Gestiunea unui magazine online

Horga Daria

241

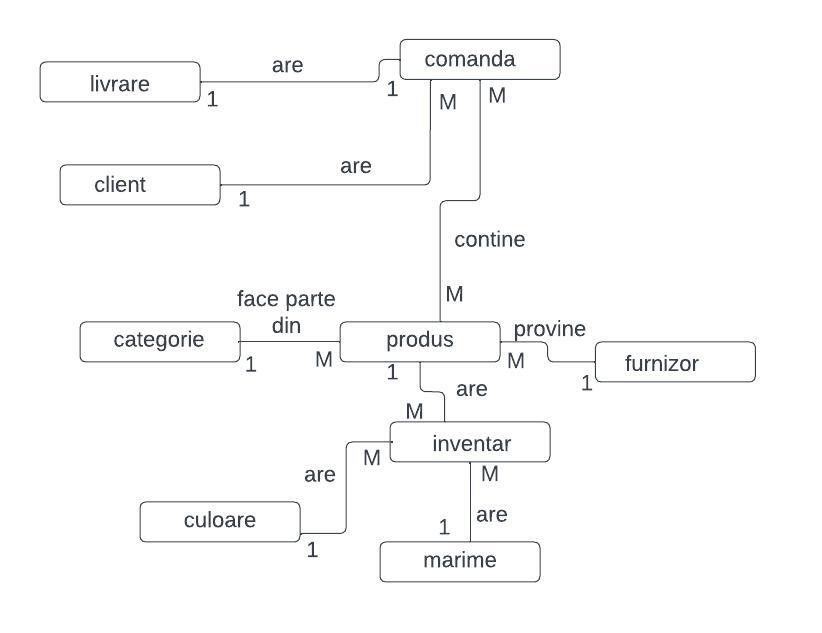
Universitatea din Bucuresti

1. Prezentați pe scurt baza de date

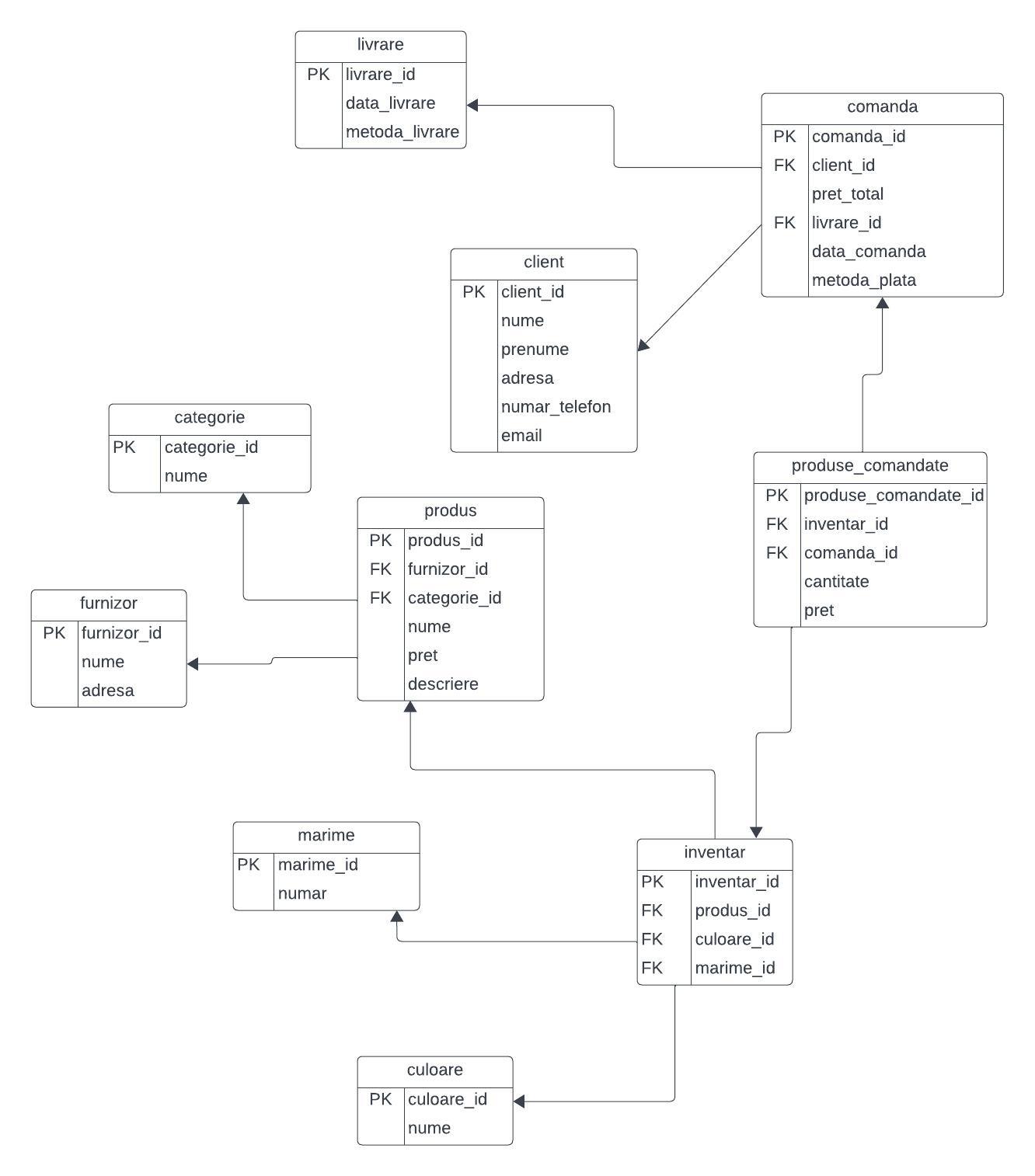
Baza de date oferă o structură organizată pentru gestionarea unei afaceri de vânzare de produse, cu informații referitoare la furnizori, categorii de produse, clienți, livrare, comenzi și stocuri de produse. Scopul ei este să permită stocarea eficientă și accesarea informațiilor despre produse, clienți și tranzacții, facilitând astfel managementul afacerii și îmbunătățind procesele de vânzare, livrare și inventar. Prin relațiile dintre tabele, baza de date susține monitorizarea stocurilor, gestionarea comenzilor și urmărirea istoricului tranzacțiilor, contribuind la eficiența și transparența operațiunilor comerciale.

Fiecare produs este înregistrat în baza de date cu informații detaliate, cum ar fi numele, prețul, categoria, furnizorul și o scurtă descriere a acestuia. Pentru a ține evidența stocurilor și pentru a oferi clienților opțiuni variate, fiecare produs are asociate mai multe culori și mărimi. Clienții sunt înregistrați cu datele personale, iar vânzările sunt înregistrate pentru fiecare comandă efectuată, având disponibile și informații despre livrare.

2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română



3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.



4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, definind toate constrângerile de integritate necesare

CREATE TABLE furnizor (

furnizor\_id NUMBER(10) PRIMARY KEY,

nume VARCHAR(100) NOT NULL,

adresa VARCHAR(300)

);

CREATE TABLE categorie (

categorie\_id NUMBER(10) PRIMARY KEY,

nume VARCHAR(100) NOT NULL

);

CREATE TABLE livrare (

livrare\_id NUMBER(10) PRIMARY KEY,

data\_livrare DATE,

metoda\_livrare VARCHAR(100)

);

CREATE TABLE client (

client\_id NUMBER(10) PRIMARY KEY,

nume VARCHAR(100) NOT NULL,

prenume VARCHAR(100) NOT NULL,

adresa VARCHAR(300) NOT NULL,

numar\_telefon NUMBER(10) NOT NULL,

email VARCHAR(100) NOT NULL

);

CREATE TABLE comanda (

comanda\_id NUMBER(10) PRIMARY KEY,

client\_id NUMBER(10),

pret\_total NUMBER(10),

livrare\_id NUMBER(10),

data\_comanda DATE,

metoda\_plata VARCHAR(100),

FOREIGN KEY (client\_id) REFERENCES client(client\_id),

FOREIGN KEY (livrare\_id) REFERENCES livrare(livrare\_id)

);

CREATE TABLE produs (

produs\_id NUMBER(10) PRIMARY KEY,

furnizor\_id NUMBER(10),

categorie\_id NUMBER(10),

nume VARCHAR(100) NOT NULL,

pret NUMBER(10) NOT NULL,

descriere VARCHAR(500),

FOREIGN KEY (furnizor\_id) REFERENCES furnizor(furnizor\_id),

FOREIGN KEY (categorie\_id) REFERENCES categorie(categorie\_id)

);

CREATE TABLE marime (

marime\_id NUMBER(10) PRIMARY KEY,

numar NUMBER(10) NOT NULL

);

CREATE TABLE culoare (

culoare\_id NUMBER(10) PRIMARY KEY,

nume varchar(20) NOT NULL

);

CREATE TABLE inventar (

inventar\_id NUMBER(10) PRIMARY KEY,

produs\_id NUMBER(10),

culoare\_id NUMBER(10),

marime\_id NUMBER(10),

FOREIGN KEY (produs\_id) REFERENCES produs(produs\_id),

FOREIGN KEY (culoare\_id) REFERENCES culoare(culoare\_id),

FOREIGN KEY (marime\_id) REFERENCES marime(marime\_id)

);

CREATE TABLE produse\_comandate (

produse\_comandate\_id NUMBER(10) PRIMARY KEY,

inventar\_id NUMBER(10),

comanda\_id NUMBER(10),

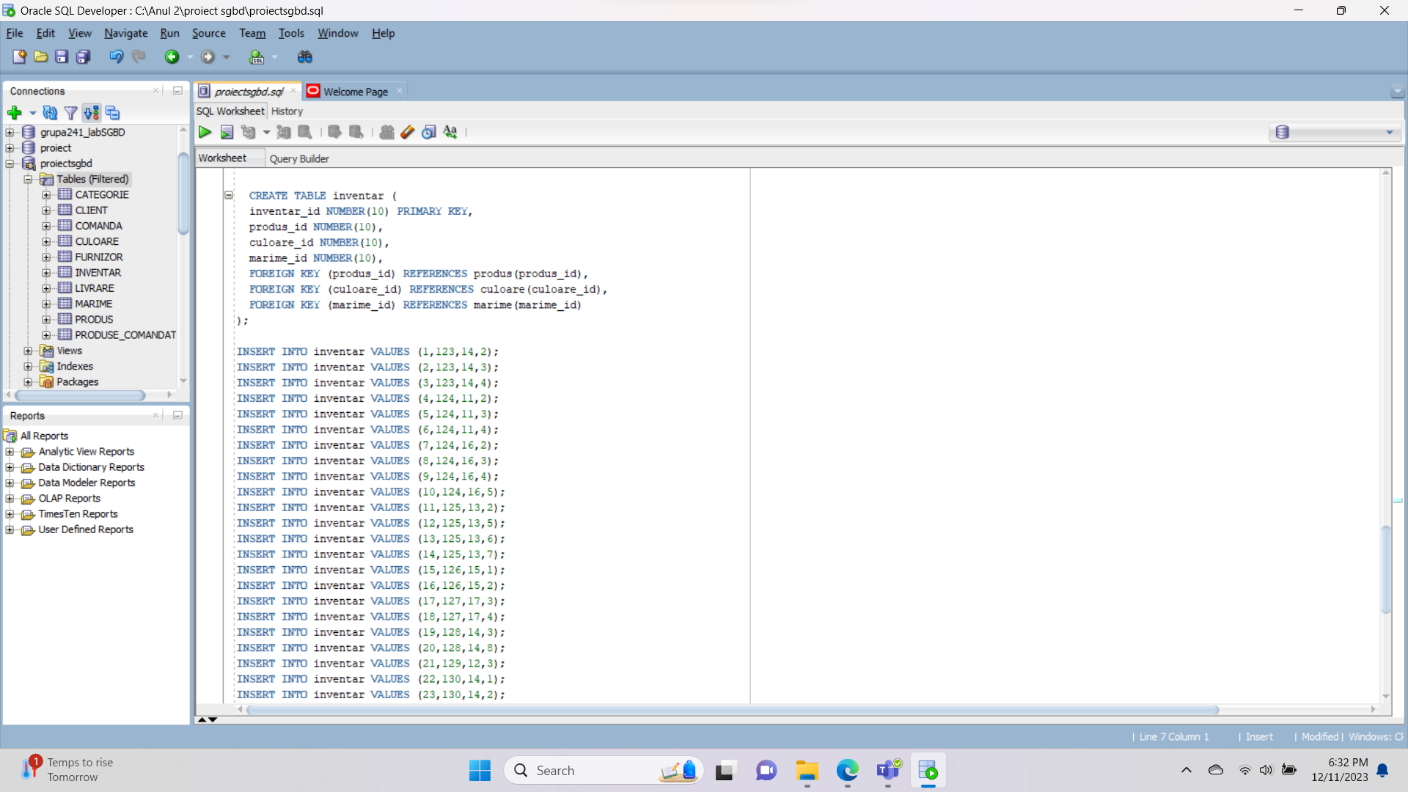
cantitate NUMBER(10),

pret NUMBER(10),

FOREIGN KEY (inventar\_id) REFERENCES inventar(inventar\_id),

FOREIGN KEY (comanda\_id) REFERENCES comanda(comanda\_id)

);



