

Academia de Studii Economice
Facultatea de Cibernetica, Statistica si Informatica Economica

Proiect - SGBD
Gestiunea unui magazin

Student: Boabă Ionuț-Constantin

Grupa: 1049 C

1. Descrierea temei proiectului

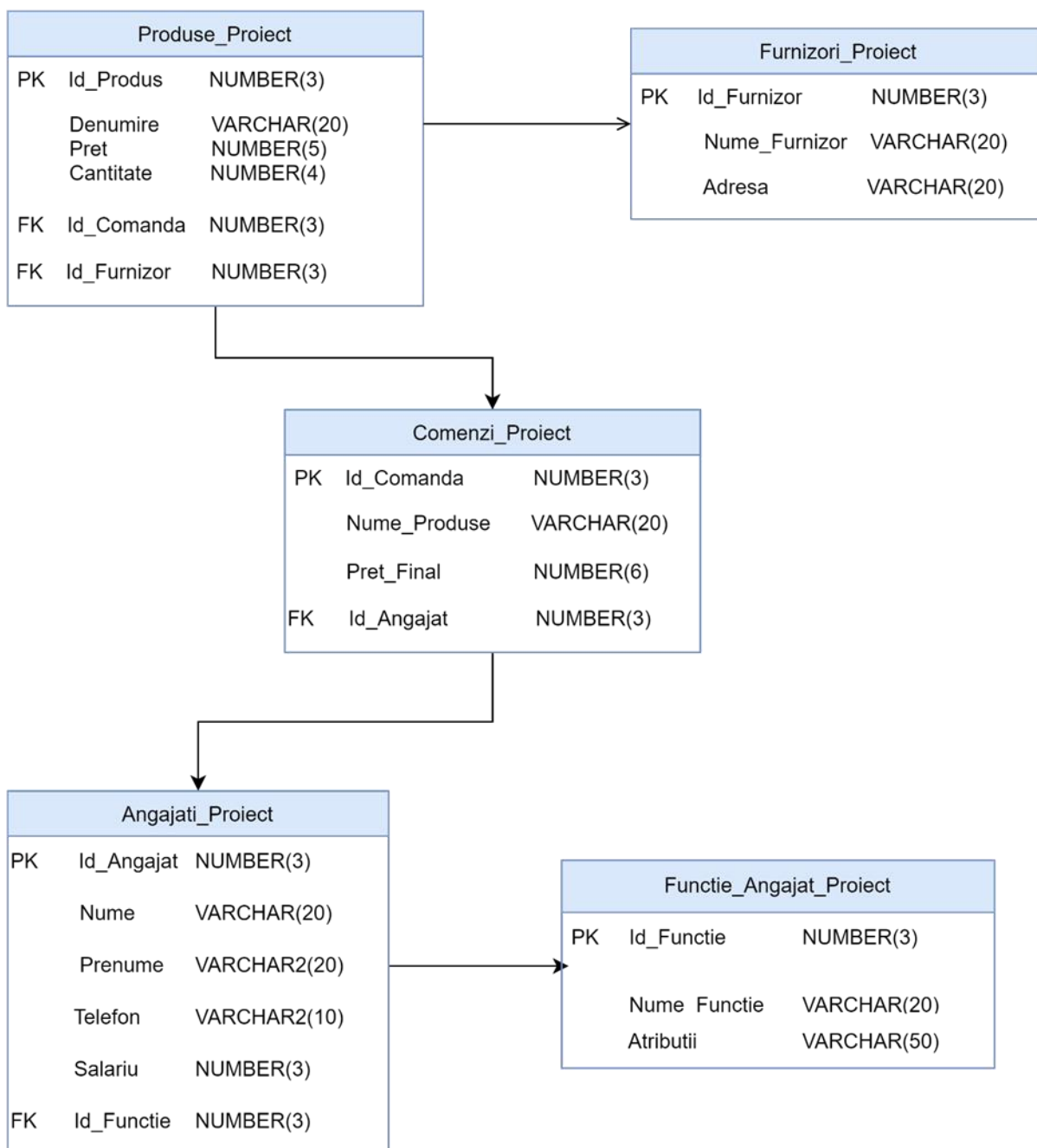
Tema proiectului:

- gestiunea activității unui magazin

Descrierea proiectului:

- baza de date permite unui manager de supermarket să gestioneze detalii despre angajați, furnizori, produse pentru vânzare și comenzile plasate
- angajații sunt caracterizați de poziția pe care o dețin și de atribuțiile pe care le au
- comenzile sunt caracterizate de produsele pe care le conțin, fiecare produs având un furnizor specific.

2. Schema conceptuală a bazei de date



3. Tema 1 – proiect - structuri

1. Actualizarea salariului angajaților în funcție de numărul de comenzi preluate

```
DECLARE
```

```
v_id angajati_proiect.id_angajat%type:= &id;
```

```
v_nume angajati_proiect.numa%type;
```

```
v_salariul angajati_proiect.salariu%type;
```

```
v_nrcom NUMBER;
```

```
BEGIN
```

```
SELECT nume, salariu INTO v_nume, v_salariul FROM Angajati_Proiect WHERE id_angajat = v_id;
```

```
DBMS_OUTPUT.PUT_LINE('Angajatul '||v_nume||' are salariul initial '||v_salariul);
```

```
SELECT count(id_comanda) INTO v_nrcom FROM Comenzi_Proiect WHERE id_angajat=v_id;
```

```
DBMS_OUTPUT.PUT_LINE(v_nrcom);
```

```
IF v_nrcom BETWEEN 0 AND 2 THEN
```

```
v_salariul:=v_salariul * 1.1;
```

```
ELSIF v_nrcom > 2 THEN
```

```
v_salariul:=v_salariul * 1.2;
```

