cuprins)](#_u80ko911s3ko)

[Activitate: ex –> 10:50](#_pem3vp71dcye)

[Pentru adaugare enunțuri și rezolvări utilizați sablonul:](#_bji8l4ugdk0v)

[link upload soluții activitate](#_6s4kj9cpquxz)

[8.10.2024, Laborator\_PLSQL0.pdf](#_ravd79jj1tdx)

[Activitate: ex 9-20 –> 11:20](#_qm9je9lihl95)

[22.10.2024, Laborator\_PLSQL1.pdf](#_viau69ta7fom)

[Activitate: Exemple 1-4 (la exemplul 4 afisati Invat PLSQL incepand cu 22 octombrie 2024 –> 11:25](#_fv87je7ao9p2)

[5.11.2024, Laborator\_PLSQL2.pdf](#_t4kzmjuv84kr)

[Activitate: ex de mai jos –> 11:30](#_9olbs9scp77z)

[19.11.2024, Laborator\_PLSQL3.pdf](#_rw47yjwmv5qp)

[Activitate: ex4 (folosind LOOP si WHILE) -> 11:00](#_foh1a63c3a64)

[Activitate: -> 11:50 Sa se afiseze folosind expresie cursor, apoi cu 2 cursoare (unul dependent de calalalt) pentru fiecare job angajatii care lucreaza pe acel job. Tratati cazurile in care pe un job nu lucreaza nimeni si cazul in care un angajat nu are job.](#_phguoo2dcmfa)

[3.12.2024, Laborator\_PLSQL4.pdf](#_58rdl26xc9u7)

[Activitate: 4-7 -> 11:50  
Optional: Implementati 2 functii, prima returneaza o colectie cu codurile angajatilor, a doua un cursor cu codurile angajatilor si apelati.](#_e4c2c7ac432x)

[17.12.2024, Laborator\_PLSQL5.pdf](#_cxuu46gz6ami)

[Activitate: -> 11:50](#_99gvp4p4ar9j)

[14.01.2024](#_h92lgfto7u2s)

[nu s-a tinut](#_tf2ah6o6pvi1)

[END](#_2uqxu8r2j5cl)

#### Data, denumire fișier (aplicați stilul Titlu 4 pt. adăugare în Cuprins)

Shift + Enter pt. linie nouă în același paragraf

Debifati ghilimelele tipografice de la Instrumente -> Preferințe

##### Activitate: ex –> 10:50

##### Pentru adaugare enunțuri și rezolvări utilizați sablonul:

| /\*  nr exercitiu, enunț, nume rezolvitor, grupa  Enunt \*/  Rezolvare: |
| --- |

##### link upload soluții activitate

[**https://docs.google.com/forms/d/e/1FAIpQLSdp1bsN\_6Aixgq23LfiBDLmBTqf\_Sgvo3GWaS4fjzfF3h6yug/viewform?usp=sf\_link**](https://docs.google.com/forms/d/e/1FAIpQLSdp1bsN_6Aixgq23LfiBDLmBTqf_Sgvo3GWaS4fjzfF3h6yug/viewform?usp=sf_link)

#### 8.10.2024, Laborator\_PLSQL0.pdf

##### Activitate: ex 9-20 –> 11:20

#### 22.10.2024, Laborator\_PLSQL1.pdf

##### Activitate: Exemple 1-4 (la exemplul 4 afisati Invat PLSQL incepand cu 22 octombrie 2024 –> 11:25

##### 

| SET SERVEROUTPUT ON  SET VERIFY OFF  VARIABLE g\_mesaj VARCHAR2(100)  DEFINE p\_nr = 23  UNDEFINE p\_nr  DECLARE  v\_nr NUMBER NOT NULL:=&&p\_nr;  v\_mesaj VARCHAR2(50):='&p\_mesaj';  BEGIN  :g\_mesaj:='2024';  DBMS\_OUTPUT.PUT\_LINE(v\_mesaj||' '|| v\_nr|| ' octombrie '||:g\_mesaj|| ' e zi insorita!');  --NULL;  END;  /  PRINT g\_mesaj  DECLARE  v\_nume employees.last\_name%TYPE;  v\_cod employees.department\_id%TYPE:=&p\_cod;  BEGIN  SELECT last\_name  INTO v\_nume  FROM employees  WHERE department\_id = v\_cod;  DBMS\_OUTPUT.PUT\_LINE('ang cu codul '||v\_cod||' are numele '||v\_nume);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('NU SUNT ang '||SQLCODE||' '||SQLERRM);  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('sunt mai multi ang');  END;  / |
| --- |