5. Adăugați informații coerente în tabelele create.

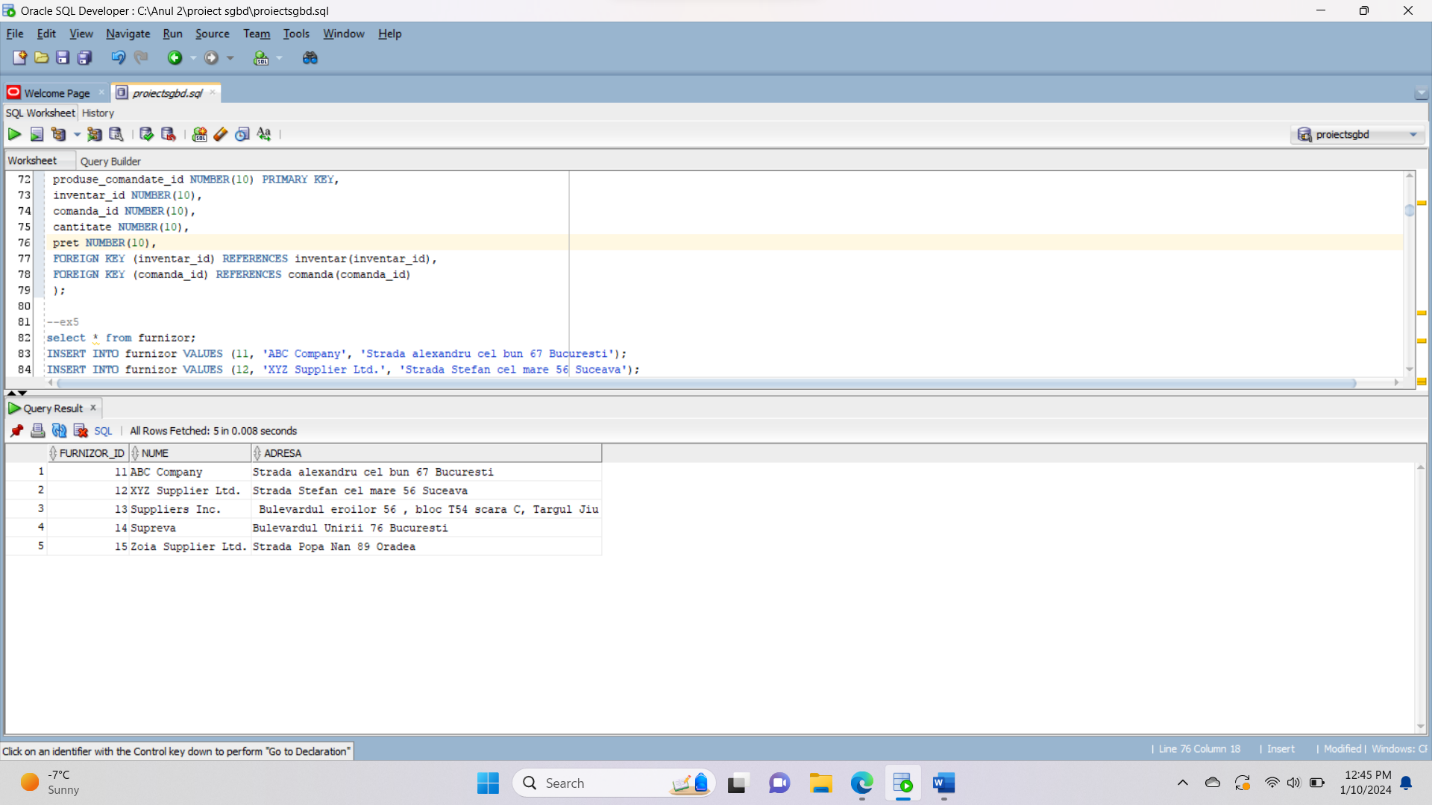
INSERT INTO furnizor VALUES (11, 'ABC Company', 'Strada alexandru cel bun 67 Bucuresti');

INSERT INTO furnizor VALUES (12, 'XYZ Supplier Ltd.', 'Strada Stefan cel mare 56 Suceava');

INSERT INTO furnizor VALUES (13, 'Suppliers Inc.', ' Bulevardul eroilor 56 , bloc T54 scara C, Targul Jiu');

INSERT INTO furnizor VALUES (14, 'Supreva', 'Bulevardul Unirii 76 Bucuresti');

INSERT INTO furnizor VALUES (15, 'Zoia Supplier Ltd.', 'Strada Popa Nan 89 Oradea');



INSERT INTO categorie

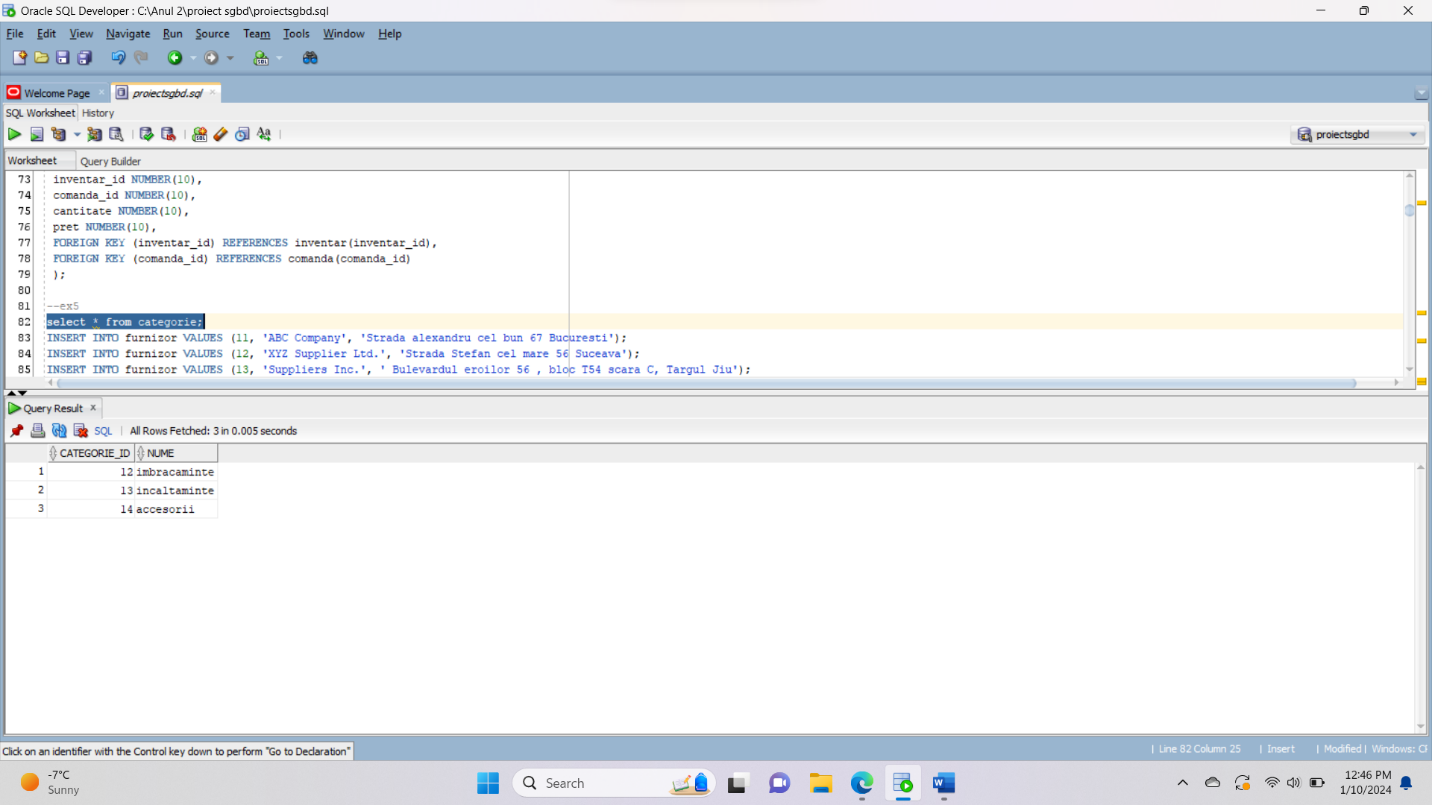
VALUES(12,'imbracaminte');

INSERT INTO categorie

VALUES(13,'incaltaminte');

INSERT INTO categorie

VALUES(14,'accesorii');



INSERT INTO client

VALUES(21,'Ionescu','Raluca','Alexandru Ioan Cuza 56 Bucuresti',0722222222,'ionescuraluca@yahoo.com');

INSERT INTO client

VALUES(22,'Popescu','Alexandru','Independentei 22 bloc T63 scara B Bucuresti',07222345682,'alex\_popescu@yahoo.com');

INSERT INTO client

VALUES(23,'Anghel','Andreea','Alexandru Osobescu Ploiesti bloc 17A',0771234509,'anghelandreea234@gmail.com');

INSERT INTO client

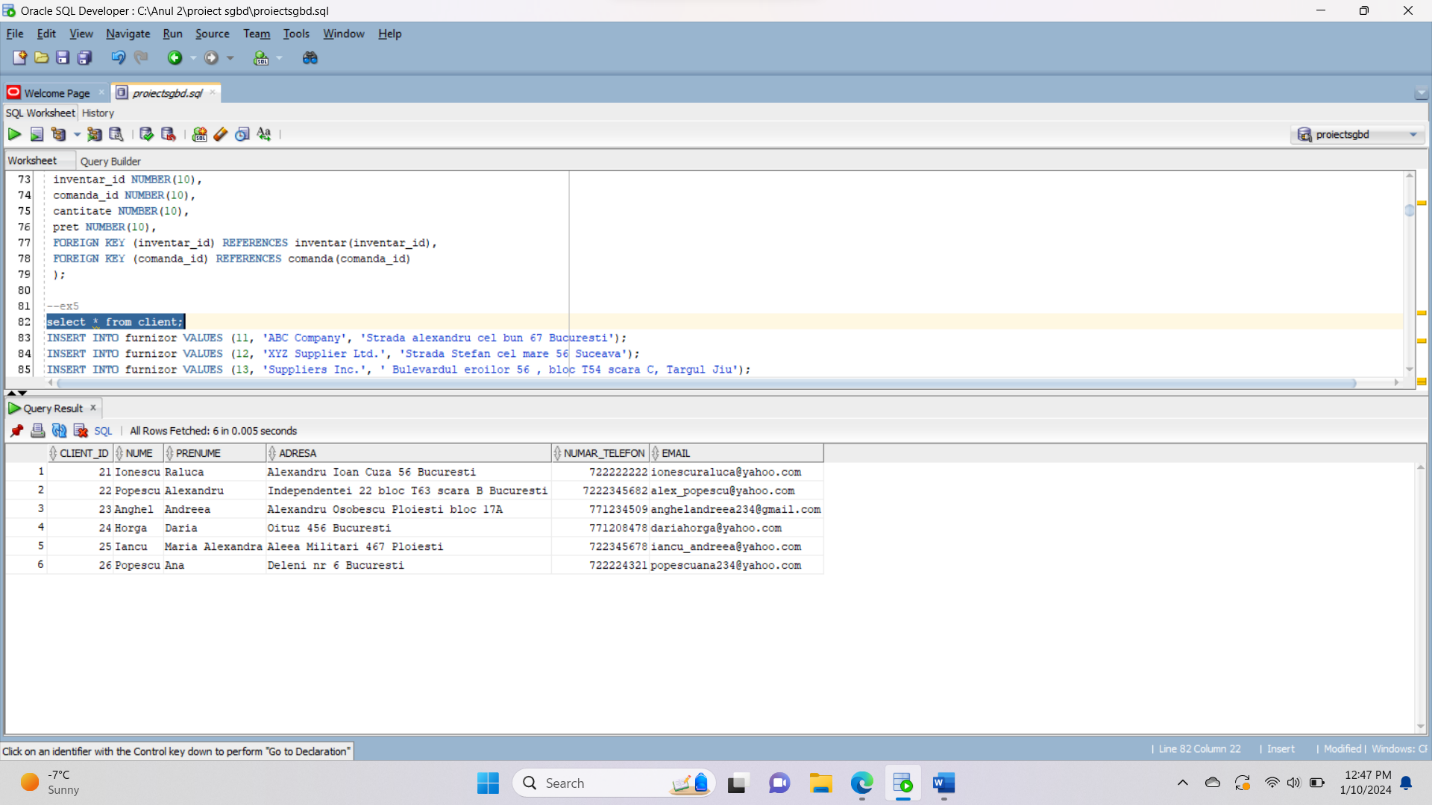
VALUES(24,'Horga','Daria','Oituz 456 Bucuresti',0771208478,'dariahorga@yahoo.com');

INSERT INTO client

VALUES(25,'Iancu', 'Maria Alexandra','Aleea Militari 467 Ploiesti',0722345678,'iancu\_andreea@yahoo.com');

INSERT INTO client

VALUES(26,'Popescu', 'Ana','Deleni nr 6 Bucuresti',0722224321,'popescuana234@yahoo.com');



INSERT INTO livrare VALUES (1,TO\_DATE('12-09-2023','DD-MM-YYYY'),'curier');

INSERT INTO livrare VALUES (2,TO\_DATE('12-10-2023','DD-MM-YYYY'),'easybox');

INSERT INTO livrare VALUES (3,TO\_DATE('16-11-2023','DD-MM-YYYY'),'curier');

INSERT INTO livrare VALUES (4,TO\_DATE('23-11-2023','DD-MM-YYYY'),'curier');

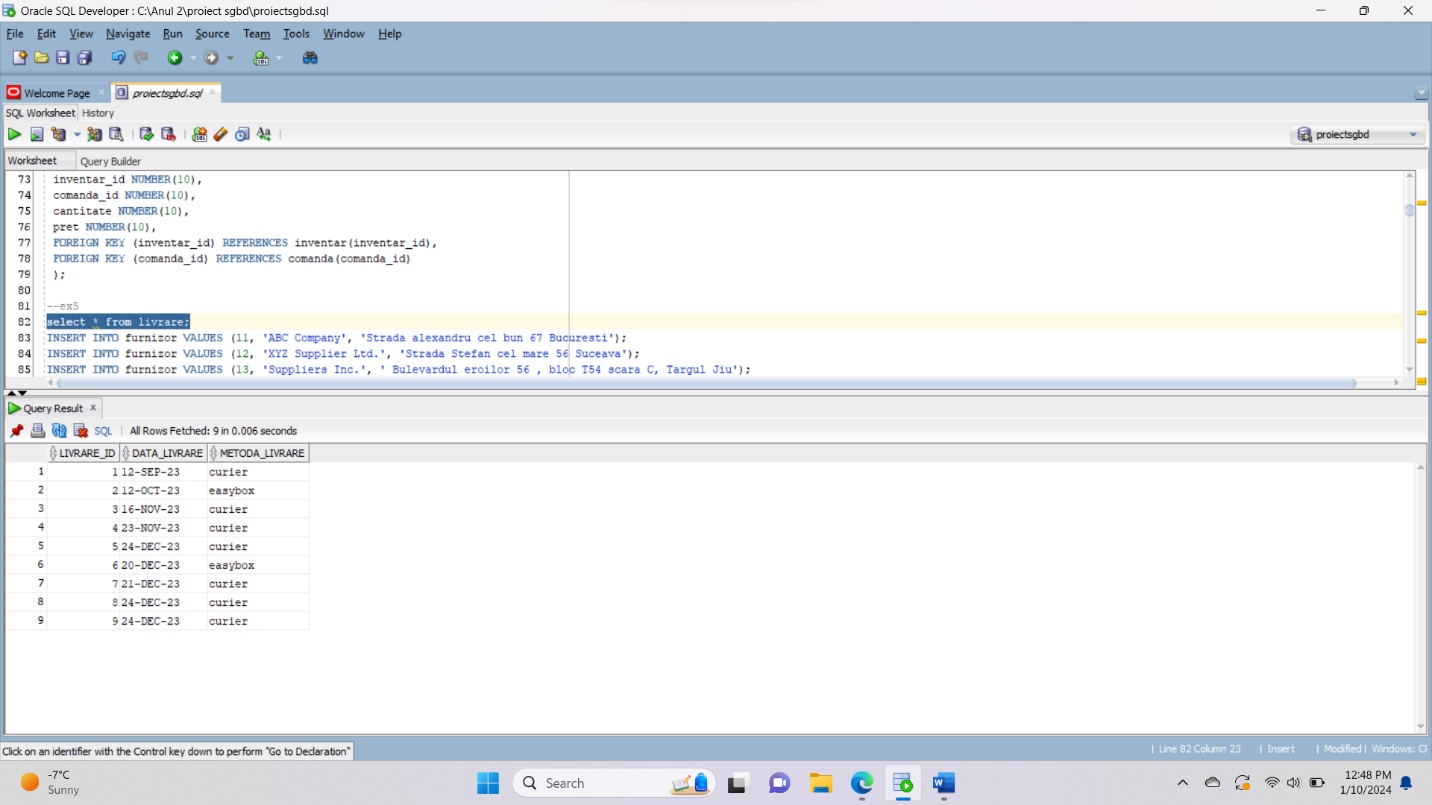
INSERT INTO livrare VALUES (5,TO\_DATE('24-12-2023','DD-MM-YYYY'),'curier');