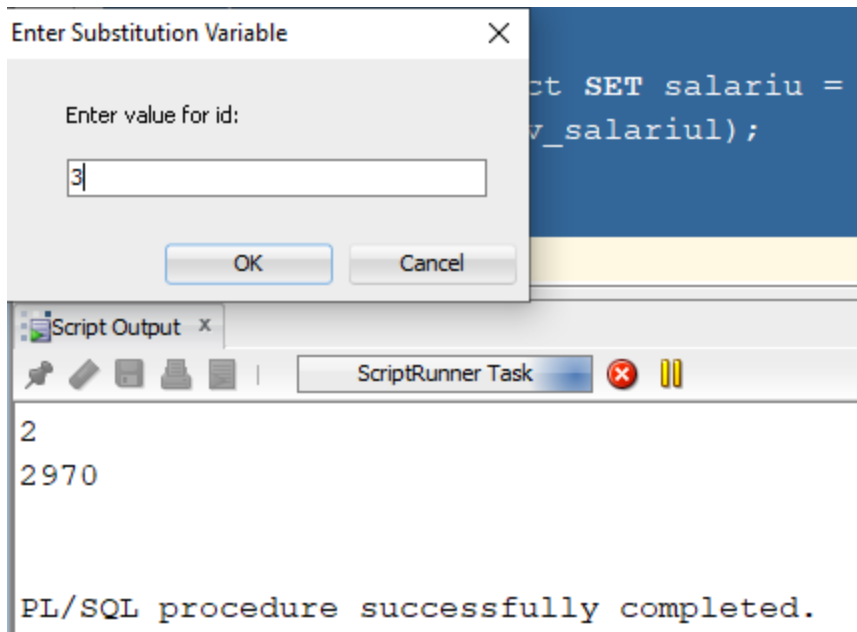
```
END IF;
```

```
UPDATE Angajati_Proiect SET salariu = v_salariul WHERE id_angajat = v_id;
```

```
DBMS_OUTPUT.PUT_LINE(v_salariul);
```

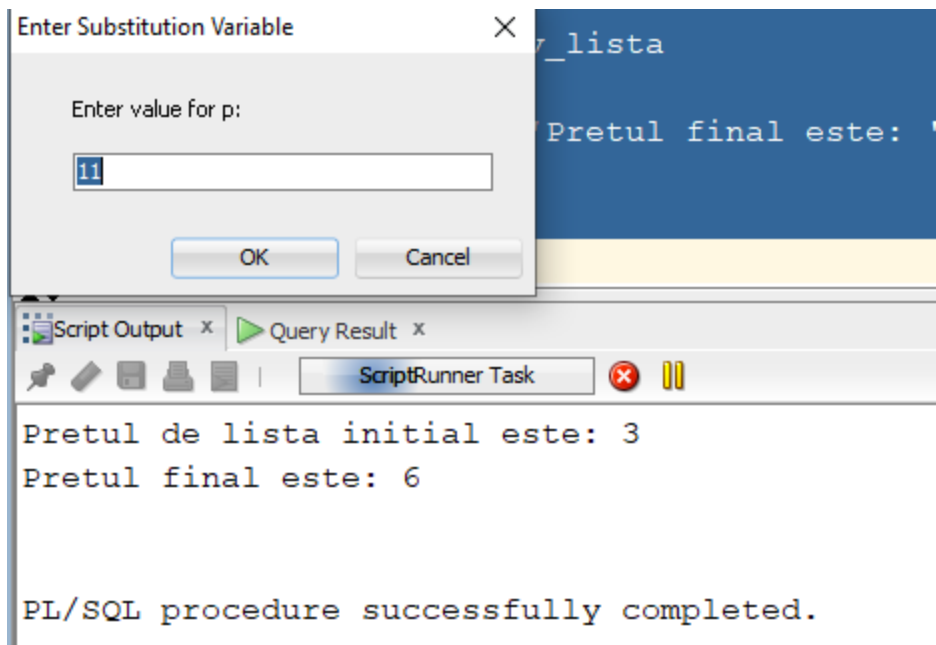
```
END;
```

```
/
```



2. Actualizarea pretului unui produs in functie de pretul sau initial folosind structura case – when

```
DECLARE
v_lista produse_proiect.pret%type;
BEGIN
SELECT pret into v_lista from produse_proiect where id_produc=&p;
dbms_output.put_line ('Pretul de lista initial este: '||v_lista);
v_lista:= CASE WHEN v_lista < 15 THEN 2 * v_lista
               WHEN v_lista between 15 and 20 THEN 1.5 * v_lista
               ELSE 1.25* v_lista
               END;
dbms_output.put_line('Pretul final este: '||v_lista);
end;
/
```



3. Afisarea în ordine a angajaților cu id-urile în intervalul 1-10 atât timp cât salariul acestora este mai mic decât media folosind structura loop – end loop.

```
DECLARE
```

```
v_sal angajati_proiect.salariu%type;
```

```
v_salMediu v_sal%type;
```

```
i number(4):=1;
```

```
BEGIN
```

```
SELECT avg(salariu) into v_salmediu from angajati_proiect;
```

```
dbms_output.put_line('Salariul mediu este: '||v_salmediu);
```

```
loop
```

```
select salariu into v_sal from angajati_proiect where id_angajat=i;
```

```
dbms_output.put_line('Salariatul cu id-ul '||i||' are salariul: '||v_sal);
```

```
i:=i+1;
```

```
exit when v_sal > v_salmediu or i > 10;
```

```
end loop;
```

end;

/

```
Salariul mediu este: 2797
Salariatul cu id-ul 1 are salariul: 3500

PL/SQL procedure successfully completed.
```

4. Afisarea mediilor tuturor tipurilor de salarii

DECLARE

v_id_functie angajati_proiect.id_functie%type;

v_nume_functie functie_proiect.nume_functie%type;

v_media_salarii NUMBER;

CURSOR c_angajati IS

SELECT DISTINCT id_functie FROM angajati_proiect;

BEGIN

OPEN c_angajati;

LOOP FETCH c_angajati INTO v_id_functie;

EXIT WHEN c_angajati%NOTFOUND;

SELECT nume_functie INTO v_nume_functie FROM functie_proiect WHERE id_functie =
v_id_functie;

SELECT DISTINCT avg(salariu) INTO v_media_salarii FROM angajati_proiect WHERE id_functie =
v_id_functie;

```
DBMS_OUTPUT.PUT_LINE('Media salariilor pentru functia ' || v_nume_functie || ' este ' ||  
v_media_salarii);  
END LOOP;
```

```
CLOSE c_angajati;
```

```
end;
```

```
/
```

```
Media salariilor pentru functia Casier este 2500  
Media salariilor pentru functia Distribuitor este 2889  
Media salariilor pentru functia Paznic este 2400  
Media salariilor pentru functia Femeie de servici este 2100  
Media salariilor pentru functia Supervisor este 3650
```