#### 5.11.2024, Laborator\_PLSQL2.pdf

| SET SERVEROUTPUT ON  DECLARE  TYPE tab\_ind IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;  t tab\_ind;  TYPE tab\_imb IS TABLE OF NUMBER;  t2 tab\_imb:=tab\_imb();  TYPE vec IS VARRAY(10) OF NUMBER;  t3 vec:=vec();  BEGIN  t(1):=1;  DBMS\_OUTPUT.PUT\_LINE(t.COUNT||' '||t.FIRST||' '||t.LAST);  t2.EXTEND;  t2(1):=2;  DBMS\_OUTPUT.PUT\_LINE(t2.COUNT||' '||t2.FIRST||' '||t2.LAST||' '|| t2(1));  t3.EXTEND;  t3(1):=3;  DBMS\_OUTPUT.PUT\_LINE(t3.COUNT||' '||t3.FIRST||' '||t3.LAST||' '|| t3(1));  END;  /  DECLARE  TYPE tab\_ind IS TABLE OF employees.employee\_id%TYPE INDEX BY PLS\_INTEGER;  t tab\_ind;  BEGIN  SELECT employee\_id  BULK COLLECT INTO t  FROM employees;  FOR i IN 1..t.COUNT LOOP  DBMS\_OUTPUT.PUT\_LINE(t(i));  END LOOP;  END;  /  CREATE OR REPLACE TYPE tab\_imb IS TABLE OF NUMBER;  /  SELECT \*  FROM USER\_TYPES;  CREATE TABLE test (a NUMBER PRIMARY KEY, b tab\_imb)  NESTED TABLE b STORE AS test\_imb\_spr;  INSERT INTO test VALUES (1,tab\_imb(2));  INSERT INTO test VALUES (2,tab\_imb(3,5));  INSERT INTO test VALUES (3,tab\_imb(4,7,9));  COMMIT;  SELECT \*  FROM test;  SELECT x.a,y.\*  FROM test x, TABLE(x.b) y;  UPDATE TABLE(SELECT b FROM test WHERE a = 3) y  SET VALUE(y) = 10  WHERE COLUMN\_VALUE = 9;  CREATE OR REPLACE TYPE vec IS VARRAY(10) OF NUMBER;  /  SELECT \*  FROM USER\_TYPES;  CREATE TABLE test2 (a NUMBER PRIMARY KEY, b vec);  INSERT INTO test2 VALUES (1,vec(2));  INSERT INTO test2 VALUES (2,vec(3,5));  INSERT INTO test2 VALUES (3,vec(4,7,9));  COMMIT;  SELECT \*  FROM test2;  SELECT x.a,y.\*  FROM test2 x, TABLE(x.b) y;  --nu se poate:  UPDATE TABLE(SELECT b FROM test2 WHERE a = 3) y  SET VALUE(y) = 10  WHERE COLUMN\_VALUE = 9; |
| --- |

##### Activitate: ex de mai jos –> 11:30

1. Sa se salveze intr-o colectie codurile celor 7 angajati cel mai bine platiti, apoi sa se actualize salariul acestora cu 20%. Sa se afiseze valoare veche, resp. noua a salariilor. Testati pt toate tipurile de colectii si folositi FORALL.
2. Trimiteti textul afisat si/sau efectul comenzilor de la exemplul 15.