INSERT INTO livrare VALUES (6,TO\_DATE('20-12-2023','DD-MM-YYYY'),'easybox');

INSERT INTO livrare VALUES (7,TO\_DATE('21-12-2023','DD-MM-YYYY'),'curier');

INSERT INTO livrare VALUES (8,TO\_DATE('24-12-2023','DD-MM-YYYY'),'curier');

INSERT INTO livrare VALUES (9,TO\_DATE('24-12-2023','DD-MM-YYYY'),'curier');



INSERT INTO comanda VALUES (1, 21,260 ,1,TO\_DATE('10-09-2023','DD-MM-YYYY'),'numerar');

INSERT INTO comanda VALUES (2, 22,110 ,2,TO\_DATE('10-10-2023','DD-MM-YYYY'),'numerar');

INSERT INTO comanda VALUES (3, 23,150 ,3,TO\_DATE('14-11-2023','DD-MM-YYYY'),'card');

INSERT INTO comanda VALUES (4, 24,240 ,4,TO\_DATE('20-11-2023','DD-MM-YYYY'),'numerar');

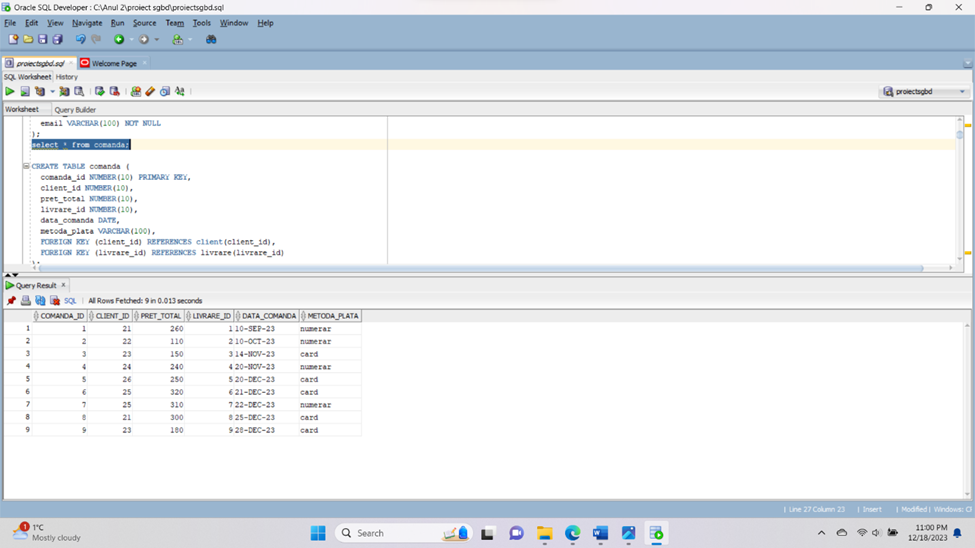
INSERT INTO comanda VALUES (5, 26,250 ,5,TO\_DATE('20-12-2023','DD-MM-YYYY'),'card');

INSERT INTO comanda VALUES (6, 25,320 ,6,TO\_DATE('21-12-2023','DD-MM-YYYY'),'card');

INSERT INTO comanda VALUES (7, 25,310 ,7,TO\_DATE('22-12-2023','DD-MM-YYYY'),'numerar');

INSERT INTO comanda VALUES (8, 21,300 ,8,TO\_DATE('25-12-2023','DD-MM-YYYY'),'card');

INSERT INTO comanda VALUES (9, 23,180 ,9,TO\_DATE('28-12-2023','DD-MM-YYYY'),'card');



INSERT INTO produs VALUES (123, 11, 12, 'rochie midi decupata', 150, 'Rochie midi confectionata cu tesatura din amestec de viscoza, cu guler rotund, fara maneca, cu decupaje laterale ajustabile cu funde');

INSERT INTO produs VALUES (124, 15, 12, 'rochie mini cu imprimeu', 120, 'Rochie cu guler rotund, decolteu în V, mâneca lunga cu elastic în partea inferioara si aplicatie cu insertii de dantela');

INSERT INTO produs VALUES (125, 11, 12, 'camasa satinata', 100, 'Camasa cu guler si rever. Decolteu în V. Mâneca lunga cu manseta. Închidere frontala cu nasturi');

INSERT INTO produs VALUES (126, 15, 12, 'fusta mini denim', 90, 'Fusta cu talie înalta. Cinci buzunare. Efect prespalat. Închidere frontala cu fermoar si nasture metalic');

INSERT INTO produs VALUES (127, 15, 12, 'fusta-pantalon cu cute', 110, 'Fusta-pantalon cu talie înalta. Fermoar ascuns în cusatura. Dublura la interior, tip shorts');

INSERT INTO produs VALUES (128, 12, 13, 'botine joase', 250, 'Pantofi tip ghete plate din piele. Catarame în lateral. Talpa tip track. Închidere prin intermediul unui fermoar. Vârf rotund');

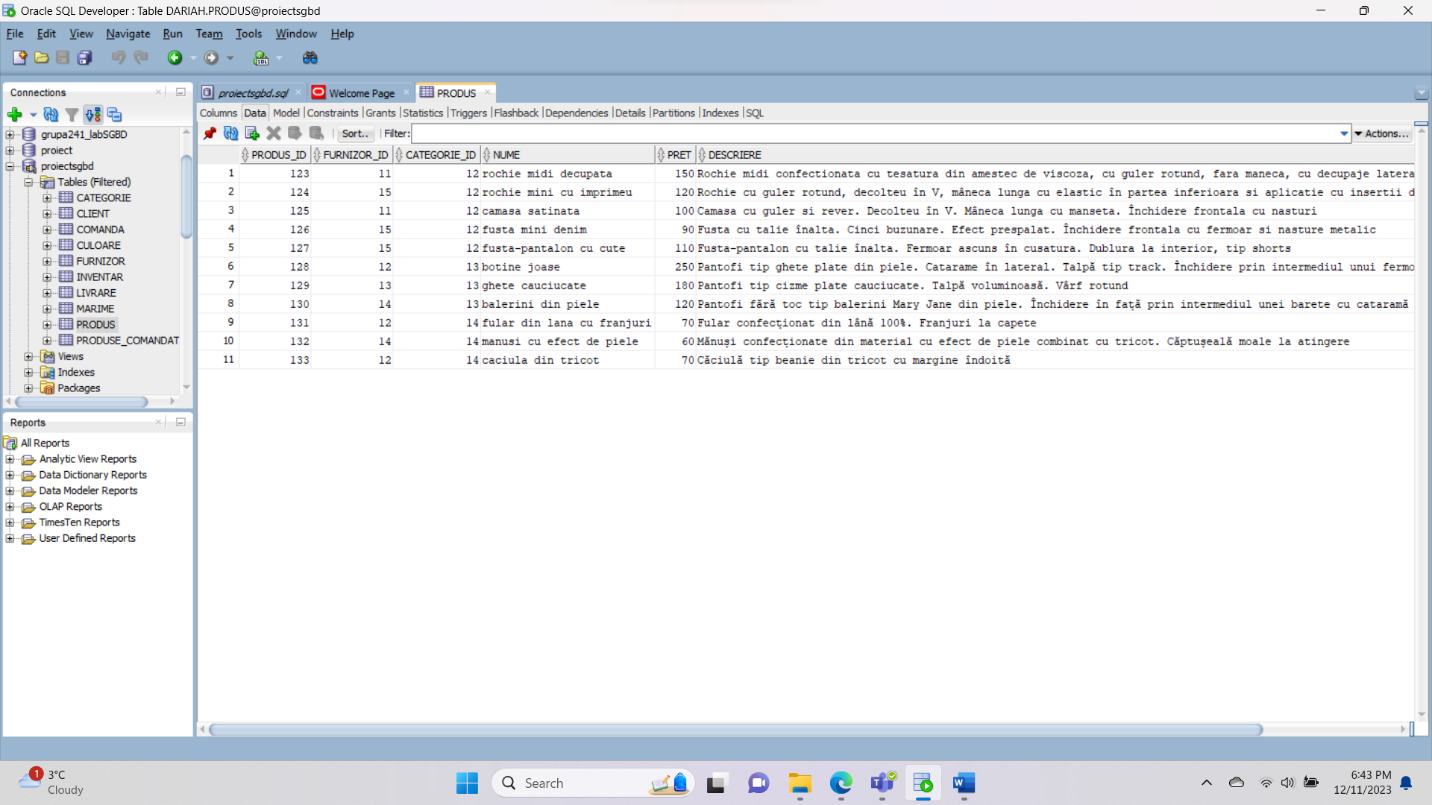
INSERT INTO produs VALUES (129, 13, 13, 'ghete cauciucate', 180, 'Pantofi tip cizme plate cauciucate. Talpa voluminoasa. Vârf rotund');

INSERT INTO produs VALUES (130, 14, 13, 'balerini din piele', 120, 'Pantofi fara toc tip balerini Mary Jane din piele. Închidere în fata prin intermediul unei barete cu catarama');

INSERT INTO produs VALUES (131, 12, 14, 'fular din lana cu franjuri', 70, 'Fular confectionat din lâna 100%. Franjuri la capete');

INSERT INTO produs VALUES (132, 14, 14, 'manusi cu efect de piele', 60, 'Manusi confectionate din material cu efect de piele combinat cu tricot. Captuseala moale la atingere');

INSERT INTO produs VALUES (133, 12, 14, 'caciula din tricot', 70, 'Caciula tip beanie din tricot cu margine îndoita');



INSERT INTO marime VALUES (1,32);

INSERT INTO marime VALUES (2,34);

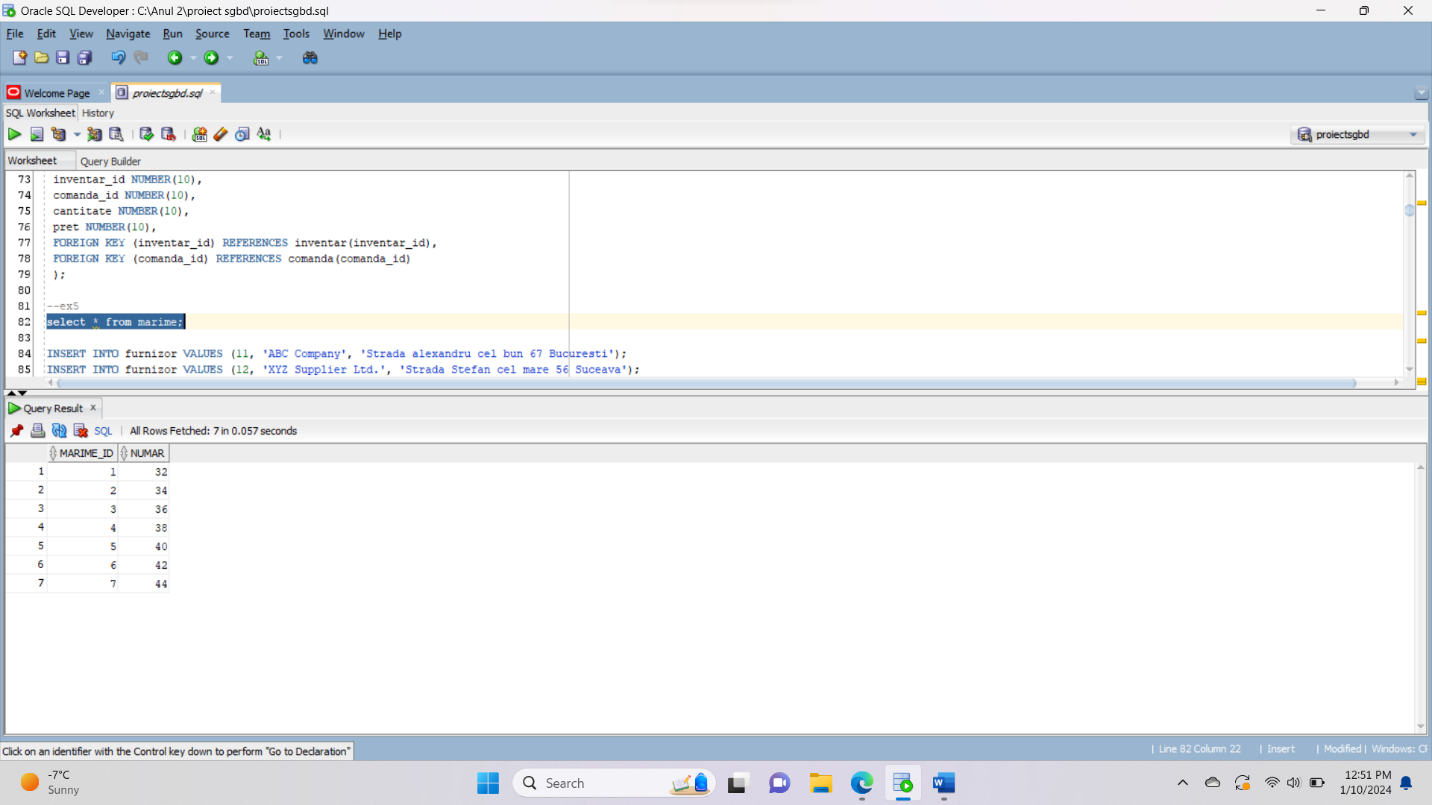
INSERT INTO marime VALUES (3,36);

INSERT INTO marime VALUES (4,38);

INSERT INTO marime VALUES (5,40);

INSERT INTO marime VALUES (6,42);

INSERT INTO marime VALUES (7,44);



INSERT INTO culoare VALUES (11,'rosu');

INSERT INTO culoare VALUES (12,'albastru');

INSERT INTO culoare VALUES (13,'alb');

INSERT INTO culoare VALUES (14,'negru');