5. Sa se afiseze printr-un ciclu FOR numele și salariile angajaților care au salariul mai mare de 3000

```
declare
```

```
cursor angajati_cursor is select id_angajat, nume, salariu from angajati_proiect where salariu > 3000;
```

```
begin
```

```
dbms_output.put_line('Lista cu salariile angajatilor cu salariul > 3000');
```

```
for angajati_rec in angajati_cursor loop
```

```
dbms_output.put_line('Salariatul '||angajati_rec.nume||' are salariul: '||angajati_rec.salariu);
```

```
end loop;
```

```
end;
```

```
/
```



```
Lista cu salariariile angajatilor cu salariul > 3000
Salariatul Boaba are salariul: 3500
Salariatul Cornea are salariul: 3267
Salariatul Enaru are salariul: 3800
```

```
PL/SQL procedure successfully completed.
```

6. Sa se modifice preturile tuturor produselor in functie de de cantitatea acestora folosindu-se structura while.

```
BEGIN
```

```
SELECT COUNT(*) INTO v_count FROM produse_proiect;
```

```
WHILE v_index <= 22 LOOP
```

```
SELECT pret, cantitate INTO v_pret_produș, v_cantitate
```

```
FROM produse_proiect WHERE id_produș = v_index;
```

```
IF v_cantitate >= 10 THEN
```

```
    v_pret_produș := v_pret_produș * 0.9;
```

```
ELSE
```

```
    v_pret_produș := v_pret_produș * 1.2;
```

```
END IF;
```

```
UPDATE produse_proiect SET pret = v_pret_produș WHERE id_produș = v_index;
```

```
DBMS_OUTPUT.PUT_LINE('Produsul ' || v_index || ' are pretul nou de ' || v_pret_produș);
```

```
v_index := v_index + 1;
```

```
END LOOP;
```

```
END;
```

/

```
Produsul 11 are pretul nou de 2.7  
Produsul 12 are pretul nou de 3.6  
Produsul 13 are pretul nou de 4.5  
Produsul 14 are pretul nou de 2.7  
Produsul 15 are pretul nou de 2.7  
Produsul 16 are pretul nou de 3.6  
Produsul 17 are pretul nou de 1.8  
Produsul 18 are pretul nou de .9  
Produsul 19 are pretul nou de 4.5  
Produsul 20 are pretul nou de 1.8  
Produsul 21 are pretul nou de 14.4  
Produsul 22 are pretul nou de 8.4
```

```
PL/SQL procedure successfully completed.
```

4. Tema 2 – proiect - excepții

1. "Să se obțină adresa furnizorului cu ID-ul 99 din tabela FURNIZOR_PROIECT și să se afișeze adresa respectivă. Dacă nu există niciun furnizor cu acest ID, să se afișeze un mesaj corespunzător."

```
DECLARE
```

```
    v_furnizor_adresa VARCHAR2(20);
```

```
BEGIN
```

```
    SELECT ADRESA INTO v_furnizor_adresa FROM FURNIZOR_PROIECT WHERE  
ID_FURNIZOR = 99;
```

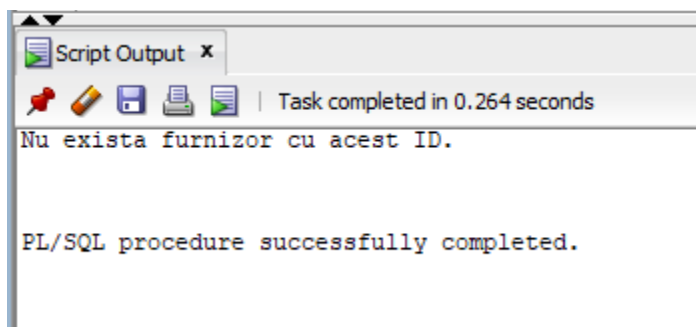
```
    DBMS_OUTPUT.PUT_LINE('Adresa furnizorului este: ' || v_furnizor_adresa);
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Nu exista furnizor cu acest ID.');
```

```
END;
```



2. Să se afișeze numărul de comenzi finalizate cu succes de catre un anumit angajat, pe baza ID-ului angajatului dat ca parametru. Dacă angajatul nu există în baza de date, afișați un mesaj corespunzător. În cazul în care apar alte erori în timpul executării codului, afișați un mesaj general de eroare. În cazul în care angajatul nu s-a ocupat de comenzi să se arunce o excepție personalizată.

```
DECLARE
```

```
  v_id_angajat ANGAJATI_PROIECT.ID_ANGAJAT%TYPE := 105;
```

```
  v_nr_comenzi NUMBER(3);
```

```
BEGIN
```

```
  SELECT COUNT(*) INTO v_nr_comenzi FROM COMENZI_PROIECT WHERE ID_ANGAJAT =  
  v_id_angajat;
```

```
  IF v_nr_comenzi = 0 THEN
```

```
    RAISE_APPLICATION_ERROR(-20001, 'Angajatul cu id-ul ' || v_id_angajat || ' nu s-a ocupat de nicio  
comanda.');
```

```
  END IF;
```

```
  DBMS_OUTPUT.PUT_LINE('Angajatul cu id-ul ' || v_id_angajat || ' are ' || v_nr_comenzi || ' comenzi.');
```

```
EXCEPTION
```

```
  WHEN NO_DATA_FOUND THEN
```

```
    DBMS_OUTPUT.PUT_LINE('Angajatul cu id-ul ' || v_id_angajat || ' nu exista!');
```

```
  WHEN OTHERS THEN
```

```
    DBMS_OUTPUT.PUT_LINE('A aparut o eroare!');
```

```
END;
```

```
Angajatul cu id-ul 2 are 2 comenzi.
```

```
PL/SQL procedure successfully completed.
```