#### 19.11.2024, Laborator\_PLSQL3.pdf

##### 

| SET SERVEROUTPUT ON  --Soluția 1: cursor clasic  DECLARE  CURSOR c\_emp IS  SELECT last\_name, salary\*12 sal\_an  FROM emp\_spr  WHERE department\_id = 50;  v\_emp c\_emp%ROWTYPE;  BEGIN  OPEN c\_emp;  /\*IF c\_emp%FOUND THEN DBMS\_OUTPUT.PUT\_LINE('cursor deschis');  else  DBMS\_OUTPUT.PUT\_LINE('cursor inchis');  end if;\*/    FETCH c\_emp INTO v\_emp;  WHILE (c\_emp%FOUND) LOOP  DBMS\_OUTPUT.PUT\_LINE (' Nume:' || v\_emp.last\_name || ' are salariul anual : ' || v\_emp.sal\_an);  FETCH c\_emp INTO v\_emp;  END LOOP;  CLOSE c\_emp;  END;  /  --Soluția 2: cursor clasic  DECLARE  CURSOR c\_emp IS  SELECT last\_name, salary\*12 sal\_an  FROM emp\_spr  WHERE department\_id = 50;  v\_emp c\_emp%ROWTYPE;  BEGIN  OPEN c\_emp;  LOOP  FETCH c\_emp INTO v\_emp;  --EXIT WHEN c\_emp%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE (' Nume:' || v\_emp.last\_name || ' are salariul anual : ' || v\_emp.sal\_an);  END LOOP;  CLOSE c\_emp;  END;  /  --Soluția 3: nu mai este nevoie explicit de OPEN, FETCH, CLOSE !!! ciclu cursor  DECLARE  CURSOR c\_emp IS  SELECT last\_name, salary\*12 sal\_an  FROM emp\_spr  WHERE department\_id = 50;  BEGIN  FOR v\_emp IN c\_emp LOOP  DBMS\_OUTPUT.PUT\_LINE (' Nume:' || v\_emp.last\_name || ' are  salariul anual : ' || v\_emp.sal\_an);  END LOOP;  END;  /  --Soluția 4: ciclu cursor cu subcerere  BEGIN  FOR v\_rec IN (SELECT last\_name, salary\*12 sal\_an  FROM employees  WHERE department\_id = 50) LOOP    DBMS\_OUTPUT.PUT\_LINE (' Nume:' || v\_rec.last\_name || ' are salariul anual : ' || v\_rec.sal\_an);  END LOOP;  END;  / |
| --- |

##### Activitate: ex4 (folosind LOOP si WHILE) -> 11:00

##### 

| ACCEPT p\_optiune PROMPT 'Introduceti optiunea (1,2 sau 3) '  UNDEFINE p\_optiune  DECLARE  TYPE emp\_tip IS REF CURSOR RETURN emp\_spr%ROWTYPE;  v\_emp emp\_tip;  v\_optiune NUMBER := &p\_optiune;  v\_ang emp\_spr%ROWTYPE;  BEGIN  IF v\_optiune = 1 THEN  OPEN v\_emp FOR SELECT \* FROM emp\_spr;  ELSIF v\_optiune = 2 THEN  OPEN v\_emp FOR SELECT \*  FROM emp\_spr  WHERE salary BETWEEN 10000 AND 20000;  ELSIF v\_optiune = 3 THEN  OPEN v\_emp FOR SELECT \*  FROM emp\_spr  WHERE TO\_CHAR(hire\_date, 'YYYY') = 1990;  ELSE  DBMS\_OUTPUT.PUT\_LINE('Optiune incorecta');  END IF;  /\*IF v\_emp%ISOPEN THEN  LOOP  FETCH v\_emp into v\_ang;  EXIT WHEN v\_emp%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE(v\_ang.last\_name);  END LOOP;  DBMS\_OUTPUT.PUT\_LINE('Au fost procesate '||v\_emp%ROWCOUNT  || ' linii');  CLOSE v\_emp;  END IF;\*/  FOR i IN v\_emp LOOP  DBMS\_OUTPUT.PUT\_LINE(i.last\_name);  END LOOP;  END;  /  DECLARE  CURSOR c\_regiune IS  SELECT region\_name,  CURSOR (SELECT country\_name    FROM countries c  WHERE c.region\_id = r.region\_id)  FROM regions r;  v\_regiune regions.region\_name%TYPE;  v\_tara SYS\_REFCURSOR;  TYPE tara\_nume IS TABLE OF countries.country\_name%TYPE  INDEX BY BINARY\_INTEGER;  v\_nume\_tara tara\_nume;  BEGIN  OPEN c\_regiune;  LOOP  FETCH c\_regiune INTO v\_regiune, v\_tara;  EXIT WHEN c\_regiune%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE (CHR(10)||v\_regiune);  FETCH v\_tara BULK COLLECT INTO v\_nume\_tara;  FOR ind IN v\_nume\_tara.FIRST..v\_nume\_tara.LAST LOOP  DBMS\_OUTPUT.PUT\_LINE (CHR(9)||v\_nume\_tara (ind));  END LOOP;  END LOOP;  CLOSE c\_regiune;  END;  /  SELECT job\_id,job\_title FROM jobs WHERE job\_id NOT IN (SELECT NVL(job\_id,' ') FROM employees);  SELECT employee\_id,first\_name, last\_name FROM employees WHERE job\_id IS NULL;  INSERT INTO jobs VALUES  ( 'No'  , 'No\_stress'  , 100000  , 500000  );    INSERT INTO employees VALUES  ( 99  , 'Almighty'  , 'ME'  , 'AM'  , '515.123.4567'  , TO\_DATE('15-NOI-2023', 'dd-MON-yyyy')  , NULL  , 200000  , NULL  , NULL  , 90  );  COMMIT; |
| --- |