INSERT INTO culoare VALUES (15,'verde');

INSERT INTO culoare VALUES (16,'maro');

INSERT INTO culoare VALUES (17,'violet');

INSERT INTO culoare VALUES (18,'roz');



INSERT INTO inventar VALUES (1,123,14,2);

INSERT INTO inventar VALUES (2,123,14,3);

INSERT INTO inventar VALUES (3,123,14,4);

INSERT INTO inventar VALUES (4,124,11,2);

INSERT INTO inventar VALUES (5,124,11,3);

INSERT INTO inventar VALUES (6,124,11,4);

INSERT INTO inventar VALUES (7,124,16,2);

INSERT INTO inventar VALUES (8,124,16,3);

INSERT INTO inventar VALUES (9,124,16,4);

INSERT INTO inventar VALUES (10,124,16,5);

INSERT INTO inventar VALUES (11,125,13,2);

INSERT INTO inventar VALUES (12,125,13,5);

INSERT INTO inventar VALUES (13,125,13,6);

INSERT INTO inventar VALUES (14,125,13,7);

INSERT INTO inventar VALUES (15,126,15,1);

INSERT INTO inventar VALUES (16,126,15,2);

INSERT INTO inventar VALUES (17,127,17,3);

INSERT INTO inventar VALUES (18,127,17,4);

INSERT INTO inventar VALUES (19,128,14,3);

INSERT INTO inventar VALUES (20,128,14,3);

INSERT INTO inventar VALUES (21,129,12,3);

INSERT INTO inventar VALUES (22,130,14,1);

INSERT INTO inventar VALUES (23,130,14,2);

INSERT INTO inventar VALUES (24,130,14,3);

INSERT INTO inventar VALUES (25,130,14,4);

INSERT INTO inventar VALUES (26,131,18,3);

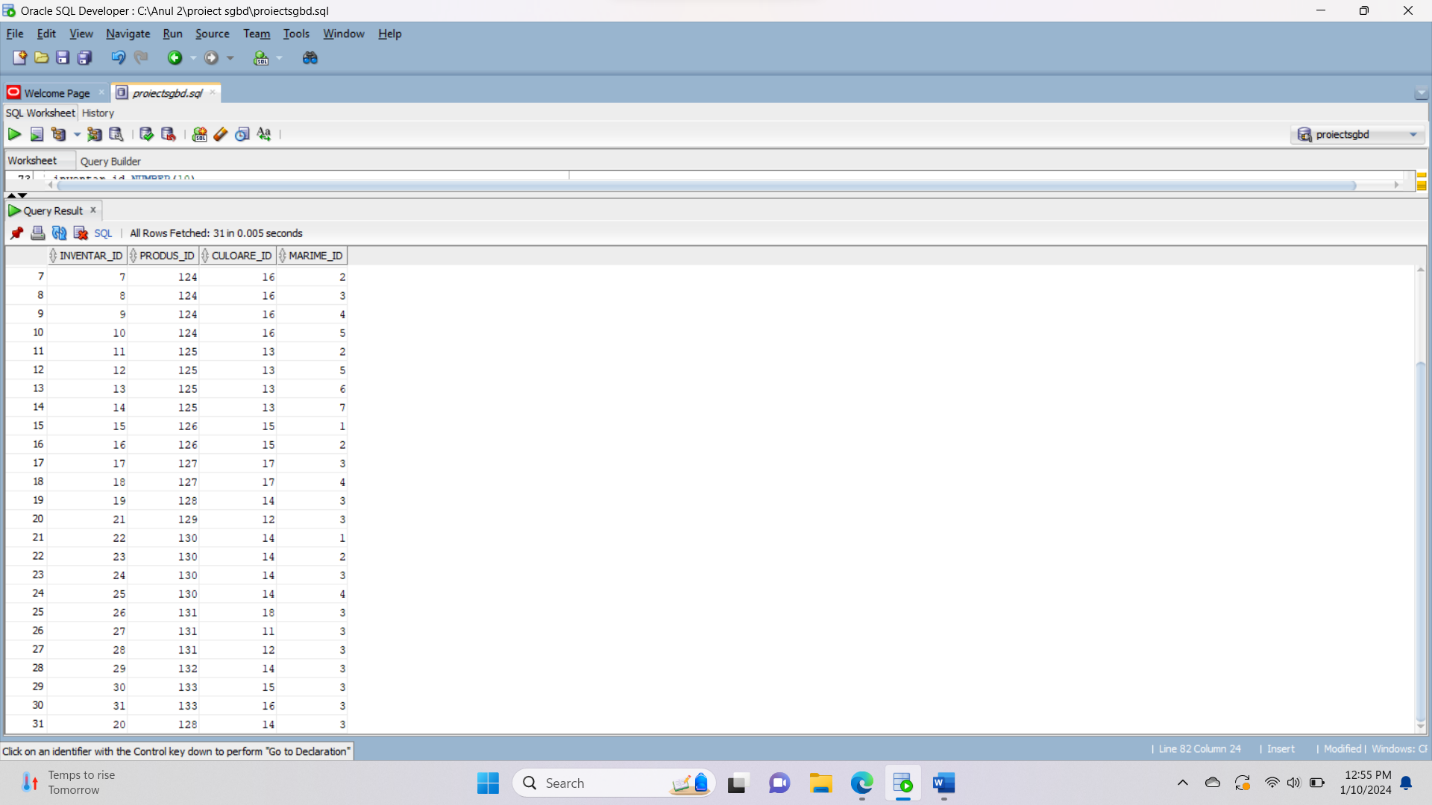
INSERT INTO inventar VALUES (27,131,11,3);

INSERT INTO inventar VALUES (28,131,12,3);

INSERT INTO inventar VALUES (29,132,14,3);

INSERT INTO inventar VALUES (30,133,15,3);

INSERT INTO inventar VALUES (31,133,16,3);



INSERT INTO produse\_comandate VALUES (1,29,1,1,60);

INSERT INTO produse\_comandate VALUES (2,12,1,2,200);

INSERT INTO produse\_comandate VALUES (3,18,2,1,110);

INSERT INTO produse\_comandate VALUES (4,3,3,1,150);

INSERT INTO produse\_comandate VALUES (5,13,4,1,100);

INSERT INTO produse\_comandate VALUES (6,31,4,1,70);

INSERT INTO produse\_comandate VALUES (7,27,4,1,70);

INSERT INTO produse\_comandate VALUES (8,19,5,1,250);

INSERT INTO produse\_comandate VALUES (9,19,6,1,250);

INSERT INTO produse\_comandate VALUES (10,28,6,1,70);

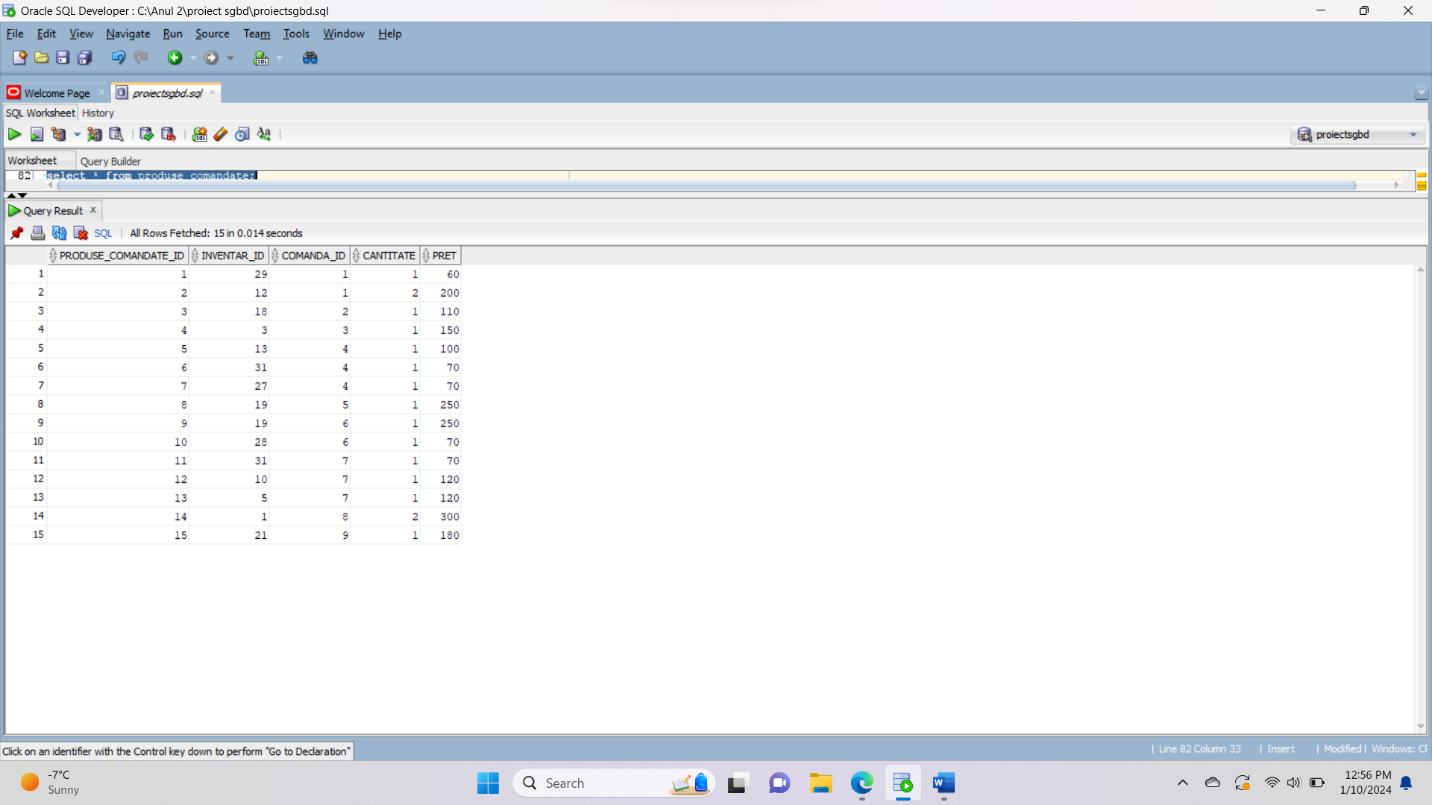
INSERT INTO produse\_comandate VALUES (11,31,7,1,70);

INSERT INTO produse\_comandate VALUES (12,10,7,1,120);

INSERT INTO produse\_comandate VALUES (13,5,7,1,120);

INSERT INTO produse\_comandate VALUES (14,1,8,2,300);

INSERT INTO produse\_comandate VALUES (15,21,9,1,180);



6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Se da id-ul unui produs. Afisati pentru fiecare comanda care contine respectivul produs: detalii despre client(nume, prenume, email si id), id-ul comenzii si produsele. Afisati 'nu exista' in cazul in care nu au fost plasate comenzi cu acel produs

CREATE OR REPLACE PROCEDURE afisare\_produse\_comanda ( p\_produs\_id produs.produs\_id%TYPE)

AS

TYPE clienti\_i IS TABLE OF client%ROWTYPE; -- tablou imbricat

clienti\_table clienti\_i := clienti\_i();

TYPE vector IS VARRAY(100) OF comanda%ROWTYPE; -- VARRAY

comanda\_table vector := vector();

numar NUMBER(10):= 0;

BEGIN

--numaram cate comenzi au produsul respectiv

SELECT COUNT(\*)

INTO numar

FROM produse\_comandate pc, inventar i,comanda c

WHERE i.inventar\_id = pc.inventar\_id and pc.inventar\_id=c.comanda\_id and i.produs\_id=p\_produs\_id;

comanda\_table.extend(numar+1);

--selectam datele despre o comanda care are produsul respectiv

SELECT distinct c.\*

BULK COLLECT INTO comanda\_table

FROM comanda c

JOIN produse\_comandate pc ON c.comanda\_id = pc.comanda\_id

JOIN inventar i ON pc.inventar\_id = i.inventar\_id

WHERE i.produs\_id = p\_produs\_id;

IF comanda\_table.COUNT=0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista comenzi cu produsul respectiv!');

RETURN;

END IF;

--selectam doar clientii care au dat o comanda cu produsul respectiv

FOR i IN comanda\_table.FIRST .. comanda\_table.LAST LOOP

clienti\_table.EXTEND;

SELECT DISTINCT cl.\*

INTO clienti\_table(clienti\_table.LAST)

FROM client cl

WHERE cl.client\_id = comanda\_table(i).client\_id;

END LOOP;

--daca am gasit comenzi afisam produsele

IF comanda\_table.COUNT > 0 THEN

FOR i IN clienti\_table.FIRST .. clienti\_table.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Client ID: ' || clienti\_table(i).client\_id);

DBMS\_OUTPUT.PUT\_LINE(clienti\_table(i).nume ||' ' || clienti\_table(i).prenume);

DBMS\_OUTPUT.PUT\_LINE(clienti\_table(i).email);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------');

FOR j IN comanda\_table.FIRST .. comanda\_table.LAST LOOP

IF comanda\_table(j).client\_id=clienti\_table(i).client\_id THEN

DECLARE

TYPE produse\_i IS TABLE OF VARCHAR(100) INDEX BY PLS\_INTEGER; --tablou indexat

produse\_table produse\_i;

BEGIN

SELECT p.nume

BULK COLLECT INTO produse\_table

FROM produse\_comandate pc, inventar i, produs p

WHERE p.produs\_id=i.produs\_id and i.inventar\_id = pc.inventar\_id and pc.comanda\_id=comanda\_table(j).comanda\_id;

DBMS\_OUTPUT.PUT\_LINE(' Comanda ID: ' || comanda\_table(j).comanda\_id);

DBMS\_OUTPUT.PUT\_LINE('---------------------');

IF produse\_table.COUNT > 0 THEN

FOR k IN produse\_table.FIRST .. produse\_table.LAST LOOP

numar := numar + 1;

DBMS\_OUTPUT.PUT\_LINE(k || '. ' || produse\_table(k));

END LOOP;

END IF;

END;

END IF;

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END IF;