3. Sa se selecteze primii cinci angajați cu cel mai mare salariu și sa se afiseze numele, prenumele și salariul în ordine descrescătoare a salariului. În cazul în care cursorul este deja deschis, se va afișa un mesaj de avertizare, iar în cazul oricărei alte erori, se va afișa un mesaj de eroare.

```
DECLARE
```

```
    v_nume VARCHAR2(20);
```

```
    v_prenume VARCHAR2(20);
```

```
    v_salariu angajati_proiect.salariu%type;
```

```
    CURSOR c_salarii IS SELECT nume, prenume, salariu FROM angajati_proiect ORDER BY salariu  
DESC;
```

```
BEGIN
```

```
    OPEN c_salarii;
```

```
    FOR r_salarii IN 1..5 LOOP
```

```
        FETCH c_salarii INTO v_nume, v_prenume, v_salariu;
```

```
        EXIT WHEN c_salarii%NOTFOUND;
```

```
        DBMS_OUTPUT.PUT_LINE(r_salarii || ' ' || v_nume || ' ' || v_prenume || ' - ' || v_salariu);
```

```
    END LOOP;
```

```
    CLOSE c_salarii;
```

```
EXCEPTION
```

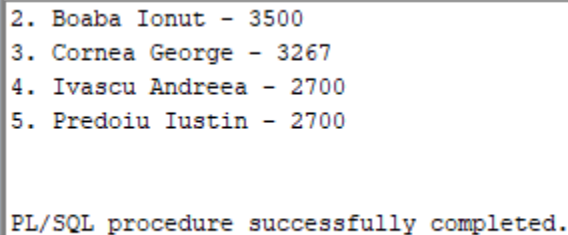
```
    WHEN CURSOR_ALREADY_OPEN THEN
```

```
        DBMS_OUTPUT.PUT_LINE('Cursorul este deja deschis');
```

WHEN OTHERS THEN

DBMS_OUTPUT.PUT_LINE('A aparut o eroare: ' || SQLERRM);

END;



```
2. Boaba Ionut - 3500
3. Cornea George - 3267
4. Ivascu Andreea - 2700
5. Predoiu Iustin - 2700

PL/SQL procedure successfully completed.
```

4. Sa se implementeze un bloc PL/SQL care să verifice dacă o funcție există în tabela functie_proiect în funcție de un ID dat. Dacă funcția există, sa se insereze un nou angajat în tabela angajati_proiect. În caz contrar, sa se declanșeze o excepție personalizată.

DECLARE

v_id_angajat angajati_proiect.id_angajat%TYPE := 1;

v_nume angajati_proiect.numename%TYPE := 'Doe';

v_prenume angajati_proiect.prenume%TYPE := 'John';

v_id_functie functie_proiect.id_functie%TYPE := 999; -- Functie inexistentă

functie_inexistenta_exception EXCEPTION;

PRAGMA EXCEPTION_INIT(functie_inexistenta_exception, -20001);

BEGIN

SELECT COUNT(*) INTO v_id_functie

FROM functie_proiect

WHERE id_functie = v_id_functie;

IF v_id_functie = 0 THEN

RAISE functie_inexistenta_exception;

END IF;

INSERT INTO angajati_proiect (id_angajat, nume, prenume, id_functie)

```

VALUES (v_id_angajat, v_nume, v_prenume, v_id_functie);
DBMS_OUTPUT.PUT_LINE('Angajat inserat cu succes!');
EXCEPTION
WHEN functie_inexistenta_exception THEN
    DBMS_OUTPUT.PUT_LINE('Exceptie: Functie inexistentă!');
WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLCODE || ' - ' || SQLERRM);
END;
/

```

Exceptie: Functie inexistentă!

PL/SQL procedure successfully completed.

5. Să se verifice dacă un anumit angajat există în tabela angajati_proiect și dacă are comenzi asociate în tabelul comenzi_proiect. În cazul în care angajatul nu există sau nu are comenzi, afișați un mesaj corespunzător. În caz contrar, afișați numele și salariul angajatului.

```

DECLARE
v_id_angajat angajati_proiect.id_angajat%TYPE := 10;
v_nume_angajat angajati_proiect.nume%TYPE;
v_salariu_angajat angajati_proiect.salariu%TYPE;

angajat_inexistent_exception EXCEPTION;
PRAGMA EXCEPTION_INIT(angajat_inexistent_exception, -20001);
v_comenzi NUMBER := 0;
BEGIN
SELECT COUNT(*) INTO v_comenzi FROM comenzi_proiect WHERE id_angajat = v_id_angajat;
IF v_comenzi = 0 THEN

```

```

        RAISE angajat_inexistent_exception;
    END IF;

    DBMS_OUTPUT.PUT_LINE('Numele angajatului: ' || v_numa_angajat);
    DBMS_OUTPUT.PUT_LINE('Salariul angajatului: ' || v_salariu_angajat);

EXCEPTION

    WHEN angajat_inexistent_exception THEN

    DBMS_OUTPUT.PUT_LINE('Angajatul cu ID-ul ' || v_id_angajat || ' nu există sau nu are comenzi asociate.');
```

```

    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLCODE || ' - ' || SQLERRM);
END;

/

```

```

PL/SQL procedure successfully completed.

Angajatul cu ID-ul 15 nu exista sau nu are comenzi asociate.

PL/SQL procedure successfully completed.

```

6. Sa se afiseze detaliile angajatilor in ordine descrescatoare folosit un cursor explicit fara parametrii.

```

DECLARE

CURSOR c_angajati IS

    SELECT *

    FROM angajati_proiect

```



```

ORDER BY salariu DESC;

v_angajat angajati_proiect%ROWTYPE;

BEGIN

OPEN c_angajati;

LOOP

    FETCH c_angajati INTO v_angajat;

    EXIT WHEN c_angajati%NOTFOUND;

    DBMS_OUTPUT.PUT_LINE('ID Angajat: ' || v_angajat.id_angajat);

    DBMS_OUTPUT.PUT_LINE('Nume: ' || v_angajat.numa);

    DBMS_OUTPUT.PUT_LINE('Prenume: ' || v_angajat.prenume);

    DBMS_OUTPUT.PUT_LINE('Salariu: ' || v_angajat.salariu);

    DBMS_OUTPUT.PUT_LINE('-----');

END LOOP;

CLOSE c_angajati;

END;

/