##### 

##### Activitate: -> 11:50 Sa se afiseze folosind expresie cursor, apoi cu 2 cursoare (unul dependent de calalalt) pentru fiecare job angajatii care lucreaza pe acel job. Tratati cazurile in care pe un job nu lucreaza nimeni si cazul in care un angajat nu are job.

#### 3.12.2024, Laborator\_PLSQL4.pdf

##### 

| CREATE OR REPLACE FUNCTION p14l4\_pnu (p\_dept employees.department\_id%TYPE)  RETURN NUMBER IS  rezultat NUMBER;  BEGIN  SELECT COUNT(\*)  INTO rezultat  FROM employees  WHERE department\_id = p\_dept  AND TO\_CHAR(hire\_date,'yyyy') > 1995;  RETURN rezultat;  END p14l4\_pnu;  /  --a)  EXEC DBMS\_OUTPUT.PUT\_LINE(p14l4\_pnu (80))  CREATE OR REPLACE FUNCTION medie\_pnu  RETURN NUMBER IS  medie NUMBER;  BEGIN  SELECT AVG(salary)  INTO medie  FROM employees;  --INSERT INTO test VALUES (1,tab\_imb(1,2));  RETURN medie;  END;  /  SELECT last\_name, job\_id, salary  FROM employees  WHERE salary >= medie\_pnu; |
| --- |

##### Activitate: 4-7 -> 11:50 Optional: Implementati 2 functii, prima returneaza o colectie cu codurile angajatilor, a doua un cursor cu codurile angajatilor si apelati.

##### 

| SET SERVEROUTPUT ON  CREATE OR REPLACE PACKAGE pachet  IS  v NUMBER;  FUNCTION functie RETURN NUMBER;  CURSOR c RETURN employees%ROWTYPE;  END;  /  CREATE OR REPLACE PACKAGE BODY pachet  IS  CURSOR c RETURN employees%ROWTYPE IS SELECT \* FROM employees;  FUNCTION functie RETURN NUMBER  IS BEGIN  SELECT COUNT(\*)  INTO v  FROM employees;  RETURN v;  END functie;  END;  /  BEGIN  DBMS\_OUTPUT.PUT\_LINE(pachet.functie);  END;  / |
| --- |