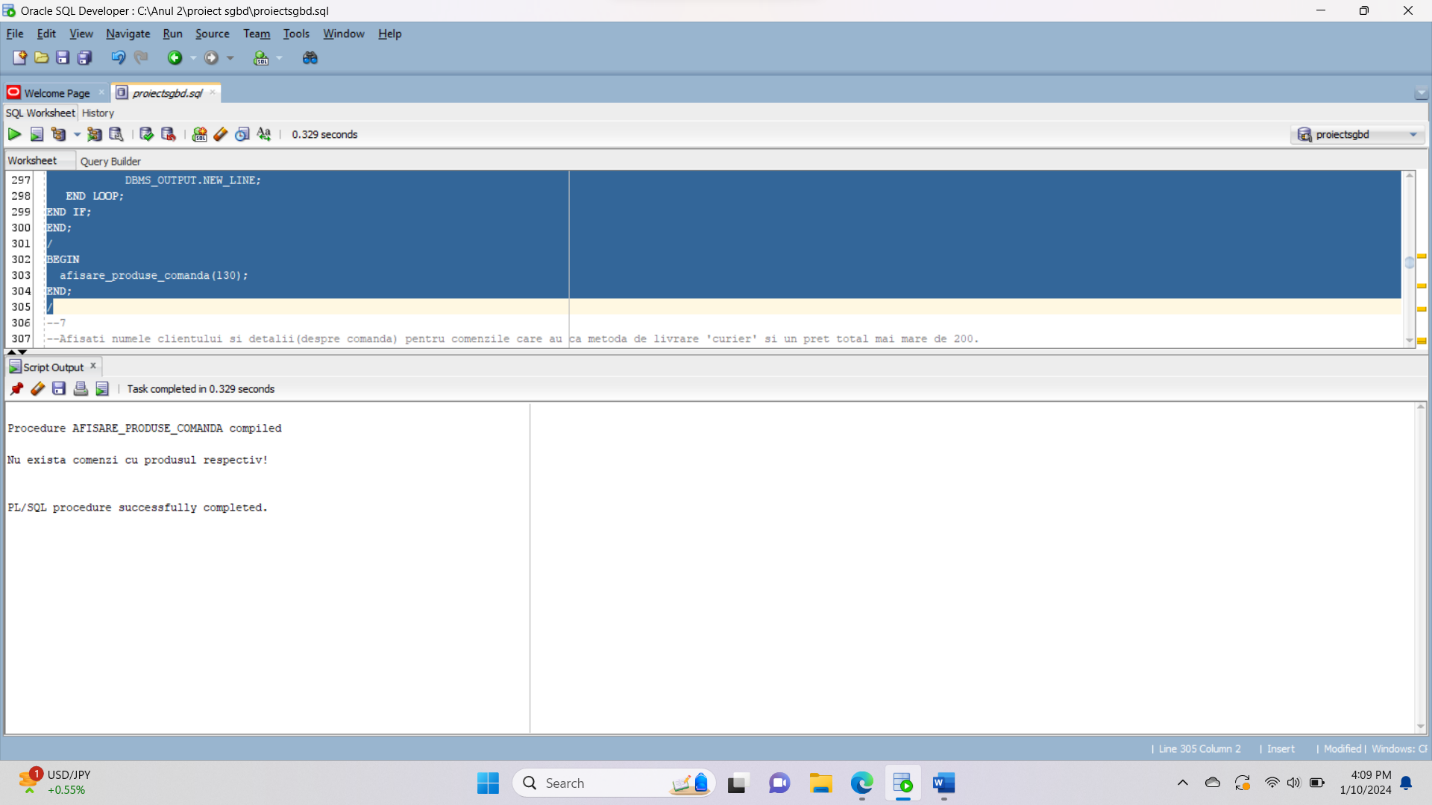
END;

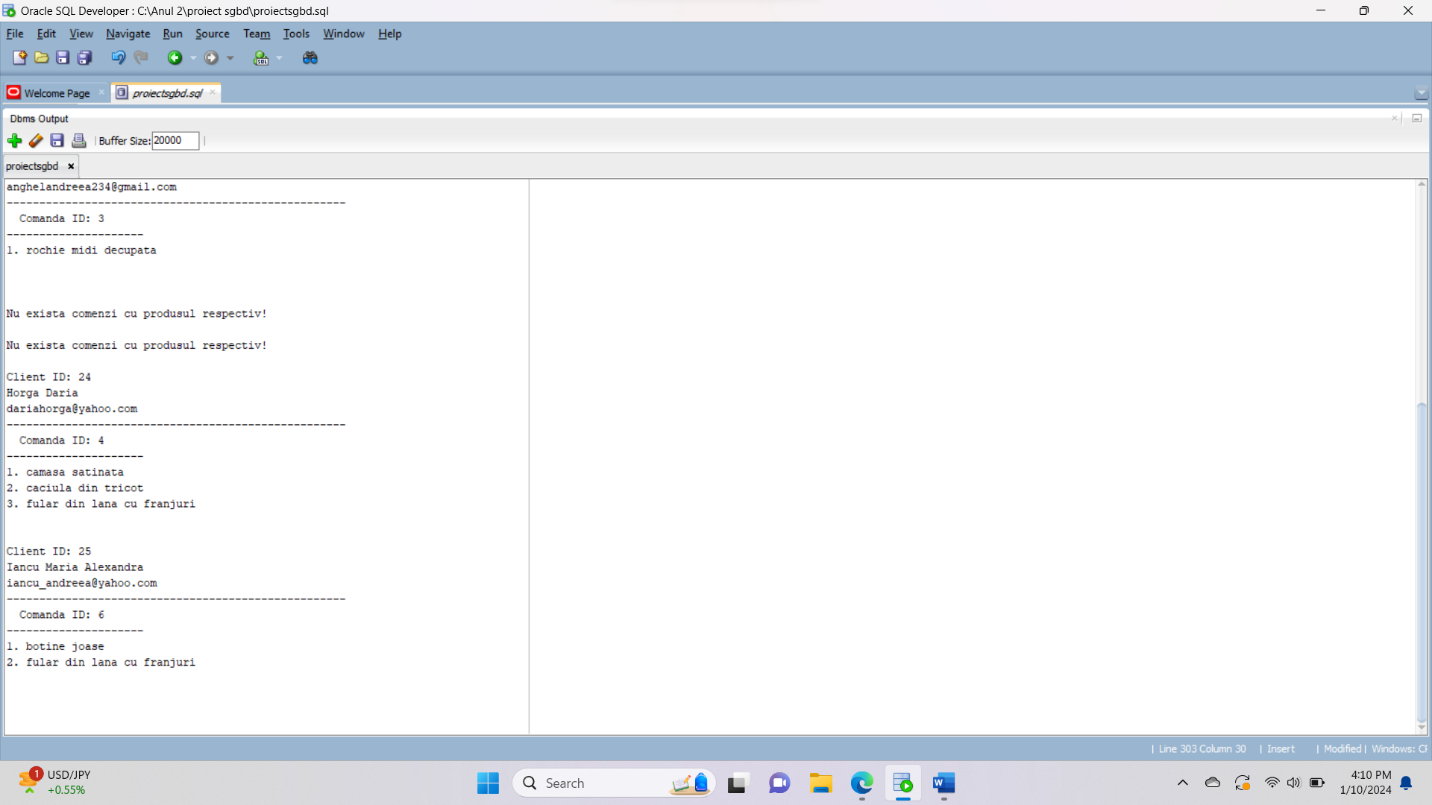
/

BEGIN

afisare\_produse\_comanda(130); END;

/





7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

Afisati numele clientului si detalii(despre comanda) pentru comenzile care au ca metoda de livrare 'curier' si un pret total mai mare de 200.

CREATE OR REPLACE PROCEDURE afisare\_clienti\_comanda( p\_livrare livrare.metoda\_livrare%TYPE, p\_pret comanda.pret\_total%TYPE) AS

--selectam numele si prenumele clientului avand id-ul de livrare

CURSOR clienti (id comanda.livrare\_id%TYPE) IS

SELECT c.nume || ' ' || c.prenume AS result

FROM client c

JOIN comanda co ON c.client\_id = co.client\_id

WHERE co.livrare\_id = id;

--selectam id-ul livrarilor care au ca metoda de livrare curier, si au ca pret\_total o suma mai mare de p\_pret

CURSOR livrari (p\_livrare livrare.metoda\_livrare%TYPE) IS

SELECT l.livrare\_id

FROM livrare l

JOIN comanda co on co.livrare\_id=l.livrare\_id

WHERE UPPER(l.metoda\_livrare) LIKE UPPER(p\_livrare) AND co.pret\_total >= p\_pret;

--selectam comenzile avand id-ul lor de livrare

CURSOR comenzi (id livrare.livrare\_id%TYPE ) IS

SELECT comanda\_id, pret\_total, metoda\_plata

FROM comanda

WHERE livrare\_id = id;

id comanda.comanda\_id%TYPE;

pret comanda.pret\_total%TYPE;

plata comanda.metoda\_plata%TYPE;

BEGIN

FOR livrari\_i in livrari(p\_livrare) LOOP

OPEN comenzi(livrari\_i.livrare\_id);

FETCH comenzi INTO id,pret,plata;

DBMS\_OUTPUT.PUT\_LINE('Comanda cu numarul ' || id);

DBMS\_OUTPUT.PUT\_LINE('------------------------');

DBMS\_OUTPUT.PUT\_LINE('Pret:' || pret);

DBMS\_OUTPUT.PUT\_LINE('Metoda plata:' || plata);

CLOSE comenzi;

FOR clienti\_i IN clienti(livrari\_i.livrare\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(clienti\_i.result);

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END;

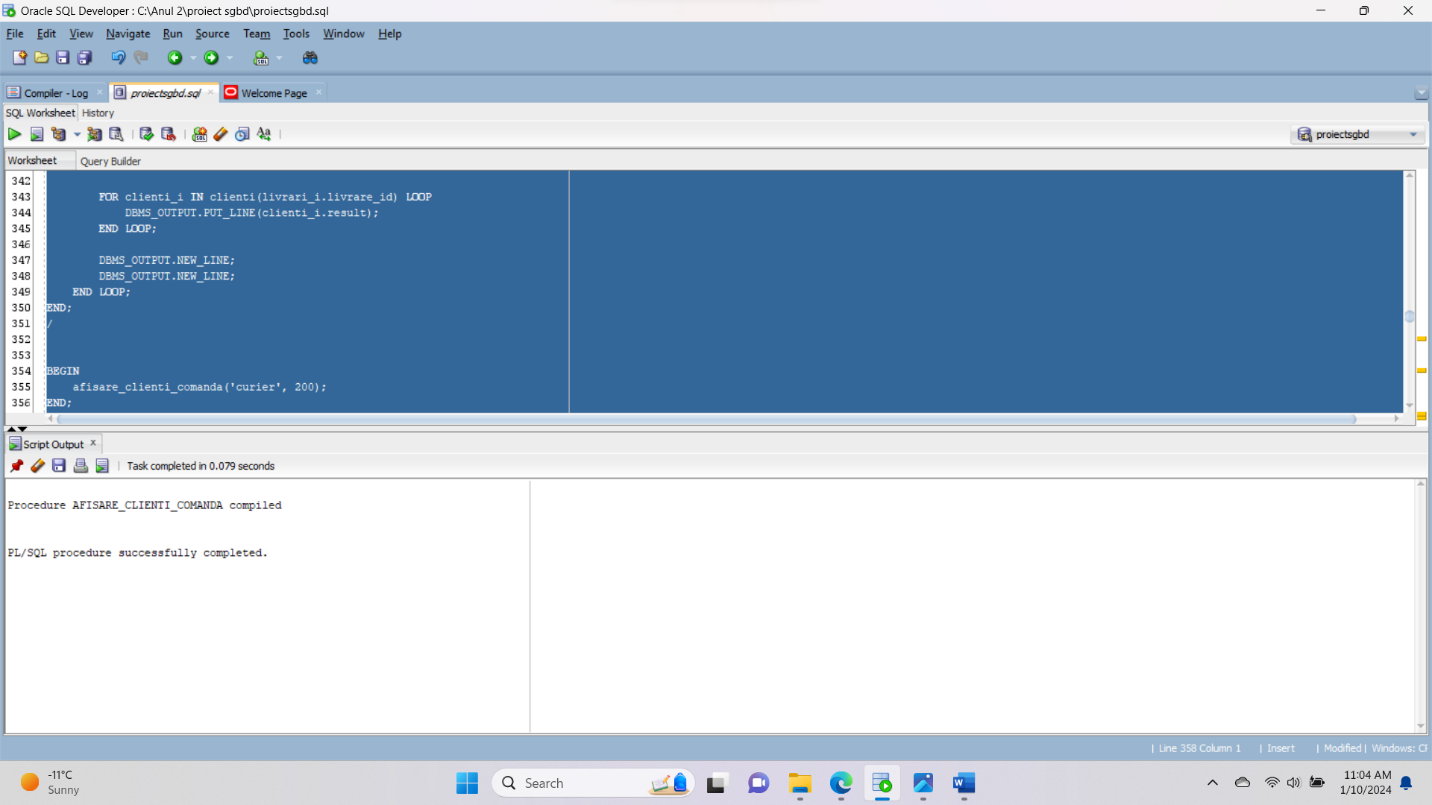
/

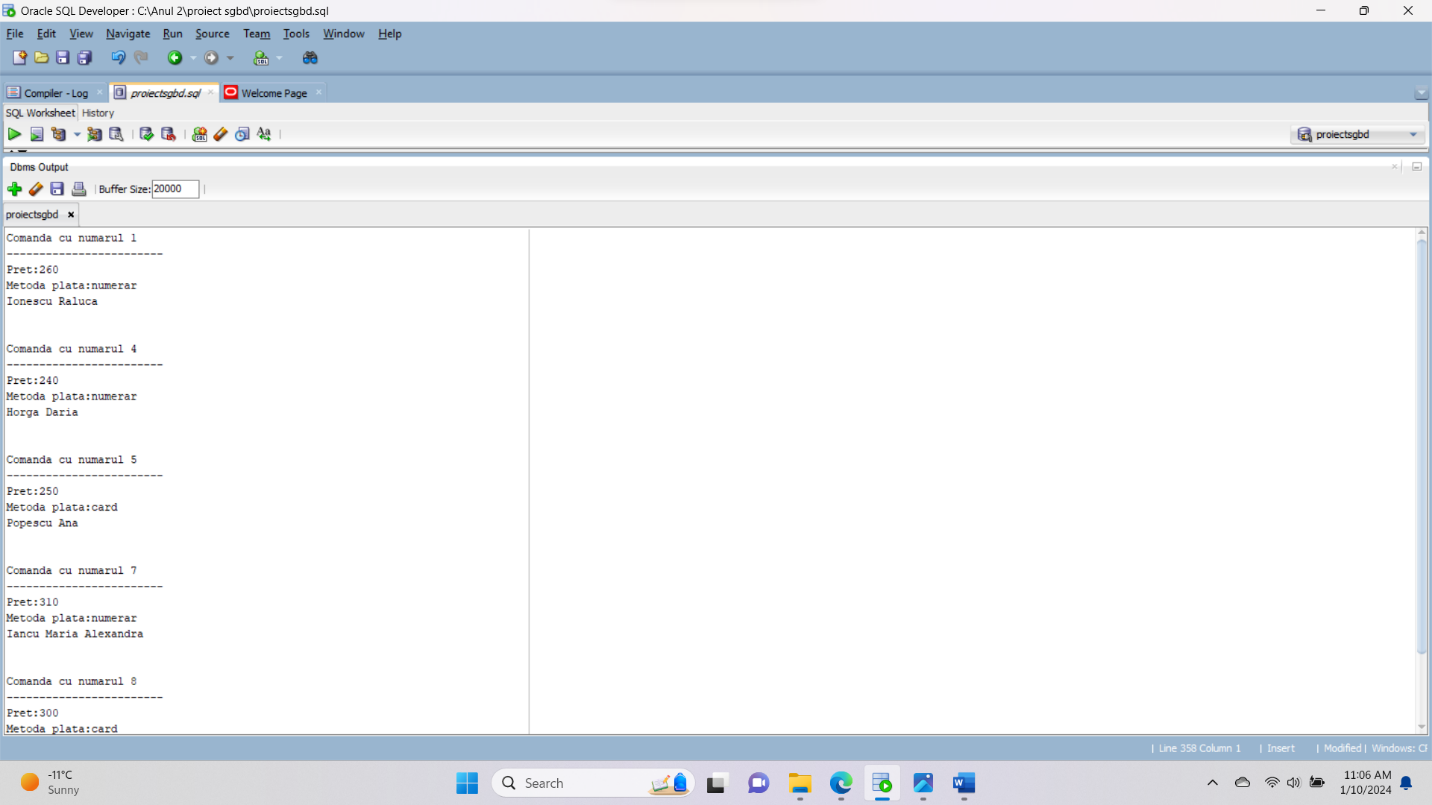
BEGIN

afisare\_clienti\_comanda('curier', 200);

END;

/





8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Definiți minim 2 excepții proprii. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

Avand o culoare de produse data, determinati numarul de produse de 'imbracaminte' cumparate pe acea culoare.