```

```

ID Angajat: 1
Nume: Boaba
Prenume: Ionut
Salariu: 3850
-----
ID Angajat: 7
Nume: Enaru
Prenume: Olivia
Salariu: 3800
-----
ID Angajat: 3
Nume: Cornea
Prenume: George
Salariu: 3267
-----
ID Angajat: 9
Nume: Ivascu
Prenume: Andreea
Salariu: 2700

```

7. Sa se calculeze suma totala a produselor pentru un anumit furnizor folosind un cursor explicit.

```
DECLARE
v_id_furnizor furnizor_proiect.id_furnizor%TYPE := 1001;
v_suma_totala NUMBER := 0;
CURSOR c_produce(p_id_furnizor IN furnizor_proiect.id_furnizor%TYPE) IS
    SELECT SUM(cantitate) AS suma_totala
    FROM produse_proiect
    WHERE id_furnizor = p_id_furnizor;
BEGIN
    OPEN c_produce(v_id_furnizor);
    FETCH c_produce INTO v_suma_totala;
    CLOSE c_produce;

    DBMS_OUTPUT.PUT_LINE('Cantitatea totală a produselor pentru furnizorul cu ID ' || v_id_furnizor || '
este: ' || v_suma_totala);
END;
```

/

```
PL/SQL procedure successfully completed.
```

```
Cantitatea totala a produselor pentru furnizorul cu ID 1001 este: 80
```

```
PL/SQL procedure successfully completed.
```

8. Sa se afiseze toate comenzile realizate de un anumit angajat, identificat prin ID-ul angajatului, folosind un cursor explicit.

```
DECLARE

CURSOR c_comenzi (p_id_angajat IN angajati_proiect.id_angajat%TYPE) IS

    SELECT * FROM comenzi_proiect WHERE id_angajat = p_id_angajat;

v_id_angajat angajati_proiect.id_angajat%TYPE := 10;

BEGIN

FOR rec IN c_comenzi(v_id_angajat) LOOP

    DBMS_OUTPUT.PUT_LINE('ID Comandă: ' || rec.id_comanda || ', Valoare: ' || rec.pret_final);

END LOOP;

END;

/
```

```
PL/SQL procedure successfully completed.
```

```
ID Comanda: 200, Valoare: 400
```

```
ID Comanda: 209, Valoare: 280
```

```
ID Comanda: 204, Valoare: 450
```

```
ID Comanda: 211, Valoare: 800
```

```
PL/SQL procedure successfully completed.
```

5. Tema 3 – proiect – funcții / proceduri / pachete

1. Realizeaza o procedura PL/SQL prin intermediul careia sa se majoreze salariul angajatului cu id_angajat = p_id_angajat. Procedura modifica_salariul primește doi parametri: p_id_angajat și procent

```
CREATE OR REPLACE
PROCEDURE modifica_salariul_procent
(p_id_angajat IN angajati_proiect.id_angajat%type, procent IN number)
IS
v_salariul angajati_proiect.salariu%type;
BEGIN
Select salariu into v_salariul from angajati_proiect where id_angajat = p_id_angajat;
dbms_output.put_line('Angajatul are salariul de '||v_salariul);
Update angajati_proiect
Set salariu = salariu * (1 + procent/100)
Where id_angajat = p_id_angajat;
Select salariu into v_salariul from angajati_proiect where id_angajat = p_id_angajat;
Dbms_output.put_line('Angajatul are acum salariul de '||v_salariul);
END;
/
show errors;

begin
  modifica_salariul_procent(1, 10);
end;
/
```

ID_ANGAJAT	PRENUME	NUME	TELEFON	SALARIU	ID_FUNCTIE
1	1 Ionut	Boaba	0770550792	3850	103
2	2 Andrei	Dumitru	0730524673	2500	100
3	3 George	Cornea	0745798792	3267	102
4	4 Raluca	Balan	0773569803	2100	104
5	5 Teodor	Stoica	0733333390	2400	101
6	6 Stefania	Dumitrescu	0756290689	2500	100
7	7 Olivia	Enaru	0726216601	3800	103
8	8 Iustin	Predoiu	0774802552	2700	102
9	9 Andreea	Ivascu	0725908076	2700	102

Boaba Ionut avea salariul initial egal cu 3500

- Realizeaza o funcție care primește ca parametru un ID de produs și returnează categoria de preț în care se încadrează produsul. Funcția verifică prețul produsului și returnează un mesaj corespunzător în funcție de intervalul de preț în care se încadrează. În cazul în care nu există un produs cu ID-ul specificat, funcția returnează un mesaj care indică acest lucru.

```
Create or replace function categorie(p_id produse_proiect.id_produs%type)
```

```
return varchar2
```

```
is
```

```
v_pret produse_proiect.pret%type;
```

```
begin
```

```
select pret into v_pret from produse_proiect where id_produs = p_id;
```

```
if v_pret < 3 then return 'produs ieftin';
```

```
elsif v_pret between 3 and 5 then return 'produs de buget';
```

```
else return 'produs scump';
```

```
end if;
```

```
exception
```

```
when no_data_found then
```

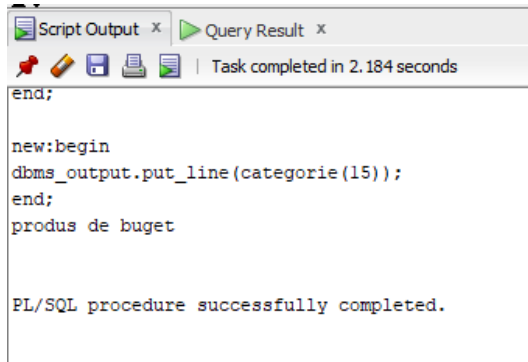
```
return 'Nu exista produs cu acest id';
```

```
end;
```

```

begin
dbms_output.put_line(categorie(&id));
end;
/

```



The screenshot shows a window titled 'Script Output' and 'Query Result'. Below the title bar, there is a status bar that says 'Task completed in 2.184 seconds'. The main area of the window displays the following text:

```

end;

new:begin
dbms_output.put_line(categorie(15));
end;
produs de buget

PL/SQL procedure successfully completed.

