Proiect SGBD

-Șerban Andrei, 242-

Proiectul ales de mine este o bază de date pentru o firmă de agricultură, proiectat pentru a eficientiza operațiunile afacerilor agricole moderne. Acesta pune împreună datele legate de managementul departamentelor, activitățile angajaților, interacțiuni cu clienți, procesare comenzi, inventarul produselor si echipamentelor agricole. Baza de date oferă perspective in timp real asupra fiecărui aspect al operațiunilor firmei, ajutând la procesul decizional, eficienta operațională si satisfacția clienților.

Următoarele diagrame prezinta funcționalitatea bazei de date.

Diagrama entitate-relație:

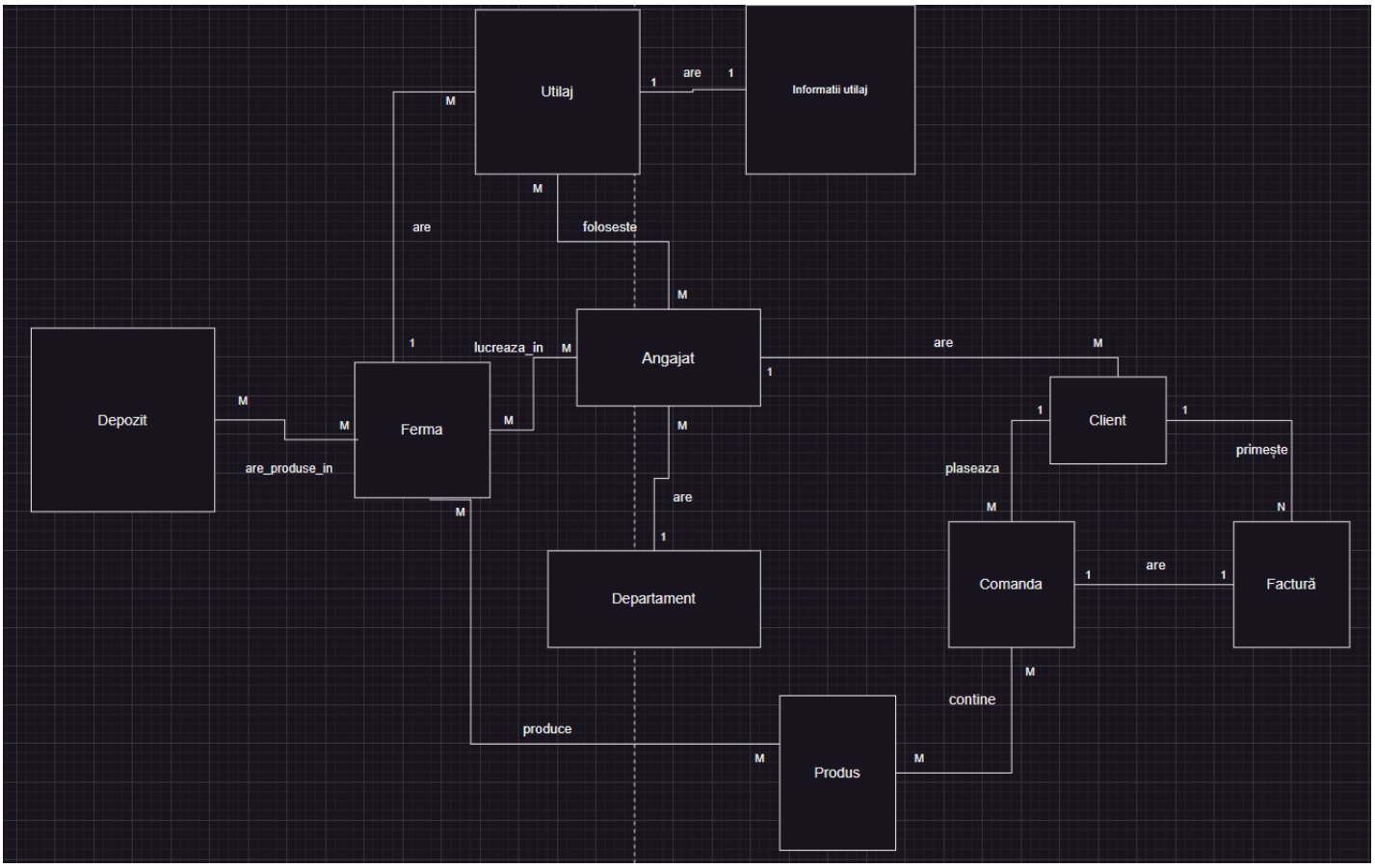
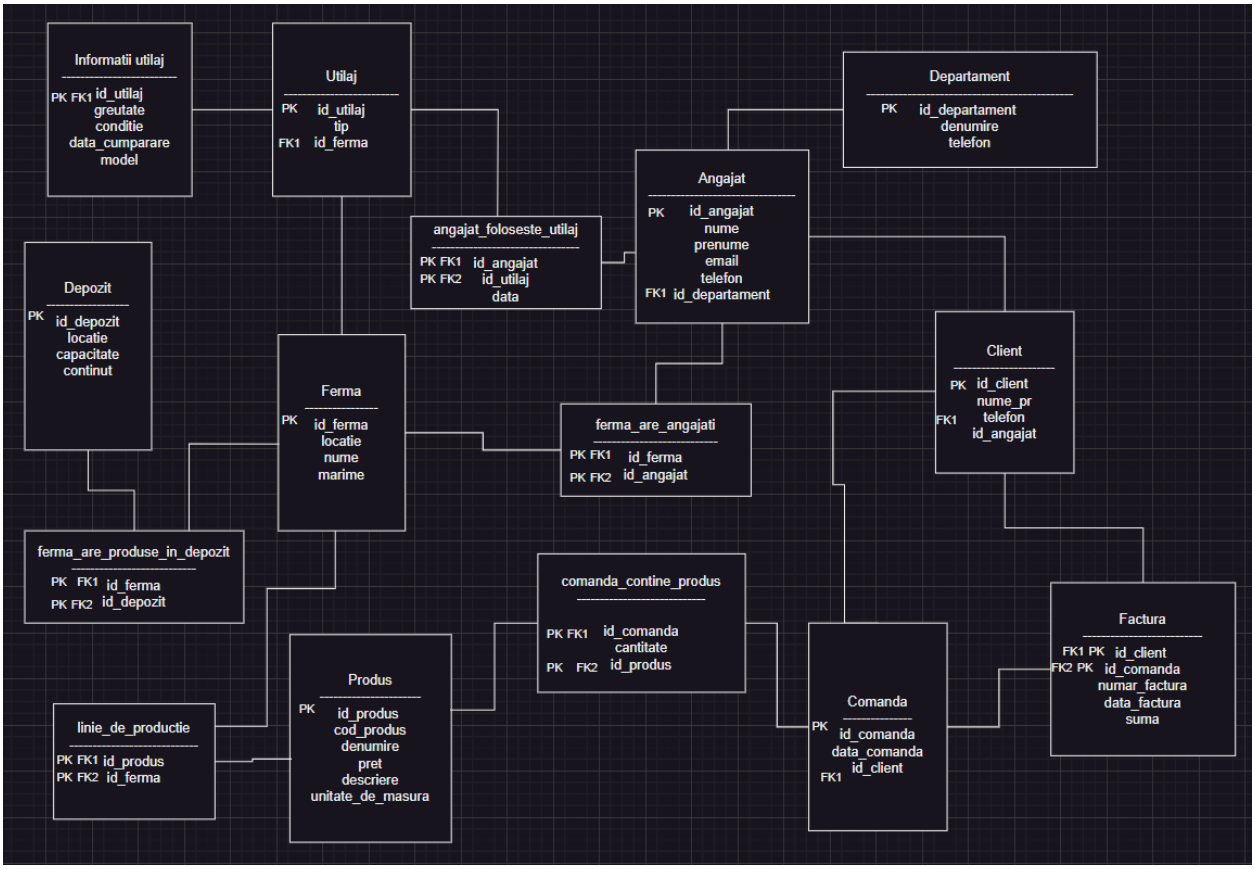


Diagrama conceptuală:



Codul folosit pentru crearea bazei de date:

```
CREATE TABLE Departament (
   id_departament NUMBER PRIMARY KEY,
   denumire VARCHAR2(100) NOT NULL,
   telefon VARCHAR2(15)
);
 CREATE TABLE Angajat (
   id_angajat NUMBER PRIMARY KEY,
   nume VARCHAR2(50) NOT NULL,
   prenume VARCHAR2(50) NOT NULL,
   email VARCHAR2(100),
   telefon VARCHAR2(15),
   id_departament NUMBER,
   CONSTRAINT fk_angajat_departament
FOREIGN KEY (id_departament)
   REFERENCES Departament (id_departament)
);
```

```
CREATE TABLE Client (
   id_client NUMBER PRIMARY KEY,
   nume_pr VARCHAR2(100) NOT NULL,
   telefon VARCHAR2(15),
   id_angajat NUMBER,
   CONSTRAINT fk_client_angajat FOREIGN KEY (id_angajat)
   REFERENCES Angajat (id_angajat)
);


CREATE TABLE Comanda (
   id_comanda NUMBER PRIMARY KEY,
   cantitate NUMBER NOT NULL,
   data_comanda DATE NOT NULL,
   id_client NUMBER,
   CONSTRAINT fk_comanda_client FOREIGN KEY (id_client)
   REFERENCES Client (id_client)
);


CREATE TABLE Factura (
   id_comanda NUMBER ,
   id_client NUMBER ,
   PRIMARY KEY (id_comanda,id_client),
   numar_factura VARCHAR2(50) NOT NULL,
   data_factura DATE NOT NULL,
   suma NUMBER NOT NULL,
   CONSTRAINT fk_factura_client FOREIGN KEY (id_client)
   REFERENCES Client (id_client),
   CONSTRAINT fk_factura_client_2 FOREIGN KEY (id_comanda)
   REFERENCES Comanda (id_comanda)
);


CREATE TABLE Produs (
   id_produs NUMBER PRIMARY KEY,
   cod_produs VARCHAR2(50) NOT NULL,
   denumire VARCHAR2(100) NOT NULL,
   pret NUMBER NOT NULL,
   descriere VARCHAR2(255),
   unitate_de_masura VARCHAR2(50)
);


CREATE TABLE Depozit (
   id_depozit NUMBER PRIMARY KEY,
   locatie VARCHAR2(100) NOT NULL,
   capacitate NUMBER NOT NULL,
   continut VARCHAR2(255)
);
```

```
CREATE TABLE Ferma (
   id_ferma NUMBER PRIMARY KEY,
   locatie VARCHAR2(100) NOT NULL,
   nume VARCHAR2(100) NOT NULL,
   marime NUMBER NOT NULL
);


CREATE TABLE Utilaj (
   id_utilaj NUMBER PRIMARY KEY,
   tip VARCHAR2(100) NOT NULL,
   id_ferma NUMBER,
   CONSTRAINT fk_utilaj_ferma FOREIGN KEY
(id_ferma)
   REFERENCES Ferma (id_ferma)
);


CREATE TABLE Informatii_Utilaj (
   id_utilaj NUMBER PRIMARY KEY,
   greutate NUMBER NOT NULL,
   conditie VARCHAR2(100),
   data_cumparare DATE,
   model VARCHAR2(100),
   CONSTRAINT fk_utilaj_informatii FOREIGN
KEY (id_utilaj)
   REFERENCES Utilaj (id_utilaj)
);


CREATE TABLE Angajat_Foloseste_Utilaj (
   id_angajat NUMBER,
   id_utilaj NUMBER,
   data DATE NOT NULL,
   PRIMARY KEY (id_angajat, id_utilaj),
   CONSTRAINT fk_afu_angajat FOREIGN KEY
(id_angajat)
   REFERENCES Angajat (id_angajat),
   CONSTRAINT fk_afu_utilaj FOREIGN KEY
(id_utilaj)
   REFERENCES Utilaj (id_utilaj)
);


CREATE TABLE Ferma_Are_Angajati (
   id_ferma NUMBER,
   id_angajat NUMBER,
   PRIMARY KEY (id_ferma, id_angajat),
   CONSTRAINT fk_faa_ferma FOREIGN KEY
(id_ferma)
   REFERENCES Ferma (id_ferma),
   CONSTRAINT fk_faa_angajat FOREIGN KEY
(id_angajat)
   REFERENCES Angajat (id_angajat)
);
```

```
CREATE TABLE
Ferma_Are_Produse_In_Depozit (
   id_ferma NUMBER,
   id_depozit NUMBER,
   PRIMARY KEY (id_ferma, id_depozit),
   CONSTRAINT fk_fapd_ferma FOREIGN KEY
(id_ferma)
   REFERENCES Ferma (id_ferma),
   CONSTRAINT fk_fapd_depozit FOREIGN KEY
(id_depozit)
   REFERENCES Depozit (id_depozit)
);

CREATE TABLE Linie_De_Productie (
   id_produs NUMBER,
   id_ferma NUMBER,
   PRIMARY KEY (id_produs, id_ferma),
   CONSTRAINT fk_ldp_produs FOREIGN KEY
(id_produs)

   REFERENCES Produs (id_produs),
   CONSTRAINT fk_ldp_ferma FOREIGN KEY
(id_ferma)
   REFERENCES Ferma (id_ferma)
);

CREATE TABLE Comanda_Contine_Produs (
   id_comanda NUMBER,
   id_produs NUMBER,
   cantitate NUMBER NOT NULL,
   PRIMARY KEY (id_comanda, id_produs),
   CONSTRAINT fk_ccp_comanda FOREIGN KEY
(id_comanda)
   REFERENCES Comanda (id_comanda),
   CONSTRAINT fk_ccp_produs FOREIGN KEY
(id_produs)
   REFERENCES Produs (id_produs)
);
commit;
```

Datele introduse la inceput in baza de date sunt:

```
   INSERT INTO Departament VALUES (1,
'Contabilitate', '0312345679');
   INSERT INTO Departament VALUES (2,
'Vanzari', '0312345681');
```

INSERT INTO Departament VALUES (3, 'IT', '0312345682');

INSERT INTO Departament VALUES (4, 'Intretinere si Reparatii', '0312345683');

INSERT INTO Departament VALUES (5, 'Intretinere Culturi Agricole', '0312345684');

INSERT INTO Angajat VALUES (1, 'Popescu', 'Ion', 'ion.popescu@example.com', '0722345678', 5);

INSERT INTO Angajat VALUES (2, 'Ionescu', 'Maria', 'maria.ionescu@example.com', '0722345679', 1);

INSERT INTO Angajat VALUES (3, 'Georgescu', 'Andrei', 'andrei.georgescu@example.com', '0722345680', 4);

INSERT INTO Angajat VALUES (4, 'Dumitrescu', 'Elena', 'elena.dumitrescu@example.com', '0722345681', 1);

INSERT INTO Angajat VALUES (5, 'Stanescu', 'Mihai', 'mihai.stanescu@example.com', '0722345682', 5);

INSERT INTO Angajat VALUES (6, 'Ilie', 'Andrei', 'andrei.ilie@example.com', '0723000011', 2);

INSERT INTO Angajat VALUES (7, 'Popa', 'Ioana', 'ioana.popa@example.com', '0723000012', 4);

INSERT INTO Angajat VALUES (8, 'Nicolae', 'Raluca', 'raluca.nicolae@example.com', '0723000013', 3);

INSERT INTO Angajat VALUES (9, 'Marin', 'Dan', 'dan.marin@example.com', '0723000014', 2);

INSERT INTO Angajat VALUES (10, 'Dobre', 'Elena', 'elena.dobre@example.com', '0723000015', 3);

INSERT INTO Ferma VALUES (1, 'Craiova', 'Ferma legumicola din Craiova', '3');

INSERT INTO Ferma VALUES (2, 'Iasi', 'Ferma viticola din Iasi', '150');

INSERT INTO Ferma VALUES (3, 'Vaslui', 'Ferma de cereale de la Vaslui', '500');

INSERT INTO Ferma VALUES (4, 'Sibiu', 'Ferma pomicola din Sibiu', '60');

INSERT INTO Ferma VALUES (5, 'Galati', 'Ferma de cereale din Galati ', '1000');

INSERT INTO Produs VALUES (1, 'P001', 'Struguri', 2.99, 'Struguri rosii', 'kg');

INSERT INTO Produs VALUES (2, 'P002', 'Castraveti', 2.83, 'Castraveti cornison', 'kg');

INSERT INTO Produs VALUES (3, 'P003', 'Porumb', 850.00, 'Porumb', 't');

INSERT INTO Produs VALUES (4, 'P004', 'Orz', 800.00, 'Orz', 't');

INSERT INTO Produs VALUES (5, 'P005', 'Mere', 30.00, 'Mere verzi', 'lada');

INSERT INTO Produs VALUES (6, 'I001', 'Ingrasamant NPK', 17.60, 'Ingrasamant lichid', 'l');