#### 17.12.2024, Laborator\_PLSQL5.pdf

##### 

| SET SERVEROUTPUT ON  CREATE OR REPLACE TRIGGER b\_i\_emp\_spr0  BEFORE INSERT ON emp\_spr  BEGIN  IF (TO\_CHAR(SYSDATE, 'dy') IN ('sat', 'sun')) OR  (TO\_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '12:00' AND '18:00') THEN  RAISE\_APPLICATION\_ERROR (-20500, 'Nu se pot introduce inregistrari  decat in timpul orelor de lucru');  END IF;  END;  /  ALTER TRIGGER b\_i\_emp\_spr DISABLE;  INSERT INTO emp\_spr (last\_name,email,hire\_date,job\_id) VALUES ('DDF','SS',SYSDATE,'it\_prog');  ROLLBACK;  CREATE OR REPLACE TRIGGER b\_i\_emp\_spr  BEFORE INSERT OR UPDATE OR DELETE ON emp\_spr  BEGIN  IF (TO\_CHAR(SYSDATE, 'dy') IN ('sat', 'sun')) OR  (TO\_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '12:00' AND '18:00') THEN  IF DELETING THEN  RAISE\_APPLICATION\_ERROR (-20501, 'Nu se pot sterge  inregistrari decat in timpul orelor de luru');  ELSIF INSERTING THEN  RAISE\_APPLICATION\_ERROR (-20500, 'Nu se pot adauga  inregistrari decat in timpul orelor de lucru');  ELSIF UPDATING ('SALARY') THEN  RAISE\_APPLICATION\_ERROR (-20502, 'Nu se poate actualiza campul  SALARY decat in timpul orelor de lucru');  ELSE  RAISE\_APPLICATION\_ERROR (-20503, 'Nu se pot actualiza  inregistrari decat in timpul orelor de lucru');  END IF;  END IF;  END;  /  CREATE OR REPLACE TRIGGER check\_sal\_spr  BEFORE INSERT OR UPDATE OF salary, job\_id ON emp\_spr  FOR EACH ROW  WHEN (NEW.job\_id <> 'AD\_PRES')  DECLARE  v\_min employees.salary %TYPE;  v\_max employees.salary %TYPE;  BEGIN  SELECT MIN(salary), MAX(salary)  INTO v\_min, v\_max  FROM emp\_spr -- FROM copie\_emp\_spr  WHERE job\_id = :NEW.job\_id;  IF :NEW.salary < v\_min OR :NEW.salary > v\_max THEN  RAISE\_APPLICATION\_ERROR (-20505, 'In afara domeniului');  END IF;  END;  /  --Testati trigger-ul anterior:  UPDATE emp\_spr  SET salary = 3500  WHERE last\_name= 'Stiles';  DROP TRIGGER check\_sal\_spr;  CREATE TABLE new\_dept\_spr AS  SELECT d.department\_id, d.department\_name, d.location\_id,  SUM(e.salary) total\_dept\_sal  FROM employees e, departments d  WHERE e.department\_id = d.department\_id  GROUP BY d.department\_id, d.department\_name, d.location\_id;  ALTER TABLE new\_dept\_spr ADD CONSTRAINT new\_dept\_spr\_pk PRIMARY  KEY(department\_id);  CREATE TABLE new\_emp\_spr AS  SELECT employee\_id, last\_name, salary,  department\_id, email, job\_id, hire\_date  FROM employees;  ALTER TABLE new\_emp\_spr ADD CONSTRAINT new\_emp\_spr\_pk PRIMARY  KEY(employee\_id);  ALTER TABLE new\_emp\_spr ADD CONSTRAINT new\_emp\_dept\_spr\_fk  FOREIGN KEY(department\_id) REFERENCES new\_dept\_spr (department\_id);  CREATE OR REPLACE VIEW view\_emp\_spr AS  SELECT e.employee\_id, e.last\_name, e.salary, e.department\_id,  e.email, e.job\_id, d.department\_name, d.location\_id, d.total\_dept\_sal  FROM new\_emp\_spr e, new\_dept\_spr d  WHERE e.department\_id = d.department\_id;  SELECT \* FROM new\_emp\_spr;  SELECT \* FROM new\_dept\_spr;  SELECT \* FROM view\_emp\_spr;  SELECT \*  FROM user\_updatable\_columns  WHERE table\_name = UPPER('view\_emp\_spr');  CREATE OR REPLACE TRIGGER new\_emp\_dept\_spr  INSTEAD OF INSERT OR UPDATE OR DELETE ON view\_emp\_spr  FOR EACH ROW  BEGIN  IF INSERTING THEN  INSERT INTO new\_emp\_spr  VALUES(:NEW.employee\_id, :NEW.last\_name, :NEW.salary,  :NEW.department\_id, :NEW.email, :NEW.job\_id, SYSDATE);  UPDATE new\_dept\_spr  SET total\_dept\_sal = total\_dept\_sal + :NEW.salary  WHERE department\_id = :NEW.department\_id;  ELSIF DELETING THEN  DELETE FROM new\_emp\_spr  WHERE employee\_id = :OLD.employee\_id;  UPDATE new\_dept\_spr  SET total\_dept\_sal = total\_dept\_sal - :OLD.salary  