```

3. Să se realizeze o procedură care primește ca parametru id-ul unui angajat și returnează numele și salariul acestuia, în funcție de următoarele condiții: angajatul trebuie să aibă cel puțin 2 comenzi plasate și în funcție de nivelul salariului să i se aplice o creștere salarială diferită. În final, să se afișeze salariul inițial și salariul final al angajatului.

```

CREATE OR REPLACE PROCEDURE cauta_angajat
(p_id_angajat IN angajati_proiect.id_angajat%type,
p_nume OUT angajati_proiect.nume%type,
p_salariul OUT angajati_proiect.salariu%type)
IS
BEGIN
Select nume, salariu into p_nume, p_salariul from angajati_proiect where id_angajat = p_id_angajat
AND id_angajat IN(SELECT id_angajat FROM comenzi_proiect HAVING COUNT(ID_ANGAJAT)>=
2
GROUP BY ID_ANGAJAT);
DBMS_OUTPUT.PUT_LINE(' Angajatul '||p_nume||' are salariul initial de: '||p_salariul);
p_salariul:= CASE WHEN p_salariul < 3000 then p_salariul * 1.1

```

```

        WHEN p_salariul between 3000 and 3200 then p_salariul * 1.05
        ELSE p_salariul * 1.03
    END;

    DBMS_OUTPUT.PUT_LINE(' Angajatul '||p_nume||' are salariul final de: '||p_salariul);
END;
/

```

```

DECLARE
v_nume angajati_proiect.nume%type;
v_salariul angajati_proiect.salariu%type;
BEGIN
Cauta_angajat(10, v_nume, v_salariul);
END;
/

```

```

Procedure CAUTA_ANGAJAT compiled

Angajatul Diaconu are salariul initial de: 2500
Angajatul Diaconu are salariul final de: 2750

PL/SQL procedure successfully completed.

```

4. Sa se realizeze o functie PL/SQL care să primească ca parametru un ID de produs și o cantitate, iar în baza acestor date să se actualizeze cantitatea produsului din tabela produse_proiect. În cazul în care produsul nu există în tabelă, se va afișa un mesaj corespunzător. De asemenea, în cazul în care cantitatea introdusă este negativă, se va genera o excepție cu mesaj corespunzător. La final, se va afișa noua cantitate a produsului actualizat.

```

CREATE OR REPLACE FUNCTION modifica_stoc
(p_id_produc IN produse_proiect.id_produc%type,
 p_cantitate IN produse_proiect.cantitate%type)
RETURN produse_proiect.cantitate%type
IS
    v_cantitate_actuala produse_proiect.cantitate%type;
BEGIN
    SELECT cantitate INTO v_cantitate_actuala FROM produse_proiect WHERE id_produc =
    p_id_produc;

    IF p_cantitate < 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Cantitatea introdusa este negativa!');
    ELSE
        FOR i IN 1..ABS(p_cantitate) LOOP
            IF p_cantitate > 0 THEN
                v_cantitate_actuala := v_cantitate_actuala + 1;
            ELSE
                v_cantitate_actuala := v_cantitate_actuala - 1;
            END IF;
        END LOOP;
        UPDATE produse_proiect SET cantitate = v_cantitate_actuala WHERE id_produc = p_id_produc;
        DBMS_OUTPUT.PUT_LINE('Produsul are cantitatea noua egala cu ' || v_cantitate_actuala || ' kg');
        -- COMMIT;
    END IF;

    RETURN v_cantitate_actuala;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu exista produs cu acest id!');

```



```
END;
```

```
/
```

```
DECLARE
```

```
BEGIN
```

```
dbms_output.put_line(modifica_stoc(14, 10));
```

```
END;
```

```
/
```

```
Function MODIFICA_STOC compiled

Produsul are cantitatea noua egala cu 120 kg
120

PL/SQL procedure successfully completed.
```

5. Realizeaza o functie PL/SQL care sa primeasca ca parametru un ID de furnizor si numele furnizorului si sa numere cate produse sunt expediate de catre acest furnizor.

```
CREATE OR REPLACE FUNCTION numar_produce_furnizor(p_id_furnizor IN
furnizor_proiect.id_furnizor%TYPE, p_nume IN furnizor_proiect.nume_furnizor%TYPE)
```

```
RETURN NUMBER
```

```
IS v_numar_produce NUMBER := 0;
```

```
BEGIN
```

```
FOR produs IN (SELECT id_furnizor FROM produse_proiect WHERE id_furnizor = p_id_furnizor)
LOOP v_numar_produce := v_numar_produce + 1;
```

```
END LOOP;
```

```
DBMS_OUTPUT.PUT_LINE('Furnizorul ' || p_nume || ' expediaza ' || v_numar_produce || ' produse.');
```

```
RETURN v_numar_produce;
```

```
END;
```

/

DECLARE

v_numar_produce NUMBER;

BEGIN

v_numar_produce := numar_produce_furnizor(1007, 'Nestle SRL');

END;

/

PL/SQL procedure successfully completed.

Furnizorul Nestle SRL expedieaza 3 produse.

PL/SQL procedure successfully completed.

6. Realizeaza o procedura care să returneze data încheierii și valoarea celei mai recente comenzi înregistrate în tabela comenzi_proiect.

CREATE OR REPLACE PROCEDURE comanda_recenta(

p_data OUT comenzi_proiect.data_comanda%TYPE,

p_valoare OUT NUMBER

)

IS

BEGIN

SELECT data_comanda, pret_final

INTO p_data, p_valoare

FROM comenzi_proiect

WHERE data_comanda = (SELECT MAX(data_comanda) FROM comenzi_proiect);

EXCEPTION

WHEN NO_DATA_FOUND THEN

```

        DBMS_OUTPUT.PUT_LINE('Nu există comenzi înregistrate.');
```

WHEN OTHERS THEN

```

        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLCODE || ' - ' || SQLERRM);
END;

/

DECLARE

    v_data comenzi_proiect.data_comanda%TYPE;
    v_valoare NUMBER;
BEGIN

    comanda_recenta(v_data, v_valoare);

    DBMS_OUTPUT.PUT_LINE('Data încheierii celei mai recente comenzi: ' || v_data);
    DBMS_OUTPUT.PUT_LINE('Valoarea celei mai recente comenzi: ' || v_valoare);
END;

/
```

```

Data încheierii celei mai recente comenzi: 05-JAN-23
Valoarea celei mai recente comenzi: 400
```

```

PL/SQL procedure successfully completed.
```