INSERT INTO Produs VALUES (7, 'P001', 'Struguri', 3.50, 'Struguri albi', 'kg');

```
INSERT INTO Produs VALUES (8, 'P006', 'Pere', 35.99, 'Pere', 'lada');

INSERT INTO Produs VALUES (9, 'P007', 'Rosii', 3.50, 'Rosii Roma', 'kg');


INSERT INTO linie_de_productie VALUES (2, 1);

INSERT INTO linie_de_productie VALUES (1, 2);

INSERT INTO linie_de_productie VALUES (7, 2);

INSERT INTO linie_de_productie VALUES (3, 3);

INSERT INTO linie_de_productie VALUES (4, 3);

INSERT INTO linie_de_productie VALUES (5, 4);

INSERT INTO linie_de_productie VALUES (3, 5);

INSERT INTO linie_de_productie VALUES (4, 5);

INSERT INTO linie_de_productie VALUES (8, 4);

INSERT INTO linie_de_productie VALUES (9, 1);


INSERT INTO Client VALUES (1, 'Ion Ionescu', '0722000001', 1);

INSERT INTO Client VALUES (2, 'Maria Popescu', '0722000002', 2);

INSERT INTO Client VALUES (3, 'George Vasile', '0722000003', 3);

INSERT INTO Client VALUES (4, 'Elena Radu', '0722000004', 4);

INSERT INTO Client VALUES (5, 'Mihai Zaharia', '0722000005', 5);

INSERT INTO Client VALUES (6, 'Ion Ionescu', '0722002001', 3);


INSERT INTO Utilaj VALUES (1, 'Tractor', 1);

INSERT INTO Utilaj VALUES (6, 'Tractor', 2);

INSERT INTO Utilaj VALUES (10, 'Tractor', 4);

INSERT INTO Utilaj VALUES (7, 'Tractor', 3);

INSERT INTO Utilaj VALUES (8, 'Tractor', 5);

INSERT INTO Utilaj VALUES (9, 'Tractor', 5);

INSERT INTO Utilaj VALUES (2, 'Combinator', 1);

INSERT INTO Utilaj VALUES (11, 'Semantoare', 3);

INSERT INTO Utilaj VALUES (3, 'Semantoare', 5);

INSERT INTO Utilaj VALUES (4, 'Cultivator', 2);

INSERT INTO Utilaj VALUES (5, 'Irigator', 1);


INSERT INTO angajat_foloseste_utilaj VALUES (1, 1, TO_DATE('2022-04-24', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (3, 2, TO_DATE('2023-08-16', 'YYYY-MM-DD'));
```

```
INSERT INTO angajat_foloseste_utilaj VALUES (5, 3, TO_DATE('2023-05-08', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (1, 4, TO_DATE('2022-08-26', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (3, 5, TO_DATE('2021-06-25', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (5, 2, TO_DATE('2021-10-31', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (7, 3, TO_DATE('2022-11-18', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (7, 4, TO_DATE('2023-04-04', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (1, 5, TO_DATE('2023-11-01', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (5, 1, TO_DATE('2022-04-21', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (5, 6, TO_DATE('2023-04-21', 'YYYY-MM-DD'));

INSERT INTO angajat_foloseste_utilaj VALUES (7, 6, TO_DATE('2023-05-21', 'YYYY-MM-DD'));

INSERT INTO ferma_are_angajati VALUES (1, 1);

INSERT INTO ferma_are_angajati VALUES (2, 3);

INSERT INTO ferma_are_angajati VALUES (3, 5);

INSERT INTO ferma_are_angajati VALUES (4, 7);

INSERT INTO ferma_are_angajati VALUES (5, 1);

INSERT INTO ferma_are_angajati VALUES (1, 2);

INSERT INTO ferma_are_angajati VALUES (2, 3);

INSERT INTO ferma_are_angajati VALUES (3, 4);

INSERT INTO ferma_are_angajati VALUES (4, 5);

INSERT INTO ferma_are_angajati VALUES (5, 2);

INSERT INTO Depozit VALUES (1, 'Onesti', 100, 'Depozit de cereale');

INSERT INTO Depozit VALUES (2, 'Adjud', 200, 'Depozit de legume');

INSERT INTO Depozit VALUES (3, 'Ploiesti', 150, 'Depozit de fructe');

INSERT INTO Depozit VALUES (4, 'Focsani', 250, 'Depozit universal');

INSERT INTO Depozit VALUES (5, 'Bacau', 300, 'Depozit de cereale');

INSERT INTO ferma_are_produse_in_depozit VALUES (3, 2);

INSERT INTO ferma_are_produse_in_depozit VALUES (5, 2);
```

```
INSERT INTO ferma_are_produse_in_depozit VALUES (3, 3);

INSERT INTO ferma_are_produse_in_depozit VALUES (4, 4);

INSERT INTO ferma_are_produse_in_depozit VALUES (5, 5);

INSERT INTO ferma_are_produse_in_depozit VALUES (1, 2);

INSERT INTO ferma_are_produse_in_depozit VALUES (2, 3);

INSERT INTO ferma_are_produse_in_depozit VALUES (3, 4);

INSERT INTO ferma_are_produse_in_depozit VALUES (4, 5);

INSERT INTO ferma_are_produse_in_depozit VALUES (5, 1);


INSERT INTO comanda VALUES (1, TO_DATE('2021-01-23', 'YYYY-MM-DD'), 1);

INSERT INTO comanda VALUES (2, TO_DATE('2021-04-01', 'YYYY-MM-DD'), 2);

INSERT INTO comanda VALUES (3, TO_DATE('2023-03-24', 'YYYY-MM-DD'), 3);

INSERT INTO comanda VALUES (4, TO_DATE('2021-11-30', 'YYYY-MM-DD'), 4);

INSERT INTO comanda VALUES (5, TO_DATE('2023-10-06', 'YYYY-MM-DD'), 5);

INSERT INTO comanda VALUES (6, TO_DATE('2023-05-06', 'YYYY-MM-DD'), 5);


INSERT INTO comanda_contine_produs VALUES (1, 1, 6);

INSERT INTO comanda_contine_produs VALUES (2, 2, 7);

INSERT INTO comanda_contine_produs VALUES (3, 3, 1);

INSERT INTO comanda_contine_produs VALUES (4, 4, 2);

INSERT INTO comanda_contine_produs VALUES (5, 5, 3);

INSERT INTO comanda_contine_produs VALUES (1, 2, 6);

INSERT INTO comanda_contine_produs VALUES (2, 3, 7);

INSERT INTO comanda_contine_produs VALUES (3, 4, 8);

INSERT INTO comanda_contine_produs VALUES (4, 5, 2);

INSERT INTO comanda_contine_produs VALUES (5, 1, 3);

INSERT INTO factura VALUES (1, 1, 'FAC001', TO_DATE('2021-01-30', 'YYYY-MM-DD'), 123.45);

INSERT INTO factura VALUES (2, 2, 'FAC002', TO_DATE('2021-04-02', 'YYYY-MM-DD'), 678.90);

INSERT INTO factura VALUES (3, 3, 'FAC003', TO_DATE('2023-03-28', 'YYYY-MM-DD'), 234.56);

INSERT INTO factura VALUES (4, 4, 'FAC004', TO_DATE('2021-10-10', 'YYYY-MM-DD'), 789.01);

INSERT INTO factura VALUES (5, 5, 'FAC005', TO_DATE('2023-10-30', 'YYYY-MM-DD'), 345.67);
```

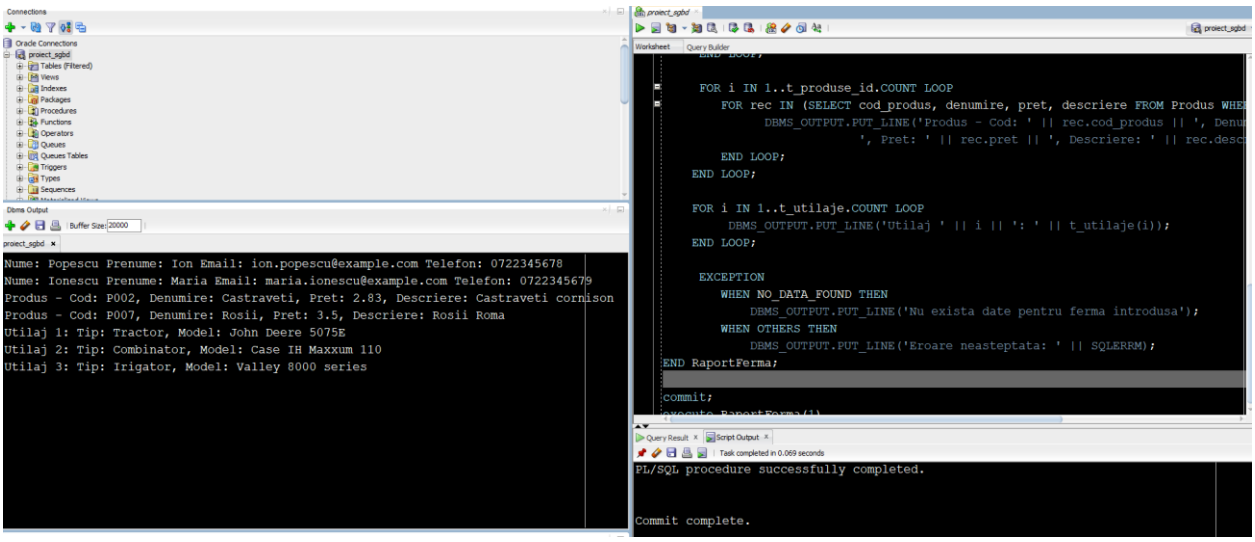
```
INSERT INTO factura VALUES (6, 5, 'FAC006', TO_DATE('2023-05-23', 'YYYY-MM-DD'), 143.47);

INSERT INTO Informatii_utilaj VALUES (1, 1500, 'Excelenta', TO_DATE('2019-04-08', 'YYYY-MM-DD'), 'John Deere 5075E');

INSERT INTO Informatii_utilaj VALUES (2, 1800, 'Buna', TO_DATE('2019-12-19', 'YYYY-MM-DD'), 'Case IH Maxxum 110');

INSERT INTO Informatii_utilaj VALUES (3, 900, 'Nou', TO_DATE('2019-03-07', 'YYYY-MM-DD'), 'Kverneland Miniair Nova');

INSERT INTO Informatii_utilaj VALUES (4, 1200, 'Uzura Medie', TO_DATE('2019-08-16', 'YYYY-MM-DD'), 'Amazone KE 3000 Super');

INSERT INTO Informatii_utilaj VALUES (5, 2000, 'Buna', TO_DATE('2019-02-22', 'YYYY-MM-DD'), 'Valley 8000 series');

INSERT INTO Informatii_utilaj VALUES (6, 1600, 'Foarte Buna', TO_DATE('2019-11-15', 'YYYY-MM-DD'), 'Massey Ferguson 6713');

INSERT INTO Informatii_utilaj VALUES (7, 1400, 'Excelenta', TO_DATE('2019-07-21', 'YYYY-MM-DD'), 'New Holland T5.105');

INSERT INTO Informatii_utilaj VALUES (8, 1550, 'Nou', TO_DATE('2019-06-30', 'YYYY-MM-DD'), 'Fendt 500 Vario');

INSERT INTO Informatii_utilaj VALUES (9, 1700, 'Uzura Redusa', TO_DATE('2019-10-12', 'YYYY-MM-DD'), 'CLAAS Arion 630');

INSERT INTO Informatii_utilaj VALUES (10, 1300, 'Buna', TO_DATE('2019-05-24', 'YYYY-MM-DD'), 'Kubota M5-091');

INSERT INTO Informatii_utilaj VALUES (11, 950, 'Uzura Medie', TO_DATE('2019-09-17', 'YYYY-MM-DD'), 'Accord Optima');

commit;
```