CREATE OR REPLACE FUNCTION afisare\_produse\_culoare(id culoare.culoare\_id%TYPE, categ\_ categorie.nume%type) RETURN NUMBER

IS

nr\_produse NUMBER;

TYPE tip\_tabel IS TABLE OF inventar%rowtype INDEX BY PLS\_INTEGER;

tabel tip\_tabel;

categ categorie.categorie\_id%type;

--exceptii

NEGATIVE\_NUMBER EXCEPTION;

NO\_DATA\_FOUND1 EXCEPTION;

NO\_DATA\_FOUND2 EXCEPTION;

BEGIN

IF id < 0 THEN -- codul culorii nu e valid

RAISE NEGATIVE\_NUMBER;

END IF;

--selectam toate produsele din inventar care sunt pe culoarea respectiva

SELECT \*

BULK COLLECT INTO tabel

FROM inventar

WHERE culoare\_id = id;

--avem mesaj de eroare in cazul in care nu le gasim

IF SQL%NOTFOUND THEN

RAISE NO\_DATA\_FOUND1;

END IF;

--cautam id-ul pentru 'imbracaminte'

SELECT categorie\_id

INTO categ

FROM categorie

WHERE UPPER(nume) LIKE UPPER(categ\_);

--cautam numarul de produse cu respectivele conditii

SELECT COUNT(i.inventar\_id)

INTO nr\_produse

FROM produse\_comandate pc

JOIN inventar i ON (i.inventar\_id = pc.inventar\_id)

JOIN produs p on (p.produs\_id=i.produs\_id)

WHERE i.culoare\_id = id and p.categorie\_id=categ;

--eroare in cazul in care nu exista produse

IF nr\_produse = 0 THEN

RAISE NO\_DATA\_FOUND2;

END IF;

RETURN nr\_produse;

EXCEPTION

WHEN NEGATIVE\_NUMBER THEN

DBMS\_OUTPUT.PUT\_LINE('Codul culorii nu poate sa fie negativ!');

RETURN -1;

WHEN NO\_DATA\_FOUND1 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produse pe culoarea cu codul ' || id);

RETURN -1;

WHEN NO\_DATA\_FOUND2 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produse de imbracaminte cumparate cu acea culoare!');

RETURN -1;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Codul de eroare: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE('Mesajul erorii: ' || SQLERRM);

RETURN -1;

END;

/

DECLARE

numar NUMBER;

id culoare.culoare\_id%type:= &id;

categ\_ categorie.nume%type := 'imbracaminte';

BEGIN

numar := afisare\_produse\_culoare(id,categ\_);

IF numar > -1 THEN

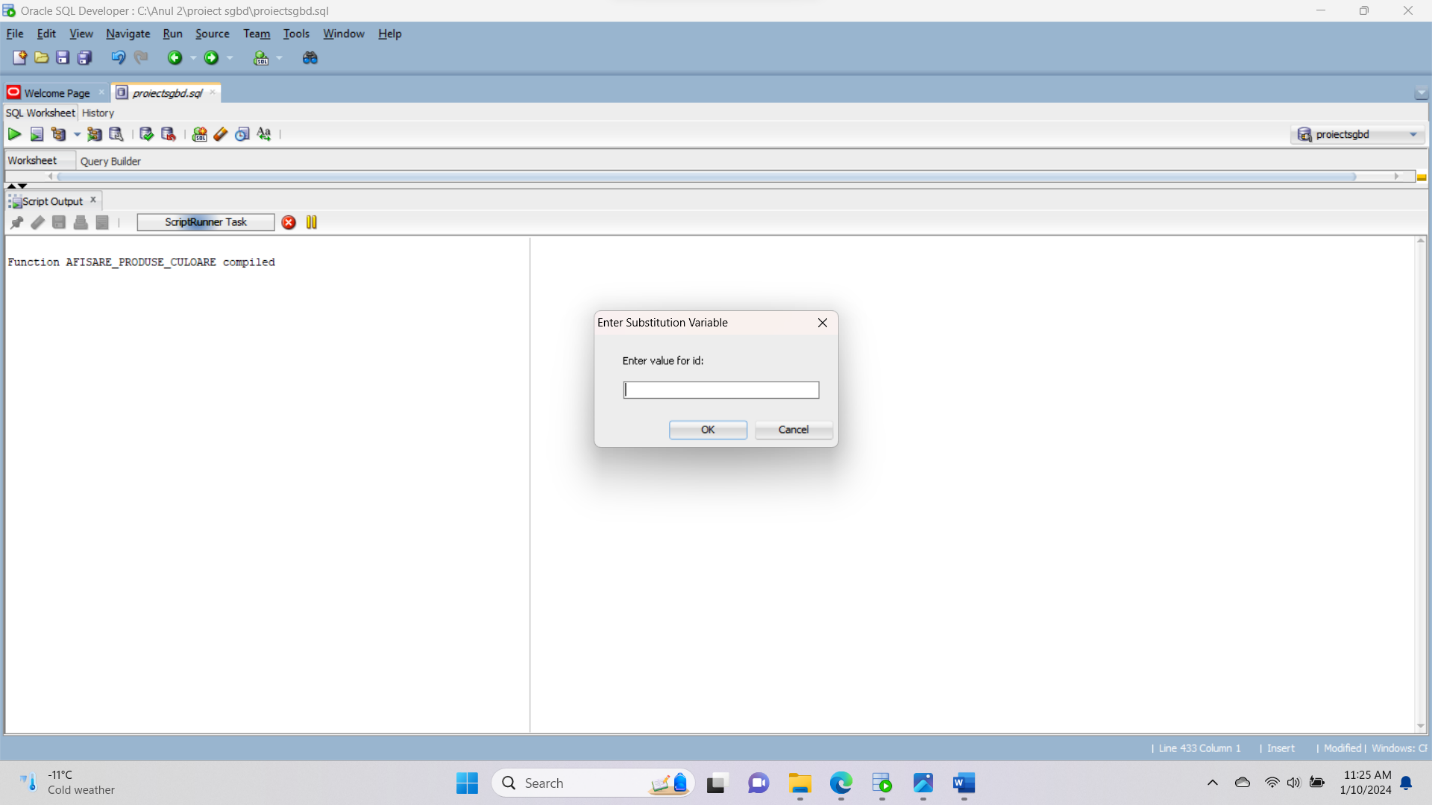
DBMS\_OUTPUT.PUT\_LINE('Rezultatul pentru culoarea cu id-ul ' || id ||' este: ' || numar );

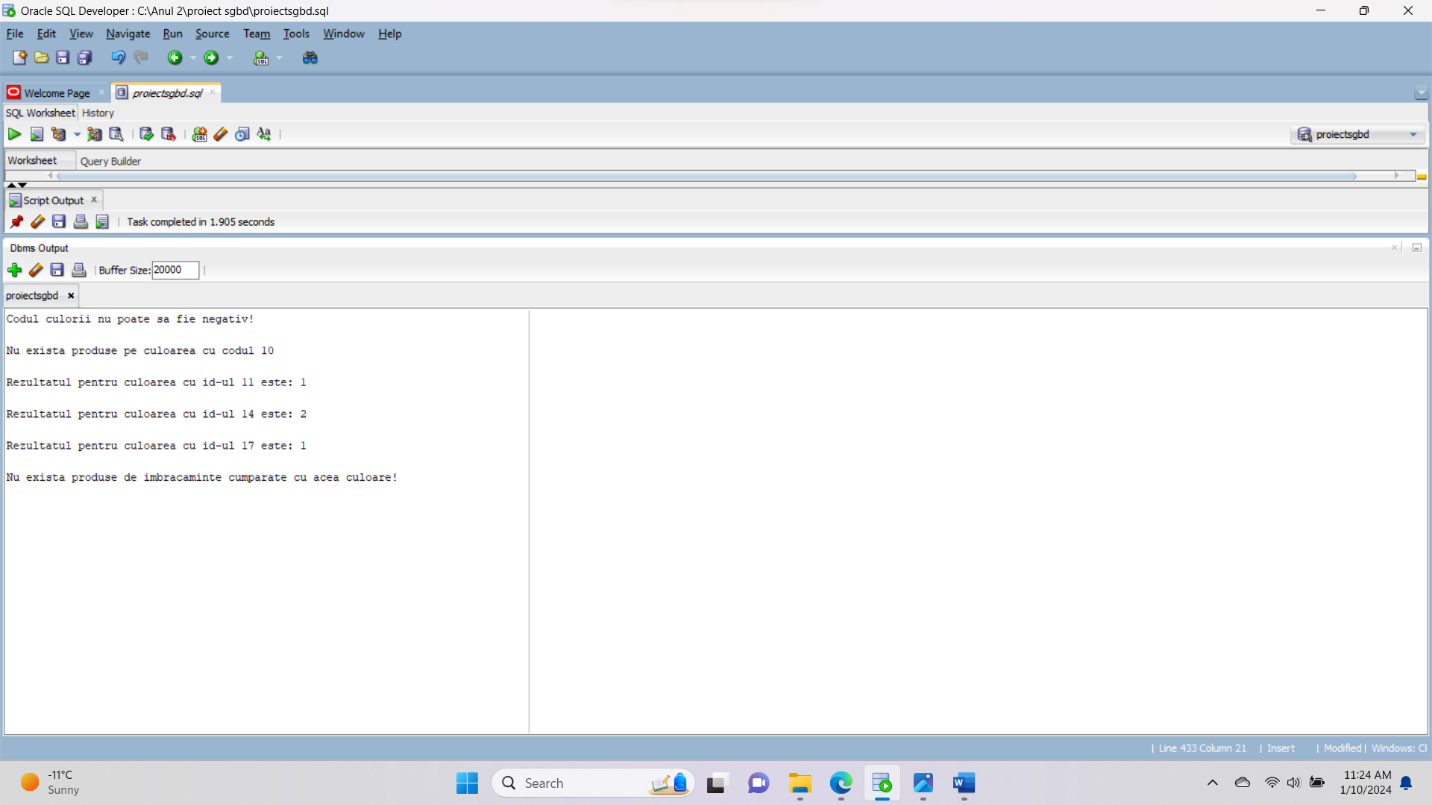
END IF;

END;

/

SET SERVEROUTPUT ON;





9.Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să utilizeze într-o singură comandă SQL 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea, incluzând excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Afisati id-ul comenzilor pe care le-a plasat un client, al carui nume este dat de la tastatura, luand in calcul doar comenzile care contin produse de la furnizorul 'XYZ Supplier Ltd.'

CREATE OR REPLACE PROCEDURE afisare\_clienti\_comenzi\_(nume\_client client.nume%type,nume\_furnizor furnizor.nume%type)

AS

TYPE tabel\_index IS TABLE OF comanda.comanda\_id%type INDEX BY PLS\_INTEGER;

comenzi tabel\_index;

TYPE tabel\_index\_clienti IS TABLE OF client%rowtype INDEX BY PLS\_INTEGER;

clienti tabel\_index\_clienti;

NO\_DATA\_FOUND1 EXCEPTION;

NO\_DATA\_FOUND2 EXCEPTION;

TOO\_MANY\_ROWS1 EXCEPTION;

BEGIN

--selectam clientii cu numele respectiv

SELECT \*

BULK COLLECT INTO clienti

FROM client

WHERE UPPER(nume) = UPPER(nume\_client);

--eroare daca nu il gasim

IF SQL%NOTFOUND THEN

RAISE NO\_DATA\_FOUND1;

END IF;

--eroare daca avem mai multi clienti cu numele respectiv

IF clienti.count >= 2 THEN

RAISE TOO\_MANY\_ROWS1;

END IF;

--selectam id-ul comenzii cu proprietatile respective

SELECT DISTINCT c.comanda\_id

BULK COLLECT INTO comenzi

FROM comanda c

JOIN produse\_comandate pc ON pc.comanda\_id = c.comanda\_id

JOIN inventar i ON i.inventar\_id = pc.inventar\_id

JOIN client cl ON (cl.client\_id = c.client\_id)

JOIN produs p ON p.produs\_id=i.produs\_id

JOIN furnizor f ON f.furnizor\_id=p.furnizor\_id

WHERE UPPER(f.nume) = UPPER(nume\_furnizor) and UPPER(cl.nume)=UPPER(nume\_client);

IF comenzi.count = 0 THEN

RAISE NO\_DATA\_FOUND2;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Comenzile clientului cu numele ' || nume\_client || ' care contin produse de la furnizorul ' || nume\_furnizor|| ' sunt: ');

FOR i IN comenzi.FIRST..comenzi.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(comenzi(i));

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND1 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti care sa aiba acest nume de familie!');

WHEN NO\_DATA\_FOUND2 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista comenzi care sa aiba produse ce provin de la acel furnizor!');

WHEN TOO\_MANY\_ROWS1 THEN

DBMS\_OUTPUT.PUT\_LINE('Exista mai mult de un client cu acel nume!');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Codul de eroare: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE('Mesajul erorii: ' || SQLERRM);

END;

/

DECLARE

nume client.nume%type :='&nume';