7. Realizati un pachet care sa contina:

- o functie care returneaza suma comenzilor încheiate de catre un angajat al carui id este dat ca parametru. Tratati cazul în care nu exista clientul specificat;
- o procedura care foloseste functia de mai sus pentru a returna media comenzilor realizate de toti angajatii

```

CREATE OR REPLACE PACKAGE pachet_comenzi AS
```

```
FUNCTION suma_comenzi_angajat(p_id_angajat IN comenzi_proiect.id_angajat%TYPE) RETURN  
NUMBER;
```

```
PROCEDURE media_comenzi_angajati;
```

```
END pachet_comenzi;
```

```
/
```

```
CREATE OR REPLACE PACKAGE BODY pachet_comenzi AS
```

```
FUNCTION suma_comenzi_angajat(p_id_angajat IN comenzi_proiect.id_angajat%TYPE) RETURN  
NUMBER IS
```

```
    v_suma_comenzi NUMBER := 0;
```

```
BEGIN
```

```
    SELECT SUM(pret_final) INTO v_suma_comenzi
```

```
    FROM comenzi_proiect
```

```
    WHERE id_angajat = p_id_angajat;
```

```
    RETURN v_suma_comenzi;
```

```
EXCEPTION
```

```
    WHEN NO_DATA_FOUND THEN
```

```
        RETURN 0;
```

```
END suma_comenzi_angajat;
```

```
PROCEDURE media_comenzi_angajati IS
```

```
    v_numar_angajati NUMBER := 10;
```

```
    v_suma_totala NUMBER := 0;
```

```
    v_media_comenzi NUMBER :=0;
```

```
    cursor c1 is SELECT id_angajat FROM comenzi_proiect;
```

```
BEGIN
```

```
    SELECT COUNT(*) INTO v_numar_angajati FROM angajati_proiect;
```

```
    FOR rec IN c1 LOOP
```

```
        v_suma_totala := v_suma_totala + suma_comenzi_angajat(rec.id_angajat);
```

```

END LOOP;

v_media_comenzi := v_suma_totala / v_numar_angajati;

DBMS_OUTPUT.PUT_LINE('Media comenzilor realizate de angajati: ' || v_media_comenzi);

END media_comenzi_angajati;

END pachet_comenzi;

/

```

```

DECLARE

v_suma_comenzi NUMBER;

BEGIN

v_suma_comenzi := pachet_comenzi.suma_comenzi_angajat(10);

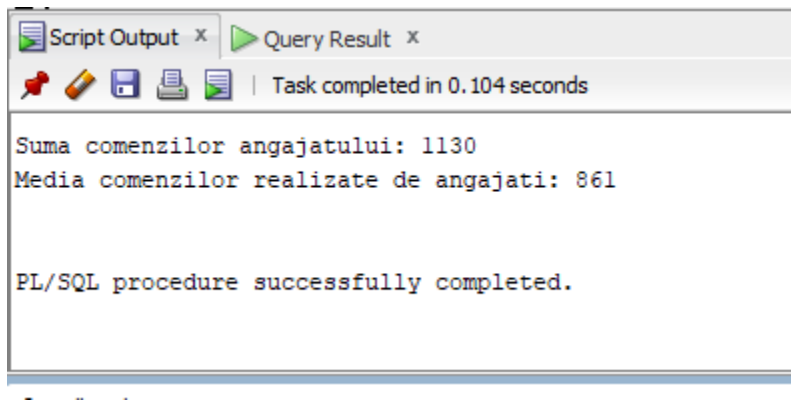
DBMS_OUTPUT.PUT_LINE('Suma comenzilor angajatului: ' || v_suma_comenzi);

pachet_comenzi.media_comenzi_angajati;

END;

/

```



8. Realizati un pachet care sa contina:

- modificarea valori comenzii cu un anumit id și verificarea dacă modificarea a fost efectuată cu succes
- stergerea unui angajat cu un anumit id si sa se afiseze un mesaj corespunzător în funcție de rezultatul stergerii angajatului

```

CREATE OR REPLACE PACKAGE BODY pachet_modificari AS

    PROCEDURE modifica_valoare_comanda(p_id_comanda IN comenzi_proiect.id_comanda%TYPE,
    p_noua_valoare IN comenzi_proiect.pret_final%TYPE) IS

    BEGIN

        UPDATE comenzi_proiect

        SET pret_final = p_noua_valoare

        WHERE id_comanda = p_id_comanda;

    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Valoarea comenzii cu id-ul ' || p_id_comanda || ' a fost modificată cu
    succes.');
```

```

EXCEPTION

    WHEN NO_DATA_FOUND THEN

        DBMS_OUTPUT.PUT_LINE('Nu există o comandă cu id-ul specificat.');
```

```

    WHEN OTHERS THEN

        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLCODE || ' - ' || SQLERRM);

END modifica_valoare_comanda;

FUNCTION sterge_angajat(p_id_angajat IN angajati_proiect.id_angajat%TYPE) RETURN BOOLEAN
IS

    v_exista_angajat NUMBER;

    BEGIN

        SELECT COUNT(*)

        INTO v_exista_angajat

        FROM angajati_proiect

        WHERE id_angajat = p_id_angajat;

    IF v_exista_angajat > 0 THEN

        DELETE FROM angajati_proiect

        WHERE id_angajat = p_id_angajat;

        COMMIT;

```

```

        DBMS_OUTPUT.PUT_LINE('Angajatul cu id-ul ' || p_id_angajat || ' a fost șters.');
```

RETURN TRUE;

ELSE

```

        DBMS_OUTPUT.PUT_LINE('Nu există un angajat cu id-ul specificat.');
```

RETURN FALSE;

END IF;

EXCEPTION

WHEN OTHERS THEN

```

        DBMS_OUTPUT.PUT_LINE('Eroare: ' || SQLCODE || ' - ' || SQLERRM);
```

RETURN FALSE;

END sterge_angajat;