WHERE department\_id = :OLD.department\_id;  ELSIF UPDATING ('salary') THEN  UPDATE new\_emp\_spr  SET salary = :NEW.salary  WHERE employee\_id = :NEW.employee\_id;  UPDATE new\_dept\_spr  SET total\_dept\_sal = total\_dept\_sal + (:NEW.salary - :OLD.salary)  WHERE department\_id = :OLD.department\_id;  ELSIF UPDATING ('department\_id') THEN  UPDATE new\_emp\_spr  SET department\_id = :NEW.department\_id  WHERE employee\_id = :OLD.employee\_id;    UPDATE new\_dept\_spr  SET total\_dept\_sal = total\_dept\_sal - :OLD.salary  WHERE department\_id = :OLD.department\_id;  UPDATE new\_dept\_spr  SET total\_dept\_sal = total\_dept\_sal + :NEW.salary  WHERE department\_id = :NEW.department\_id;  END IF; END;  /  -- adaugarea unui nou angajat  SELECT \* FROM new\_dept\_spr WHERE department\_id = 10;  INSERT INTO view\_emp\_spr  VALUES (400, 'N1', 3000, 10, 'n1@g.com', 'SA\_REP', 'Nume dept', 1000, 0);  SELECT \* FROM new\_emp\_spr WHERE employee\_id = 400;  SELECT \* FROM new\_dept\_spr WHERE department\_id = 10;  -- modificarea salariului unui angajat  UPDATE view\_emp\_spr  SET salary = salary + 1000  WHERE employee\_id = 400;  SELECT \* FROM new\_emp\_spr WHERE employee\_id = 400;  SELECT \* FROM new\_dept\_spr WHERE department\_id = 10;  -- modificarea departamentului unui angajat  SELECT \* FROM view\_emp\_spr WHERE department\_id=90;  UPDATE view\_emp\_spr  SET department\_id = 90  WHERE employee\_id = 400;  SELECT \* FROM new\_emp\_spr WHERE employee\_id = 400;  SELECT \* FROM new\_dept\_spr WHERE new\_dept\_spr.department\_id IN (10,90);  -- eliminarea unui angajat  DELETE FROM view\_emp\_spr WHERE employee\_id = 400;  SELECT \* FROM new\_emp\_spr WHERE employee\_id = 400;  SELECT \* FROM new\_dept\_spr WHERE department\_id = 90;  DROP TRIGGER new\_emp\_dept\_spr;  DROP VIEW view\_emp\_spr;  DROP TABLE new\_emp\_spr;  DROP TABLE new\_dept\_spr;  SELECT \*  FROM USER\_TRIGGERS;  DELETE FROM emp\_spr;  ALTER TABLE emp\_spr DISABLE ALL TRIGGERS;    INSERT INTO emp\_spr  SELECT \*  FROM employees;  /\*  cu un singur trigger => eroare, deoarece declanșatorul consultă chiar tabelul  la care este asociat; emp\_spr e mutating table  \*/  CREATE OR REPLACE TRIGGER mutating\_spr  BEFORE INSERT OR UPDATE OF department\_id ON emp\_spr  FOR EACH ROW  DECLARE  v\_max CONSTANT NUMBER := 45;  v\_nr NUMBER;  BEGIN  SELECT COUNT(\*) INTO v\_nr  FROM emp\_spr  WHERE department\_id = :NEW.department\_id;  IF v\_nr + 1 > v\_max THEN  RAISE\_APPLICATION\_ERROR(-20000,'Prea multi angajati in departamentul  avand codul ' ||:NEW.department\_id);  END IF;  END mutating\_spr;  /  --merge  INSERT INTO emp\_spr VALUES  (employees\_seq.NEXTVAL,'Prenume','Nume','a','t',SYSDATE,'IT\_PROG',  10000,0.1,100,50);  --mutating table  INSERT INTO emp\_spr  SELECT  employees\_seq.NEXTVAL,'Prenume','Nume','a','t',SYSDATE,'IT\_PROG',10000,0.1,100  ,50  FROM dual;  --fara PK in emp\_spr  --insert multiplu => mutating table  INSERT INTO emp\_spr  SELECT \* FROM employees WHERE department\_id = 50;  --mutating table  UPDATE emp\_spr  SET department\_id = 50  WHERE department\_id = 20;  SELECT department\_id, count(\*)  FROM emp\_spr  GROUP BY department\_id; Activitate: -> 11:50 /\*Sa se implementeze 2 triggeri, unul comanda și altul linie pe tabelul dept\_pnu care sa verifice:   1. comanda: dacă ziua saptamanii nu este l-v sa nu fie permisă modificarea datelor 2. linie: dacă noul manager nu este în baza de date sa nu se permita inserarea in tabel 3. opțional sa se verifice daca nr de joburi ale unui angajat in trecut este 2 și în caz afirmativ sa nu se permita avansarea pe o alta pozitie\*/ |
| --- |

#### 14.01.2024

##### nu s-a tinut

##### END