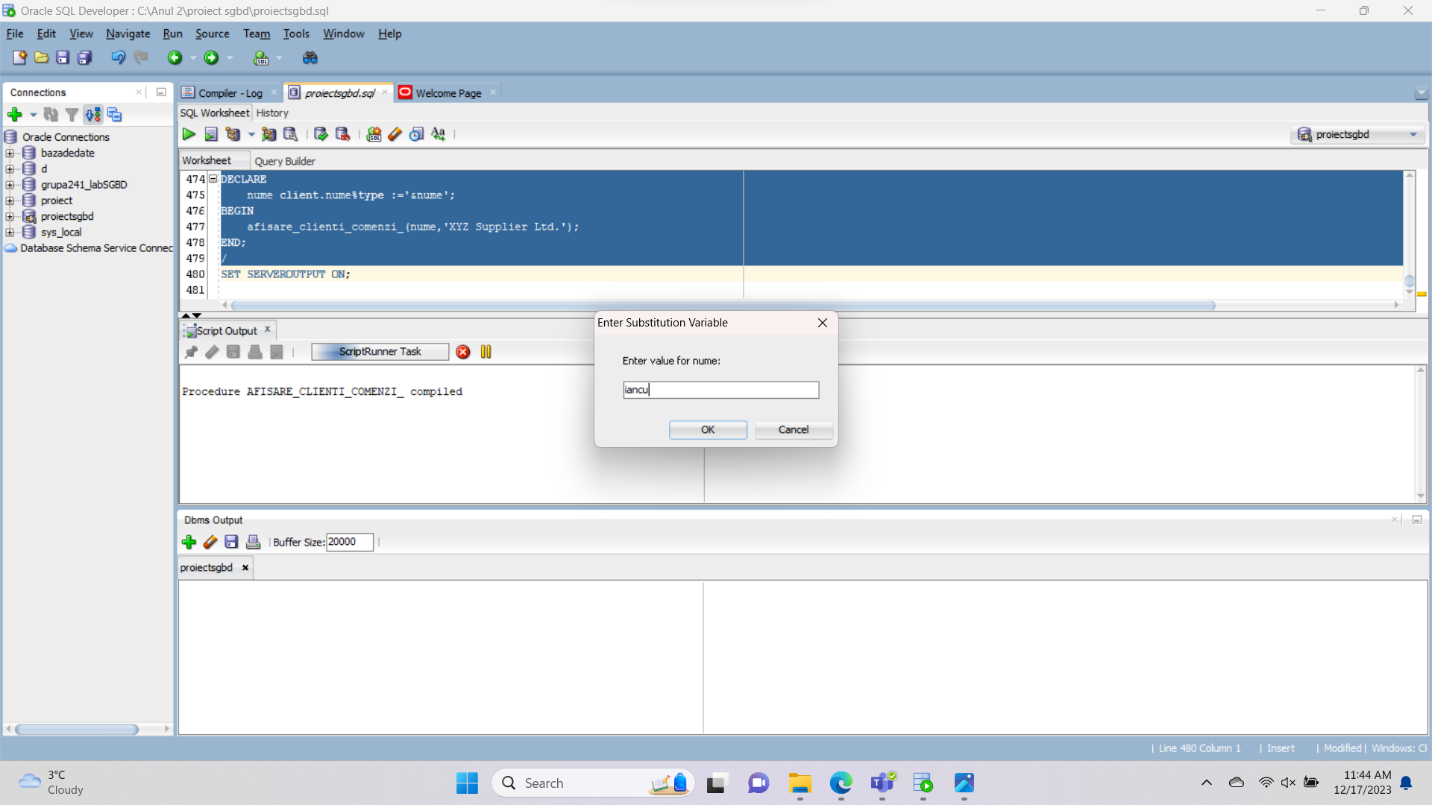
BEGIN

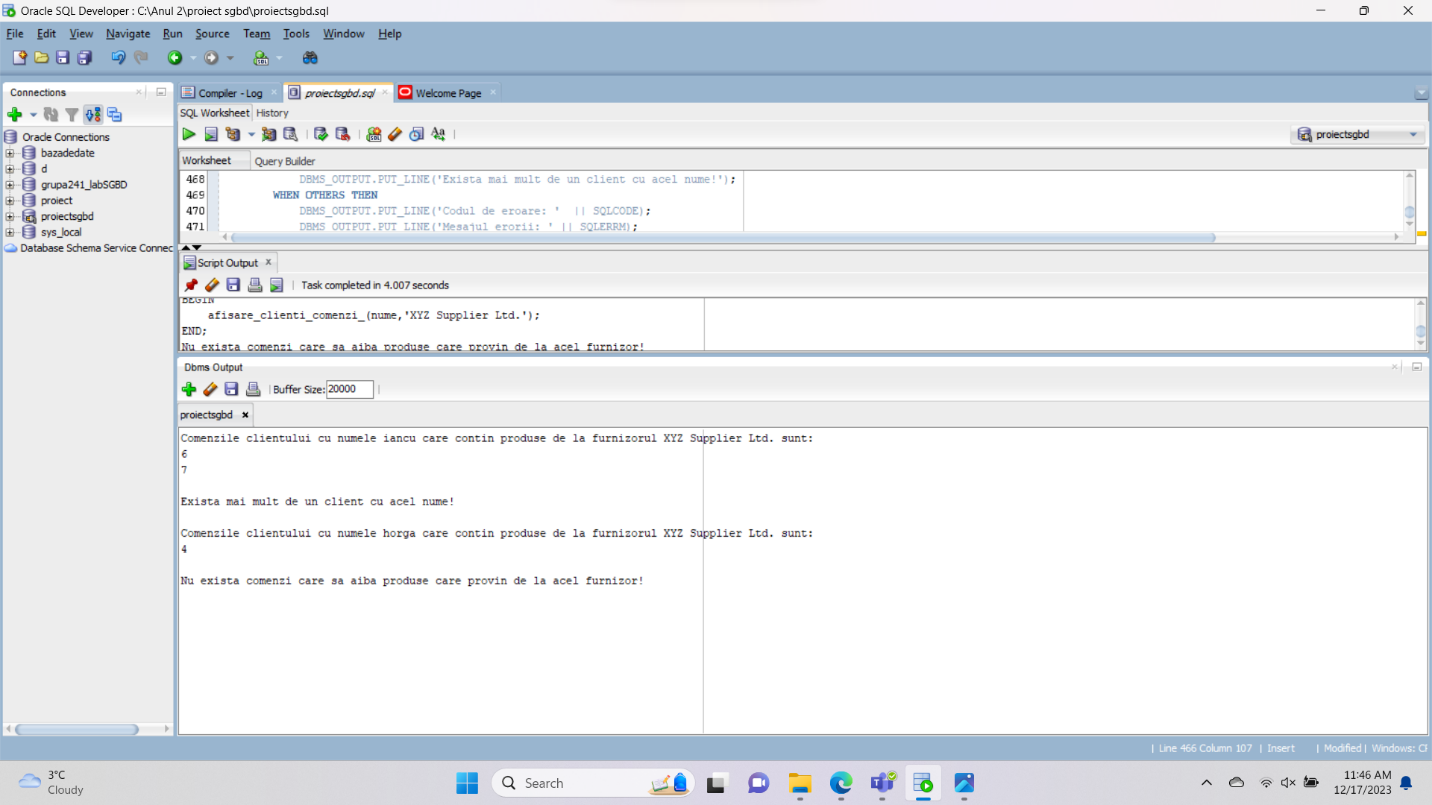
afisare\_clienti\_comenzi\_(nume,'XYZ Supplier Ltd.');

END;

/

SET SERVEROUTPUT ON;





10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.

-- Un trigger care se va declansa atunci cand vreau sa modific tabela comanda in zilele libere sau in intervalul orar 00:00-5:00

--zilele libere sunt : 1 dec, 25 dec si 1 ian

CREATE OR REPLACE TRIGGER Ex10

BEFORE INSERT OR UPDATE OR DELETE ON comanda

DECLARE

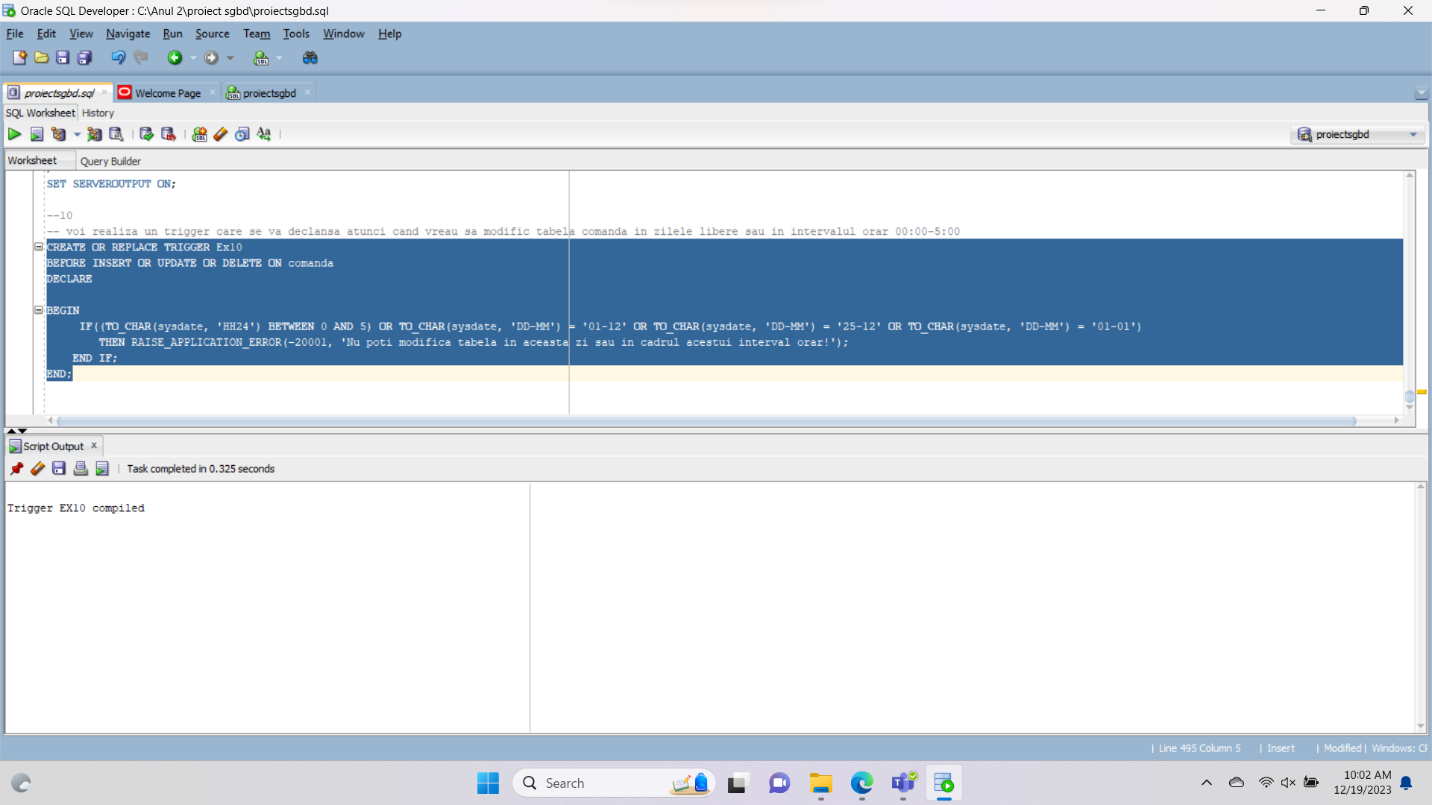
BEGIN

IF((TO\_CHAR(sysdate, 'HH24') BETWEEN 0 AND 5) OR TO\_CHAR(sysdate, 'DD-MM') = '01-12' OR TO\_CHAR(sysdate, 'DD-MM') = '25-12' OR TO\_CHAR(sysdate, 'DD-MM') = '01-01')

THEN RAISE\_APPLICATION\_ERROR(-20001, 'Nu poti modifica tabela in aceasta zi sau in cadrul acestui interval orar!');

END IF;

END;



11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

--Un trigger care se declanseaza atunci cand modific pretul unui produs; afisez pretul nou si cel vechi

CREATE OR REPLACE TRIGGER Ex11

BEFORE INSERT OR UPDATE OR DELETE ON produs

FOR EACH ROW

DECLARE

dif NUMBER;

BEGIN

dif:=:NEW.pret-:Old.pret;

DBMS\_OUTPUT.PUT\_LINE('Pretul vechi: '||:old.pret);

DBMS\_OUTPUT.PUT\_LINE('Pretul nou: '||:new.pret);

DBMS\_OUTPUT.PUT\_LINE('Diferenta: '||dif);

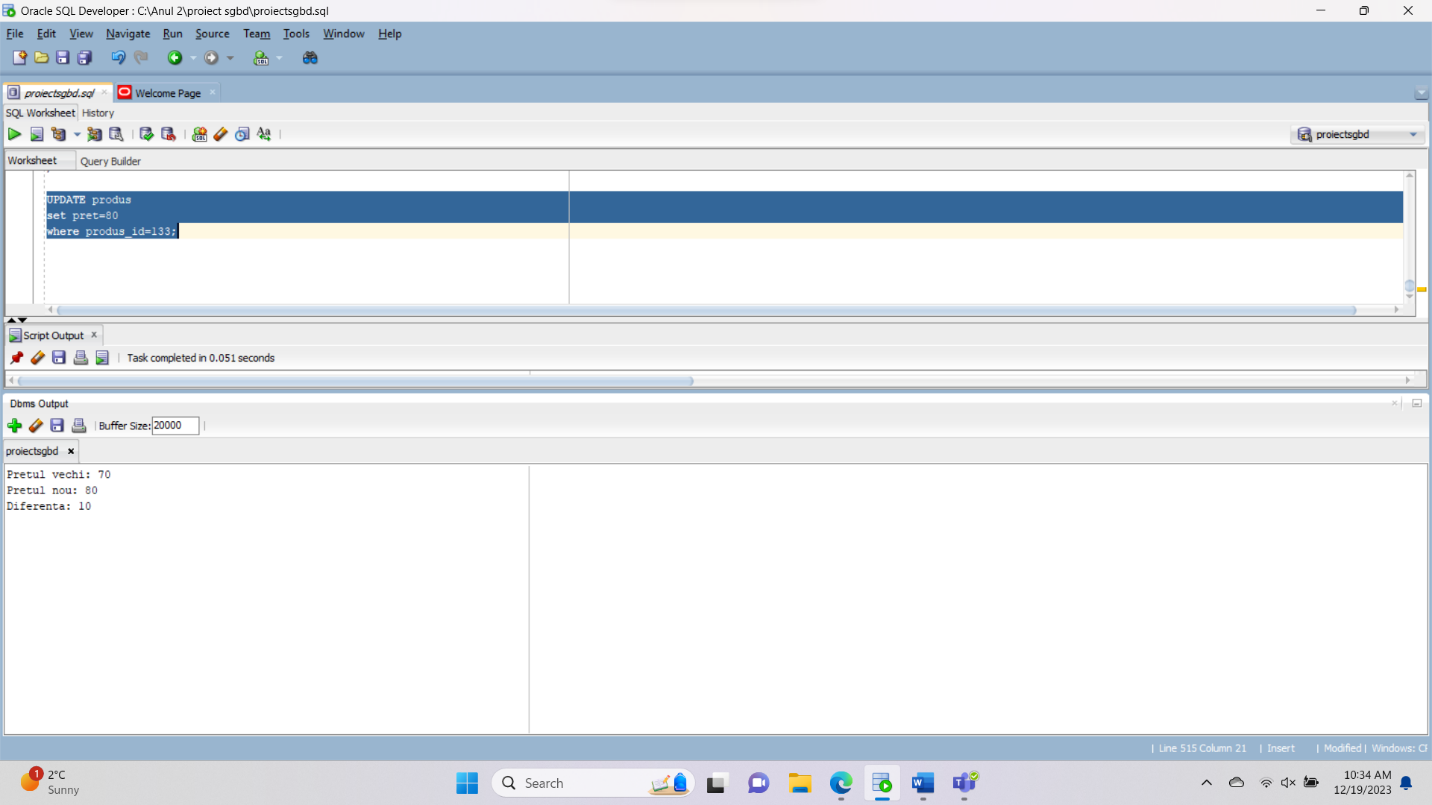
END;