6.Sa se creeze o procedura RaportFerma care va primi prin id o ferma si va retine in 3 colectii diferite utilajele si modelele utilajelor prezente in ferma, angajatii care lucreaza in ferma si produsele cultivate/facute in ferma.

```
CREATE OR REPLACE PROCEDURE RaportFerma(p_id_ferma IN number) IS
   TYPE tablou_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
   t_angajati_id tablou_indexat;
   TYPE tablou_imbricat IS TABLE OF NUMBER;
   t_produse_id tablou_imbricat := tablou_imbricat();
   v_index PLS_INTEGER := 1;
   TYPE vector IS VARRAY(105) OF VARCHAR2(255);
   t_utilaje vector := vector();
   BEGIN

    FOR rec IN (SELECT id_angajat FROM ferma_are_angajati WHERE id_ferma = p_id_ferma) LOOP
      t_angajati_id(v_index) := rec.id_angajat;
      v_index := v_index + 1;
    END LOOP;
    v_index := 1;


  FOR rec IN (SELECT id_produs FROM linie_de_productie WHERE id_ferma = p_id_ferma) LOOP
   t_produse_id.EXTEND;
   t_produse_id(v_index) := rec.id_produs;
   v_index := v_index + 1;
  END LOOP;
```

```
  FOR rec IN (SELECT u.tip, iu.model
            FROM Utilaj u
            JOIN Informatii_utilaj iu ON u.id_utilaj = iu.id_utilaj
            WHERE u.id_ferma = p_id_ferma) LOOP

      t_utilaje.EXTEND;
      t_utilaje(t_utilaje.LAST) := 'Tip: ' || rec.tip || ', Model: ' || rec.model;

  END LOOP;

    FOR i IN 1..t_angajati_id.COUNT LOOP
       FOR rec IN (SELECT nume, prenume, email, telefon FROM Angajat WHERE id_angajat =
t_angajati_id(i)) LOOP
          DBMS_OUTPUT.PUT_LINE('Nume: ' || rec.nume || ' Prenume: ' || rec.prenume ||
                    ' Email: ' || rec.email || ' Telefon: ' || rec.telefon);
      END LOOP;
    END LOOP;

    FOR i IN 1..t_produse_id.COUNT LOOP
      FOR rec IN (SELECT cod_produs, denumire, pret, descriere FROM Produs WHERE id_produs =
t_produse_id(i)) LOOP
           DBMS_OUTPUT.PUT_LINE('Produs - Cod: ' || rec.cod_produs || ', Denumire: ' ||
rec.denumire ||
                    ', Pret: ' || rec.pret || ', Descriere: ' || rec.descriere);
      END LOOP;
   END LOOP;
```

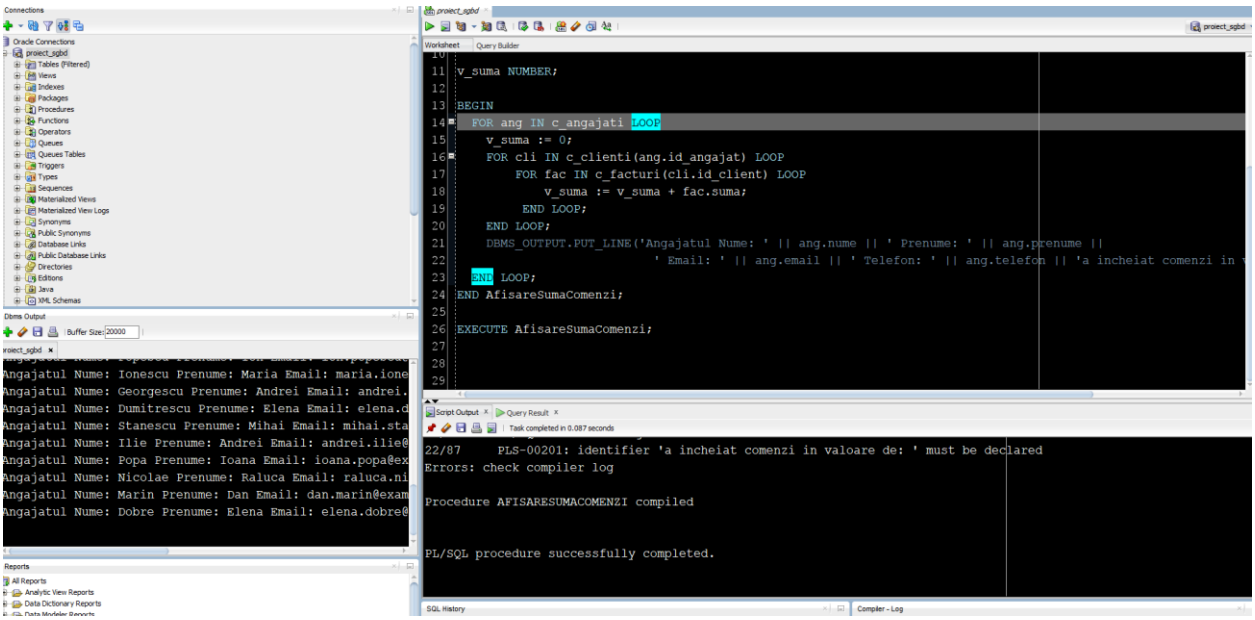
```
    FOR i IN 1..t_utilaje.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE('Utilaj ' || i || ': ' || t_utilaje(i));
    END LOOP;
END RaportFerma;

commit;
execute RaportFerma(1)
```

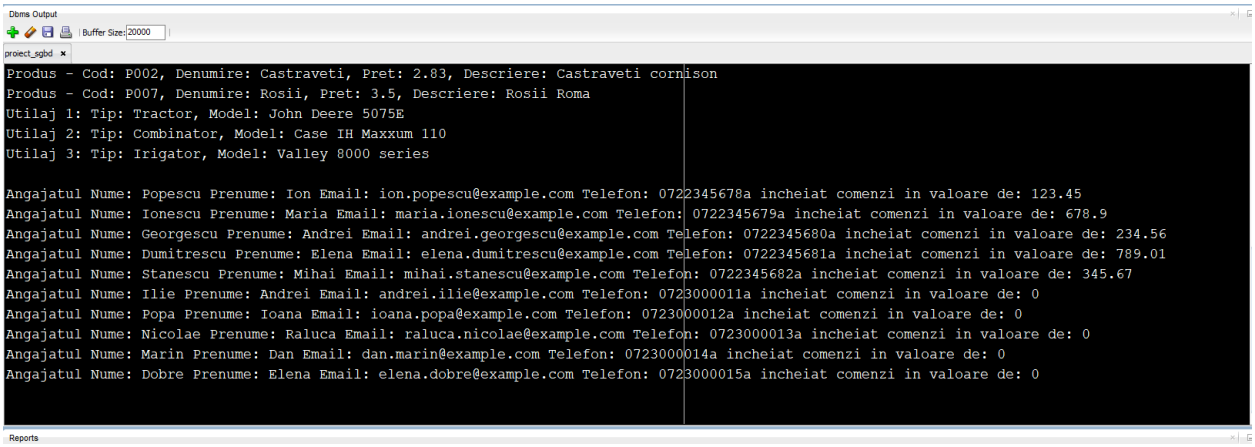


7. Calculati si afisati suma totala a comenzilor pentru fiecare angajat din companie.

```
CREATE OR REPLACE PROCEDURE AfisareSumaComenzi is
CURSOR c_angajati IS
SELECT id_angajat,nume,prenume,email,telefon FROM Angajat;

CURSOR c_clienti (p_id_angajat NUMBER) IS
SELECT id_client FROM Client WHERE id_angajat = p_id_angajat;

CURSOR c_facturi (p_id_client NUMBER) IS
```

```
SELECT suma FROM Factura WHERE id_client = p_id_client;


v_suma NUMBER;
BEGIN
  FOR ang IN c_angajati LOOP
    v_suma := 0;
    FOR cli IN c_clienti(ang.id_angajat) LOOP
        FOR fac IN c_facturi(cli.id_client) LOOP
            v_suma := v_suma + fac.suma;
        END LOOP;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Angajatul Nume: ' || ang.nume || ' Prenume: ' || ang.prenume ||
                         ' Email: ' || ang.email || ' Telefon: ' || ang.telefon || 'a incheiat comenzi in valoare
de: ' || v_suma);
  END LOOP;
END AfisareSumaComenzi;
```