END pachet_modificari;

/

```

BEGIN
```

pachet_modificari.modifica_valoare_comanda(1001, 1500); la 1500

IF pachet_modificari.sterge_angajat(765) THEN

```

        DBMS_OUTPUT.PUT_LINE('Angajatul a fost sters cu succes.');
```

ELSE

```

        DBMS_OUTPUT.PUT_LINE('Angajatul nu a putut fi sters.');
```

END IF;

END;

/

```
1 row inserted.
```

```

Valoarea comenzii cu id-ul 1001 a fost modificata cu succes.
Angajatul cu id-ul 765 a fost șters.
Angajatul a fost sters cu succes.
```

6. *Tema 4 – proiect – declansatori*

1. Sa se verifice daca valoarea campului ,pret' din tabela produse_proiect este mai mare decat 0 inainte de inserarea unui nou produs. Daca valoarea nu respecta aceasta conditie, triggerul ar trebui să arunce o exceptie sau sa anuleze operatiunea de inserare.

```
CREATE OR REPLACE TRIGGER pret_pozitiv
```

```
BEFORE INSERT ON produse_proiect
```

```
FOR EACH ROW
```

```
BEGIN
```

```
IF :NEW.pret < 0 THEN
```

```
    RAISE_APPLICATION_ERROR(-20001, 'Pretul produsului trebuie sa fie pozitiv.');
```

```
END IF;
```

```
END;
```

```
/
```

```
INSERT INTO produse_proiect (id_produs, denumire, cantitate, pret, id_furnizor, id_comanda) VALUES  
(500, 'mazare', 50, -10, 1001, 202);
```

```
Trigger PRET_POZITIV compiled
```

```
Error starting at line : 13 in command -
```

```
INSERT INTO produse_proiect (id_produs, denumire, cantitate, pret, id_furnizor, id_comanda) VALUES (500, 'mazare', 50, -10, 1001, 202)
```

```
Error report -
```

```
ORA-20001: Pretul produsului trebuie sa fie pozitiv.
```

```
ORA-06512: at "BOABAI_55.PRET_POZITIV", line 3
```

```
ORA-04088: error during execution of trigger 'BOABAI_55.PRET_POZITIV'
```


2. Sa se verifice dacă furnizorul care urmează ss fie sters are produse asociate în tabela produse_proiect. Daca exista produse asociate, triggerul ar trebui să arunce o exceptie sau sa anuleze operatiunea de stergere a furnizorului.

```
CREATE OR REPLACE TRIGGER stergere_furnizor
BEFORE DELETE ON furnizor_proiect
FOR EACH ROW
DECLARE
    v_count NUMBER;
BEGIN
    SELECT COUNT(*)
    INTO v_count
    FROM produse_proiect
    WHERE id_furnizor = :OLD.id_furnizor;

    IF v_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Nu se poate sterge furnizorul. Există produse asociate.');
```

ELSE

```
        DELETE FROM furnizor_proiect
        WHERE id_furnizor = :OLD.id_furnizor;

        COMMIT;
    END IF;
END;
```

/

```
DELETE FROM furnizor_proiect WHERE id_furnizor = 1001;
```

```

Error starting at line : 38 in command -
DELETE FROM furnizor_proiect WHERE id_furnizor = 1001
Error report -
ORA-20002: Nu se poate sterge furnizorul. Exista produse asociate.
ORA-06512: at "BOABAI_55.STERGERE_FURNIZOR", line 10
ORA-04088: error during execution of trigger 'BOABAI_55.STERGERE_FURNIZOR'

```

3. Sa se inregistreze în tabela temp_log_proiect toate operatiile de inserare, actualizare si stergere efectuate asupra tabelei comenzi_proiect. Inregistrările trebuie să contina tipul operatiei (INSERT, UPDATE sau DELETE), utilizatorul care a efectuat operatia si data curents.

```

CREATE TABLE TEMP_LOG_PROIECT (
    tip_operatie VARCHAR2(20),
    utilizator VARCHAR2(50),
    data_operatie DATE
);

```

```

CREATE OR REPLACE TRIGGER comenzi_temp_log_trigger
AFTER INSERT OR UPDATE OR DELETE ON comenzi_proiect
FOR EACH ROW
DECLARE
    v_tip_operatie VARCHAR2(20);
BEGIN
    IF INSERTING THEN
        v_tip_operatie := 'INSERT';
    ELSIF UPDATING THEN
        v_tip_operatie := 'UPDATE';
    ELSIF DELETING THEN
        v_tip_operatie := 'DELETE';
    END IF;

```

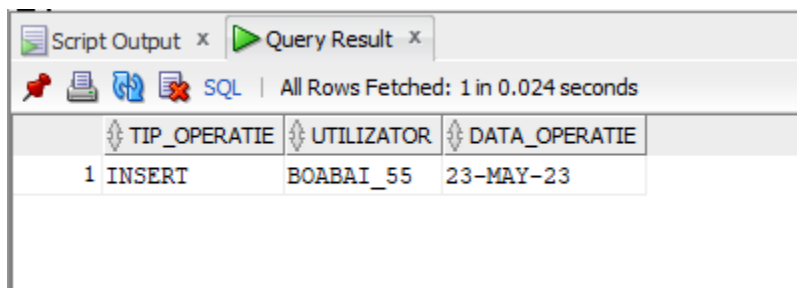
```
INSERT INTO temp_log_proiect (tip_operatie, utilizator, data_operatie)
```

```
VALUES (v_tip_operatie, USER, SYSDATE);
```

```
END;
```

```
/
```

```
INSERT INTO COMENZI_PROIECT (ID_COMANDA, NUME_PRODUSE, PRET_FINAL,  
ID_ANGAJAT) VALUES (211, 'lapte, oua,branza', 800, 10);
```



The screenshot shows a SQL query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with three columns: 'TIP_OPERATIE', 'UTILIZATOR', and 'DATA_OPERATIE'. The table contains one row of data, which is the result of the SQL query executed. The status bar indicates 'All Rows Fetched: 1 in 0.024 seconds'.

	TIP_OPERATIE	UTILIZATOR	DATA_OPERATIE
1	INSERT	BOABAI_55	23-MAY-23