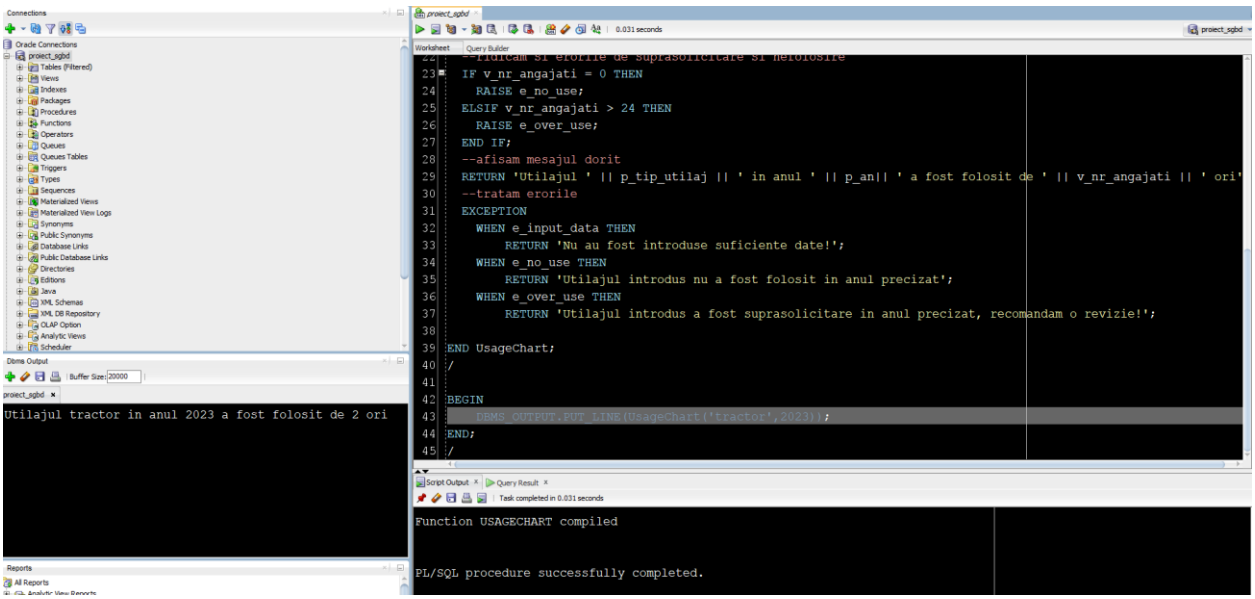


8.Sa se afiseze de cate ori a fost folosit un tip de utilaj intr-un anumit an. Tipul de utilaj si si anul va fi dat de utilizator. Daca utilajul nu a fost folosit in acel an se va afisa un mesaj corespunzator, iar daca utilajul a fost utilizat de mai mult de 24 de ori se va afisa un mesaj de suprafolosire a utilajului.

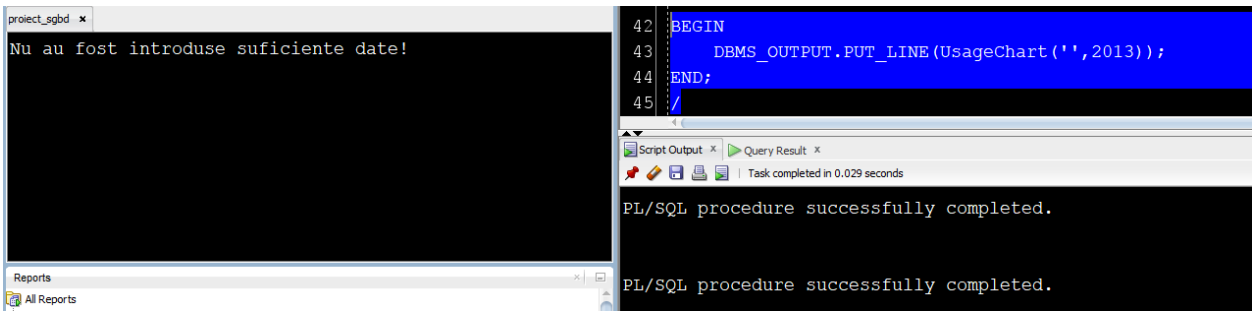
```
CREATE OR REPLACE FUNCTION UsageChart(p_tip_utilaj VARCHAR2, p_an NUMBER)

RETURN VARCHAR2 IS


  v_nr_angajati NUMBER;

  e_input_data exception;

  e_no_use exception;
```

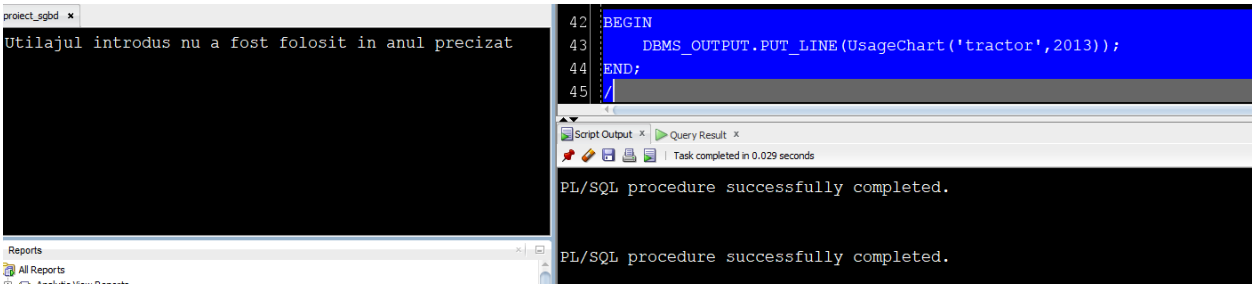
```
  e_over_use exception;

BEGIN
  --daca nu au fost introduse suficiente date ridicam eroarea de input_data
  IF p_tip_utilaj IS NULL OR p_an IS NULL THEN
    RAISE e_input_data;
  END IF;

  --calculam numarul de angajati care au folosit utilajul introdus
  SELECT COUNT(*)
  INTO v_nr_angajati
  FROM Angajat a
  JOIN angajat_foloseste_utilaj afu ON a.id_angajat = afu.id_angajat
  JOIN Utilaj u ON afu.id_utilaj = u.id_utilaj
  WHERE lower(u.tip) = lower(p_tip_utilaj)and EXTRACT(YEAR FROM TO_DATE(afu.data, 'DD-MON-
YY')) = p_an;
  --ridicam si erorile de suprasolicitare si nefolosire
  IF v_nr_angajati = 0 THEN
    RAISE e_no_use;
  ELSIF v_nr_angajati > 24 THEN
    RAISE e_over_use;
  END IF;
  --afisam mesajul dorit
  RETURN 'Utilajul ' || p_tip_utilaj || ' in anul ' || p_an|| ' a fost folosit de ' || v_nr_angajati || '
ori';
  --tratam erorile
  EXCEPTION
    WHEN e_input_data THEN
```

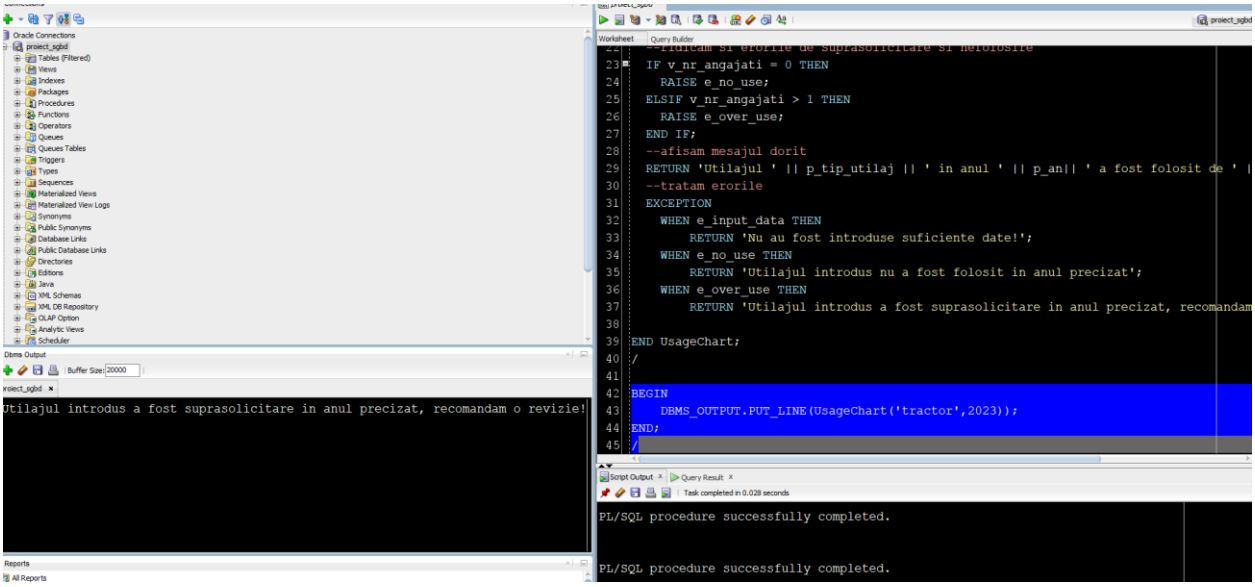
```
    RETURN 'Nu au fost introduse suficiente date!';
  WHEN e_no_use THEN
    RETURN 'Utilajul introdus nu a fost folosit in anul precizat';
  WHEN e_over_use THEN
    RETURN 'Utilajul introdus a fost suprasolicitare in anul precizat, recomandam o revizie!';

END UsageChart;
/
```



Exceptiile: (pentru exceptia de suprasolicitare, am modificat temporar conditia la mai mult de o utilizare pentru a evidentia eroarea, deoarece nu am 25 de date introduse in tabela.)







9.Pornind de la numele si prenumele unui client sa se afiseze: numele si departamentul angajatului responsabil de acest client

CREATE OR REPLACE PROCEDURE AfisareComenziClient(nume_client VARCHAR2) IS

--ne folosim de un cursor sa iteram prin detaliile cerute

CURSOR comenzi_cursor IS

SELECT c.id_comanda, f.numar_factura, a.nume || ' ' || a.prenume AS nume_angajat, d.denumire AS nume_departament

FROM Client cl

JOIN Comanda c ON cl.id_client = c.id_client

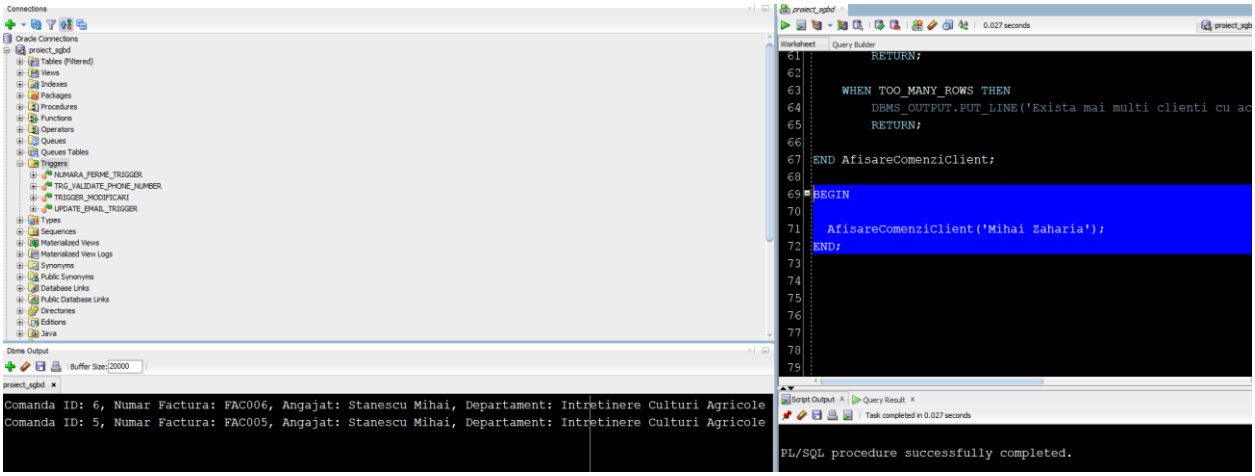
JOIN Factura f ON c.id_comanda = f.id_comanda

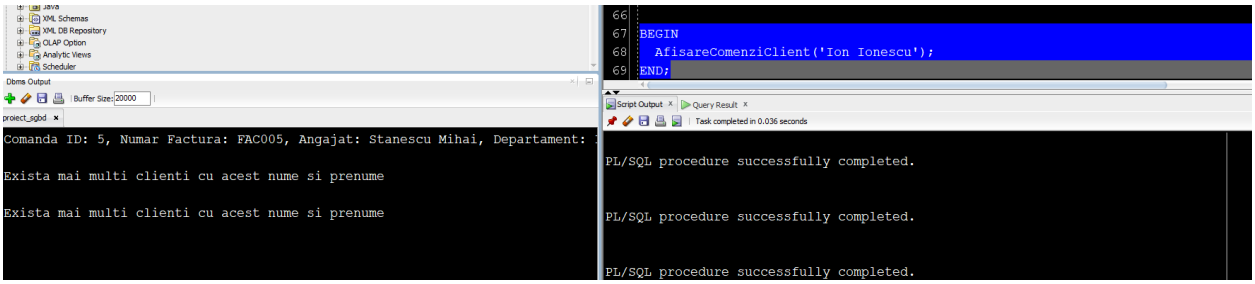
JOIN Angajat a ON cl.id_angajat = a.id_angajat

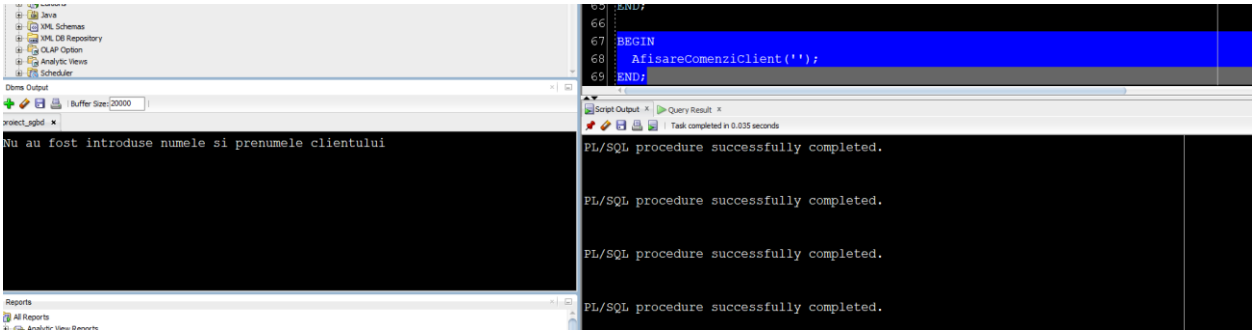
```
   JOIN Departament d ON a.id_departament = d.id_departament
   WHERE lower(cl.nume_pr) = lower(nume_client);



  v_id_comanda Comanda.id_comanda%TYPE;
  v_numar_factura Factura.numar_factura%TYPE;
  v_nume_angajat VARCHAR2(100);
  v_nume_departament Departament.denumire%TYPE;
  v_nr_clienti INT;

  e_input EXCEPTION;
BEGIN

  --verificam daca au fost introduse date
  IF nume_client IS NULL THEN
    RAISE e_input;
  END IF;

  --calculam numarul de clienti cu acest nume si prenume pentru a ne asigura ca exista doar un
client
  SELECT COUNT(*)
  INTO v_nr_clienti
  FROM Client
  WHERE lower(nume_pr) = lower(nume_client);

  --daca nu exista un client cu acest nume sau daca exista mai multi vom ridica erorile
corespondente
  IF v_nr_clienti = 0 THEN
```

```
    RAISE NO_DATA_FOUND;
  ELSIF v_nr_clienti > 1 THEN
    RAISE TOO_MANY_ROWS;
  END IF;

  --deschidem cursorul si iteram prin toate datele, pentru a afisa toate comenzile si facturile
clientului
  OPEN comenzi_cursor;
  LOOP
    FETCH comenzi_cursor INTO v_id_comanda, v_numar_factura, v_nume_angajat,
v_nume_departament;
    EXIT WHEN comenzi_cursor%NOTFOUND;



    DBMS_OUTPUT.PUT_LINE('Comanda ID: ' || v_id_comanda ||
                         ', Numar Factura: ' || v_numar_factura ||
                         ', Angajat: ' || v_nume_angajat ||
                         ', Departament: ' || v_nume_departament);
  END LOOP;
  CLOSE comenzi_cursor;
  --tratam exceptiile
  EXCEPTION
    WHEN e_input THEN
        DBMS_OUTPUT.PUT_LINE('Nu au fost introduse numele si prenumele clientului');
        RETURN;
    WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Nu exista date pentru clientul specificat');
       RETURN;
```

```
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu acest nume si prenume');
        RETURN;

END AfisareComenziClient;
```



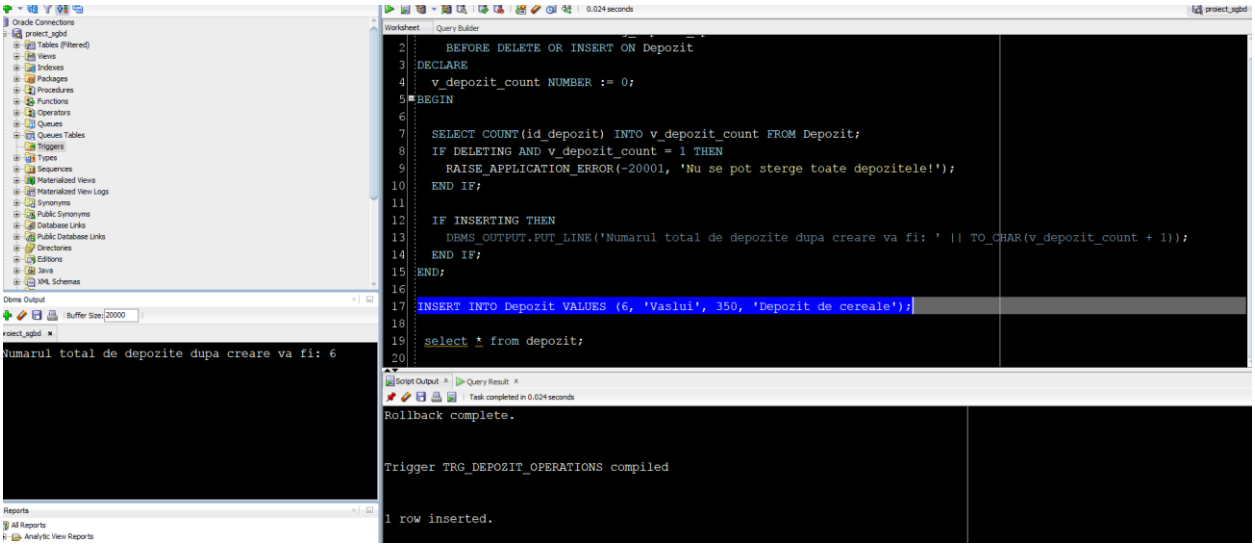




10.

```
CREATE OR REPLACE TRIGGER trg_depozit_operations
```

```
  BEFORE DELETE OR INSERT ON Depozit
DECLARE
 v_depozit_count NUMBER := 0;
BEGIN

 SELECT COUNT(id_depozit) INTO v_depozit_count FROM Depozit;
 IF DELETING AND v_depozit_count = 1 THEN
  RAISE_APPLICATION_ERROR(-20001, 'Nu se pot sterge toate depozitele!');
 END IF;

 IF INSERTING THEN
  DBMS_OUTPUT.PUT_LINE('Numarul total de depozite dupa creare va fi: ' ||
TO_CHAR(v_depozit_count + 1));
 END IF;
END;
```



```
19
20 delete from depozit;
```

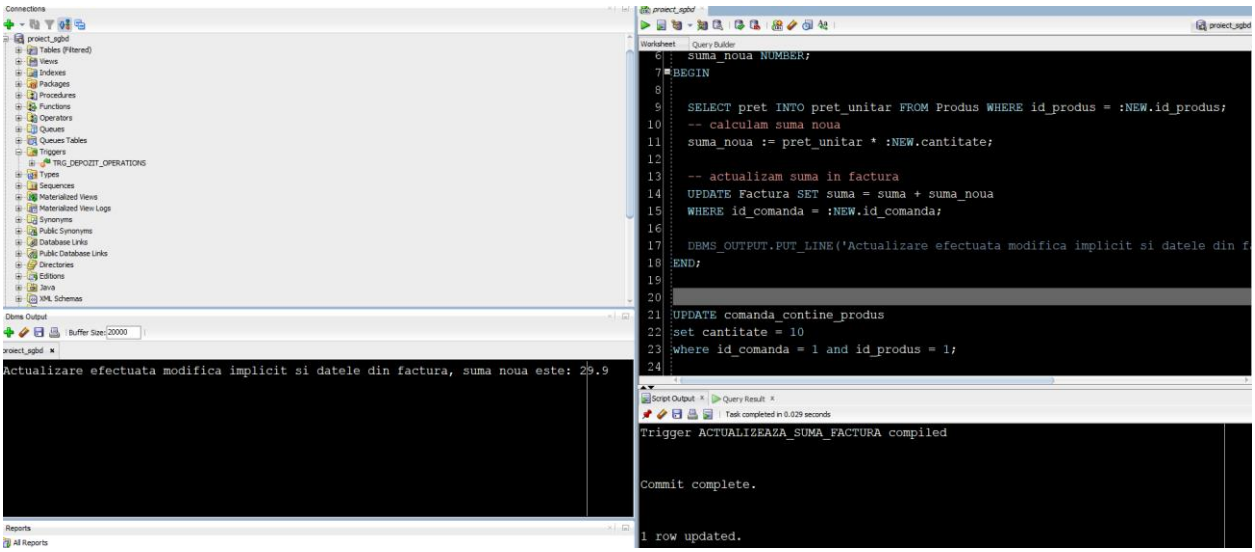
Script Output × | Query Result ×

Task completed in 0.048 seconds

```
rror starting at line : 20 in command -
elete from depozit
rror at Command Line : 20 Column : 13
rror report -
QL Error: ORA-20001: Nu se pot sterge toate depozitele!
RA-06512: at "UTILIZATOR.TRG_DEPOZIT_OPERATIONS", line 7
RA-04088: error during execution of trigger 'UTILIZATOR.TRG_DEPOZIT_OPERATIONS'
```

11. Se va crea un trigger LMD de linie care va actualiza automat suma din factura la modificarea cantitatii din comanda.

```
CREATE OR REPLACE TRIGGER actualizeaza_suma_factura

AFTER INSERT OR UPDATE  ON Comanda_Contine_Produs

FOR EACH ROW

DECLARE

  pret_unitar NUMBER;

  suma_noua NUMBER;

BEGIN


  SELECT pret INTO pret_unitar FROM Produs WHERE id_produs = :NEW.id_produs;

  -- calculam suma noua

  suma_noua := pret_unitar * :NEW.cantitate;


  -- actualizam suma in factura

  UPDATE Factura SET suma = suma + suma_noua

  WHERE id_comanda = :NEW.id_comanda;


  DBMS_OUTPUT.PUT_LINE('Actualizare efectuata modifica implicit si datele din factura, suma noua este: ' || suma_noua);

END;
```



12.Se va crea un trigger LDD care va adauga intr-o tabela auxiliara detalii despre modificarile facute pe baza de date, daca user-ul curent este „UTILIZATOR”.

```
CREATE TABLE audit_firma (
  id_modificare NUMBER PRIMARY KEY,
  tip_modificare VARCHAR2(50),
  tip_obiect VARCHAR2(50),
  nume_obiect VARCHAR2(50),
  data_modficare DATE
);
CREATE SEQUENCE audit_firma_seq START WITH 1 INCREMENT BY 1;
```

```
CREATE OR REPLACE TRIGGER audit_firma_agricultura
AFTER CREATE OR ALTER OR DROP ON SCHEMA
BEGIN
  -- verificarea pentru utilizator
  IF lower(SYS_CONTEXT('USERENV', 'SESSION_USER')) != 'utilizator' THEN
    --ridicam eroare in caz contrar
```

RAISE_APPLICATION_ERROR(-20001, 'Nu aveti autorizatie sa modificati baza de date!!');

ELSE

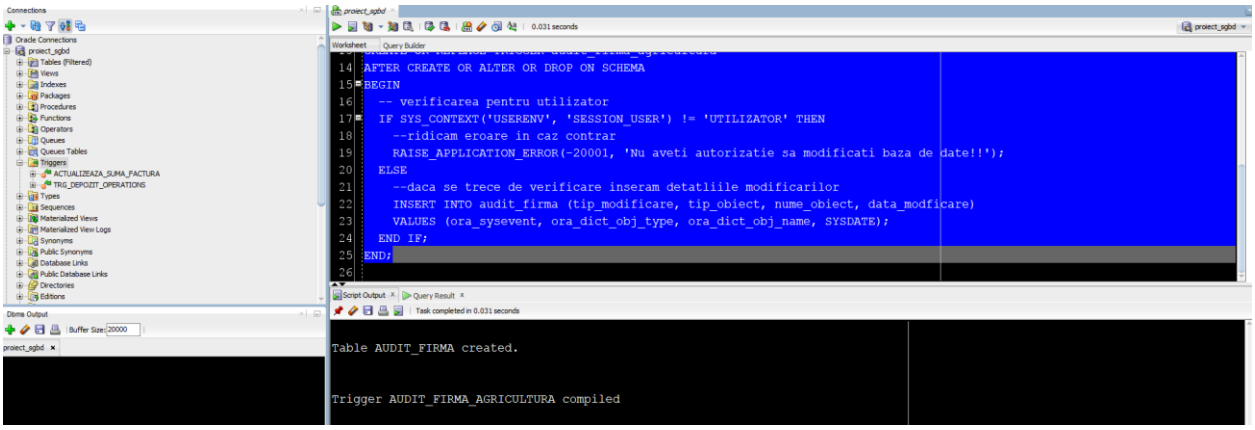
--daca se trece de verificare inseram detatliile modificarilor

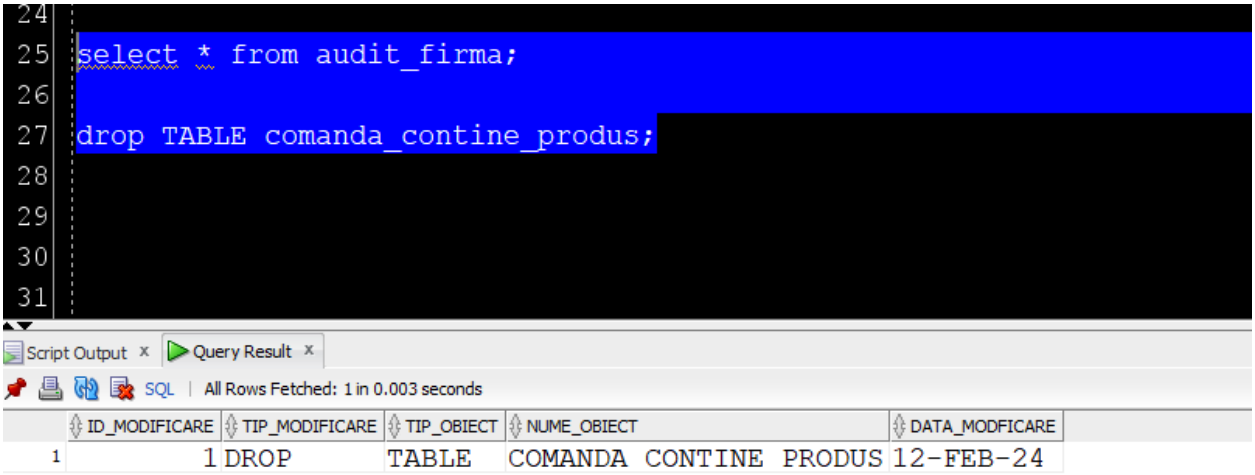
INSERT INTO audit_firma (id_modificare,tip_modificare, tip_obiect, nume_obiect, data_modficare)

VALUES (audit_firma_seq.NEXTVAL,ora_sysevent, ora_dict_obj_type, ora_dict_obj_name, SYSDATE);

END IF;

END;





13.Sa se defineasca un pachet cu toate obiectele definite in cadrul proiectului:

CREATE OR REPLACE PACKAGE pachet_proiect AS

--6

PROCEDURE RaportFerma(p_id_ferma IN number);

```
  --7
  PROCEDURE AfisareSumaComenzi ;
  --8
  FUNCTION UsageChart(p_tip_utilaj VARCHAR2, p_an NUMBER) RETURN VARCHAR2 ;
  --9
  PROCEDURE AfisareComenziClient(nume_client VARCHAR2);
END pachet_proiect;
/
CREATE OR REPLACE PACKAGE BODY pachet_proiect AS
  --6
  PROCEDURE RaportFerma(p_id_ferma IN number) IS
    TYPE tablou_indexat IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    t_angajati_id tablou_indexat;
    TYPE tablou_imbricat IS TABLE OF NUMBER;
    t_produse_id tablou_imbricat := tablou_imbricat();
    v_index PLS_INTEGER := 1;
    TYPE vector IS VARRAY(105) OF VARCHAR2(255);
    t_utilaje vector := vector();
    BEGIN

     FOR rec IN (SELECT id_angajat FROM ferma_are_angajati WHERE id_ferma = p_id_ferma) LOOP
       t_angajati_id(v_index) := rec.id_angajat;
       v_index := v_index + 1;
     END LOOP;
     v_index := 1;
```

```
  FOR rec IN (SELECT id_produs FROM linie_de_productie WHERE id_ferma = p_id_ferma) LOOP
    t_produse_id.EXTEND;
    t_produse_id(v_index) := rec.id_produs;
    v_index := v_index + 1;
  END LOOP;

  FOR rec IN (SELECT u.tip, iu.model
              FROM Utilaj u
              JOIN Informatii_utilaj iu ON u.id_utilaj = iu.id_utilaj
              WHERE u.id_ferma = p_id_ferma) LOOP

      t_utilaje.EXTEND;
      t_utilaje(t_utilaje.LAST) := 'Tip: ' || rec.tip || ', Model: ' || rec.model;

  END LOOP;

    FOR i IN 1..t_angajati_id.COUNT LOOP
       FOR rec IN (SELECT nume, prenume, email, telefon FROM Angajat WHERE id_angajat =
t_angajati_id(i)) LOOP
          DBMS_OUTPUT.PUT_LINE('Nume: ' || rec.nume || ' Prenume: ' || rec.prenume ||
                    ' Email: ' || rec.email || ' Telefon: ' || rec.telefon);
      END LOOP;
    END LOOP;

    FOR i IN 1..t_produse_id.COUNT LOOP
      FOR rec IN (SELECT cod_produs, denumire, pret, descriere FROM Produs WHERE id_produs =
t_produse_id(i)) LOOP
```

```
            DBMS_OUTPUT.PUT_LINE('Produs - Cod: ' || rec.cod_produs || ', Denumire: ' ||
rec.denumire ||
                        ', Pret: ' || rec.pret || ', Descriere: ' || rec.descriere);
        END LOOP;
    END LOOP;

    FOR i IN 1..t_utilaje.COUNT LOOP
        DBMS_OUTPUT.PUT_LINE('Utilaj ' || i || ': ' || t_utilaje(i));
    END LOOP;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista date pentru ferma introdusa');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Eroare neasteptata: ' || SQLERRM);
    END RaportFerma;

  --7
  PROCEDURE AfisareSumaComenzi is
CURSOR c_angajati IS
SELECT id_angajat,nume,prenume,email,telefon FROM Angajat;

CURSOR c_clienti (p_id_angajat NUMBER) IS
SELECT id_client FROM Client WHERE id_angajat = p_id_angajat;

CURSOR c_facturi (p_id_client NUMBER) IS
SELECT suma FROM Factura WHERE id_client = p_id_client;
```

```
v_suma NUMBER;
BEGIN
  FOR ang IN c_angajati LOOP
    v_suma := 0;
    FOR cli IN c_clienti(ang.id_angajat) LOOP
        FOR fac IN c_facturi(cli.id_client) LOOP
            v_suma := v_suma + fac.suma;
        END LOOP;
    END LOOP;
    DBMS_OUTPUT.PUT_LINE('Angajatul Nume: ' || ang.nume || ' Prenume: ' || ang.prenume ||
                         ' Email: ' || ang.email || ' Telefon: ' || ang.telefon || 'a incheiat comenzi in valoare
de: ' || v_suma);
  END LOOP;
END AfisareSumaComenzi;


  --8
  FUNCTION UsageChart(p_tip_utilaj VARCHAR2, p_an NUMBER)
RETURN VARCHAR2 IS


  v_nr_angajati NUMBER;
  e_input_data exception;
  e_no_use exception;
  e_over_use exception;


BEGIN
  --daca nu au fost introduse suficiente date ridicam eroarea de input_data
```

```
  IF p_tip_utilaj IS NULL OR p_an IS NULL THEN
    RAISE e_input_data;
  END IF;

  --calculam numarul de angajati care au folosit utilajul introdus
  SELECT COUNT(*)
  INTO v_nr_angajati
  FROM Angajat a
  JOIN angajat_foloseste_utilaj afu ON a.id_angajat = afu.id_angajat
  JOIN Utilaj u ON afu.id_utilaj = u.id_utilaj
  WHERE lower(u.tip) = lower(p_tip_utilaj)and EXTRACT(YEAR FROM TO_DATE(afu.data, 'DD-MON-
YY')) = p_an;
  --ridicam si erorile de suprasolicitare si nefolosire
  IF v_nr_angajati = 0 THEN
    RAISE e_no_use;
  ELSIF v_nr_angajati > 24 THEN
    RAISE e_over_use;
  END IF;
  --afisam mesajul dorit
  RETURN 'Utilajul ' || p_tip_utilaj || ' in anul ' || p_an|| ' a fost folosit de ' || v_nr_angajati || '
ori';
  --tratam erorile
  EXCEPTION
    WHEN e_input_data THEN
        RETURN 'Nu au fost introduse suficiente date!';
    WHEN e_no_use THEN
        RETURN 'Utilajul introdus nu a fost folosit in anul precizat';
    WHEN e_over_use THEN
```

```
      RETURN 'Utilajul introdus a fost suprasolicitare in anul precizat, recomandam o revizie!';
  END UsageChart;
  --9
  PROCEDURE AfisareComenziClient(nume_client VARCHAR2) IS

  --ne folosim de un cursor sa iteram prin detaliile cerute
  CURSOR comenzi_cursor IS
    SELECT c.id_comanda, f.numar_factura, a.nume || ' ' || a.prenume AS nume_angajat,
d.denumire AS nume_departament
    FROM Client cl
    JOIN Comanda c ON cl.id_client = c.id_client
    JOIN Factura f ON c.id_comanda = f.id_comanda
    JOIN Angajat a ON cl.id_angajat = a.id_angajat
    JOIN Departament d ON a.id_departament = d.id_departament
    WHERE lower(cl.nume_pr) = lower(nume_client);


  v_id_comanda Comanda.id_comanda%TYPE;
  v_numar_factura Factura.numar_factura%TYPE;
  v_nume_angajat VARCHAR2(100);
  v_nume_departament Departament.denumire%TYPE;
  v_nr_clienti INT;

  e_input EXCEPTION;
BEGIN

  --verificam daca au fost introduse date
```

```
  IF nume_client IS NULL THEN
    RAISE e_input;
  END IF;

  --calculam numarul de clienti cu acest nume si prenume pentru a ne asigura ca exista doar un
client
  SELECT COUNT(*)
  INTO v_nr_clienti
  FROM Client
  WHERE lower(nume_pr) = lower(nume_client);

  --daca nu exista un client cu acest nume sau daca exista mai multi vom ridica erorile
corespondente
  IF v_nr_clienti = 0 THEN
    RAISE NO_DATA_FOUND;
  ELSIF v_nr_clienti > 1 THEN
    RAISE TOO_MANY_ROWS;
  END IF;

  --deschidem cursorul si iteram prin toate datele, pentru a afisa toate comenzile si facturile
clientului
  OPEN comenzi_cursor;
  LOOP
    FETCH comenzi_cursor INTO v_id_comanda, v_numar_factura, v_nume_angajat,
v_nume_departament;
    EXIT WHEN comenzi_cursor%NOTFOUND;
```

```
    DBMS_OUTPUT.PUT_LINE('Comanda ID: ' || v_id_comanda ||
                        ', Numar Factura: ' || v_numar_factura ||
                        ', Angajat: ' || v_nume_angajat ||
                        ', Departament: ' || v_nume_departament);
  END LOOP;
  CLOSE comenzi_cursor;
  --tratam exceptiile
  EXCEPTION
    WHEN e_input THEN
        DBMS_OUTPUT.PUT_LINE('Nu au fost introduse numele si prenumele clientului');
        RETURN;
    WHEN NO_DATA_FOUND THEN
       DBMS_OUTPUT.PUT_LINE('Nu exista date pentru clientul specificat');
       RETURN;

    WHEN TOO_MANY_ROWS THEN
       DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu acest nume si prenume');
       RETURN;

END AfisareComenziClient;

END pachet_proiect;
```

proiect_sgbd

Worksheet | Query Builder

```
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista date pentru clientul specificat');
        RETURN;

    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multi clienti cu acest nume si prenume');
        RETURN;

END AfisareComenziClient;

END pachet_proiect;
```

Script Output | Task completed in 0.071 seconds

```
Package PACHET_PROIECT compiled


Package Body PACHET_PROIECT compiled
```