/

UPDATE produs

set pret=80

where produs\_id=133;



12. Definiți un trigger de tip LDD. Declanșați trigger-ul.

--Un trigger care retine in tabela history istoricul comenzilor

CREATE TABLE history(

object\_type VARCHAR(20),

object\_name VARCHAR(30),

event VARCHAR(30)

);

CREATE OR REPLACE TRIGGER t12

BEFORE CREATE OR ALTER OR DROP ON SCHEMA

BEGIN

INSERT INTO history

VALUES(SYS.dictionary\_obj\_type,SYS.dictionary\_obj\_name,SYS.sysevent);

END;

/

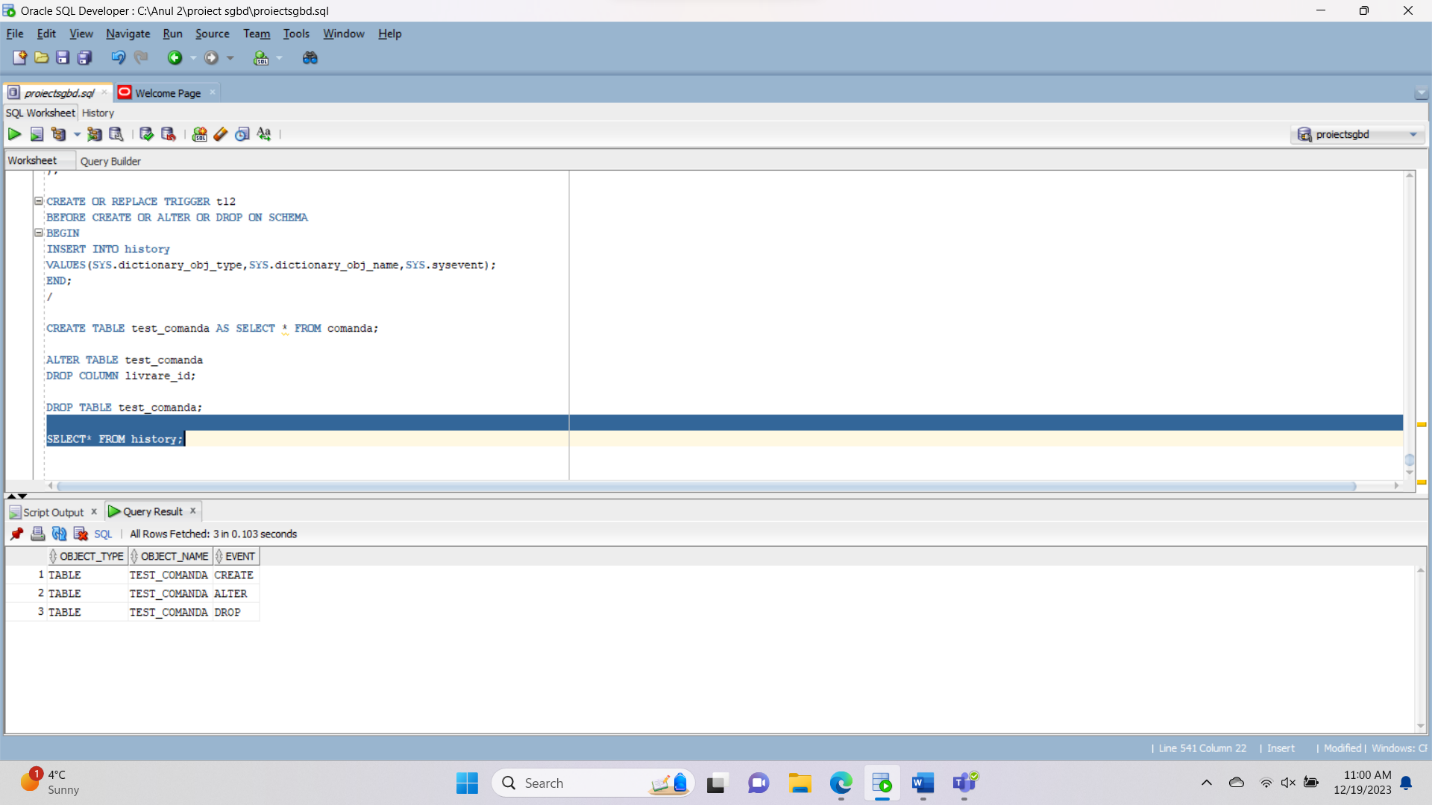
CREATE TABLE test\_comanda AS SELECT \* FROM comanda;

ALTER TABLE test\_comanda

DROP COLUMN livrare\_id;

DROP TABLE test\_comanda;

SELECT\* FROM history;



13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.

CREATE OR REPLACE PACKAGE proiect\_sgbd AS

PROCEDURE afisare\_produse\_comanda ( p\_produs\_id produs.produs\_id%TYPE);

PROCEDURE afisare\_clienti\_comanda( p\_livrare livrare.metoda\_livrare%TYPE, p\_pret comanda.pret\_total%TYPE);

FUNCTION afisare\_produse\_culoare (id culoare.culoare\_id%TYPE, categ\_ categorie.nume%type) RETURN NUMBER;

PROCEDURE afisare\_clienti\_comenzi\_(nume\_client client.nume%type,nume\_furnizor furnizor.nume%type);

END proiect\_sgbd;

/

CREATE OR REPLACE PACKAGE BODY proiect\_sgbd

AS

PROCEDURE afisare\_produse\_comanda ( p\_produs\_id produs.produs\_id%TYPE)

AS

TYPE clienti\_i IS TABLE OF client%ROWTYPE; -- tablou imbricat

clienti\_table clienti\_i := clienti\_i();

TYPE vector IS VARRAY(100) OF comanda%ROWTYPE; -- VARRAY

comanda\_table vector := vector();

numar NUMBER(10):= 0;

BEGIN

--numaram cate comenzi au produsul respectiv

SELECT COUNT(\*)

INTO numar

FROM produse\_comandate pc, inventar i,comanda c

WHERE i.inventar\_id = pc.inventar\_id and pc.inventar\_id=c.comanda\_id and i.produs\_id=p\_produs\_id;

comanda\_table.extend(numar+1);

--selectam datele despre o comanda care are produsul respectiv

SELECT distinct c.\*

BULK COLLECT INTO comanda\_table

FROM comanda c

JOIN produse\_comandate pc ON c.comanda\_id = pc.comanda\_id

JOIN inventar i ON pc.inventar\_id = i.inventar\_id

WHERE i.produs\_id = p\_produs\_id;

IF comanda\_table.COUNT=0 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista comenzi cu produsul respectiv!');

RETURN;

END IF;

--selectam doar clientii care au dat o comanda cu produsul respectiv

FOR i IN comanda\_table.FIRST .. comanda\_table.LAST LOOP

clienti\_table.EXTEND;

SELECT DISTINCT cl.\*

INTO clienti\_table(clienti\_table.LAST)

FROM client cl

WHERE cl.client\_id = comanda\_table(i).client\_id;

END LOOP;

--daca am gasit comenzi afisam produsele

IF comanda\_table.COUNT > 0 THEN

FOR i IN clienti\_table.FIRST .. clienti\_table.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Client ID: ' || clienti\_table(i).client\_id);

DBMS\_OUTPUT.PUT\_LINE(clienti\_table(i).nume ||' ' || clienti\_table(i).prenume);

DBMS\_OUTPUT.PUT\_LINE(clienti\_table(i).email);

DBMS\_OUTPUT.PUT\_LINE('----------------------------------------------------');

FOR j IN comanda\_table.FIRST .. comanda\_table.LAST LOOP

IF comanda\_table(j).client\_id=clienti\_table(i).client\_id THEN

DECLARE

TYPE produse\_i IS TABLE OF VARCHAR(100) INDEX BY PLS\_INTEGER; --tablou indexat

produse\_table produse\_i;

BEGIN

SELECT p.nume

BULK COLLECT INTO produse\_table

FROM produse\_comandate pc, inventar i, produs p

WHERE p.produs\_id=i.produs\_id and i.inventar\_id = pc.inventar\_id and pc.comanda\_id=comanda\_table(j).comanda\_id;

DBMS\_OUTPUT.PUT\_LINE(' Comanda ID: ' || comanda\_table(j).comanda\_id);

DBMS\_OUTPUT.PUT\_LINE('---------------------');

IF produse\_table.COUNT > 0 THEN

FOR k IN produse\_table.FIRST .. produse\_table.LAST LOOP

numar := numar + 1;

DBMS\_OUTPUT.PUT\_LINE(k || '. ' || produse\_table(k));

END LOOP;

END IF;

END;

END IF;

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END IF;

END afisare\_produse\_comanda;

PROCEDURE afisare\_clienti\_comanda( p\_livrare livrare.metoda\_livrare%TYPE, p\_pret comanda.pret\_total%TYPE) AS

--selectam numele si prenumele clientului avand id-ul de livrare

CURSOR clienti (id comanda.livrare\_id%TYPE) IS

SELECT c.nume || ' ' || c.prenume AS result

FROM client c

JOIN comanda co ON c.client\_id = co.client\_id

WHERE co.livrare\_id = id;

--selectam id-ul livrarilor care au ca metoda de livrare curier, si au ca pret\_total o suma mai mare de p\_pret

CURSOR livrari (p\_livrare livrare.metoda\_livrare%TYPE) IS

SELECT l.livrare\_id

FROM livrare l

JOIN comanda co on co.livrare\_id=l.livrare\_id

WHERE UPPER(l.metoda\_livrare) LIKE UPPER(p\_livrare) AND co.pret\_total >= p\_pret;

--selectam comenzile avand id-ul lor de livrare

CURSOR comenzi (id livrare.livrare\_id%TYPE ) IS

SELECT comanda\_id, pret\_total, metoda\_plata

FROM comanda

WHERE livrare\_id = id;

id comanda.comanda\_id%TYPE;

pret comanda.pret\_total%TYPE;

plata comanda.metoda\_plata%TYPE;

BEGIN

FOR livrari\_i in livrari(p\_livrare) LOOP

OPEN comenzi(livrari\_i.livrare\_id);

FETCH comenzi INTO id,pret,plata;

DBMS\_OUTPUT.PUT\_LINE('Comanda cu numarul ' || id);

DBMS\_OUTPUT.PUT\_LINE('------------------------');

DBMS\_OUTPUT.PUT\_LINE('Pret:' || pret);

DBMS\_OUTPUT.PUT\_LINE('Metoda plata:' || plata);

CLOSE comenzi;

FOR clienti\_i IN clienti(livrari\_i.livrare\_id) LOOP

DBMS\_OUTPUT.PUT\_LINE(clienti\_i.result);

END LOOP;

DBMS\_OUTPUT.NEW\_LINE;

DBMS\_OUTPUT.NEW\_LINE;

END LOOP;

END afisare\_clienti\_comanda;

FUNCTION afisare\_produse\_culoare(id culoare.culoare\_id%TYPE, categ\_ categorie.nume%type) RETURN NUMBER

IS

nr\_produse NUMBER;

TYPE tip\_tabel IS TABLE OF inventar%rowtype INDEX BY PLS\_INTEGER;

tabel tip\_tabel;

categ categorie.categorie\_id%type;

--exceptii

NEGATIVE\_NUMBER EXCEPTION;

NO\_DATA\_FOUND1 EXCEPTION;

NO\_DATA\_FOUND2 EXCEPTION;

BEGIN

IF id < 0 THEN -- codul culorii nu e valid

RAISE NEGATIVE\_NUMBER;

END IF;

--selectam toate produsele din inventar care sunt pe culoarea respectiva

SELECT \*

BULK COLLECT INTO tabel

FROM inventar

WHERE culoare\_id = id;

--avem mesaj de eroare in cazul in care nu le gasim

IF SQL%NOTFOUND THEN

RAISE NO\_DATA\_FOUND1;

END IF;

--cautam id-ul pentru 'imbracaminte'

SELECT categorie\_id

INTO categ

FROM categorie

WHERE UPPER(nume) LIKE UPPER(categ\_);

--cautam numarul de produse cu respectivele conditii

SELECT COUNT(i.inventar\_id)

INTO nr\_produse

FROM produse\_comandate pc

JOIN inventar i ON (i.inventar\_id = pc.inventar\_id)

JOIN produs p on (p.produs\_id=i.produs\_id)

WHERE i.culoare\_id = id and p.categorie\_id=categ;

--eroare in cazul in care nu exista produse

IF nr\_produse = 0 THEN

RAISE NO\_DATA\_FOUND2;

END IF;

RETURN nr\_produse;

EXCEPTION

WHEN NEGATIVE\_NUMBER THEN

DBMS\_OUTPUT.PUT\_LINE('Codul culorii nu poate sa fie negativ!');

RETURN -1;

WHEN NO\_DATA\_FOUND1 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produse pe culoarea cu codul ' || id);

RETURN -1;

WHEN NO\_DATA\_FOUND2 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista produse de imbracaminte cumparate cu acea culoare!');

RETURN -1;

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Codul de eroare: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE('Mesajul erorii: ' || SQLERRM);

RETURN -1;

END afisare\_produse\_culoare;

PROCEDURE afisare\_clienti\_comenzi\_(nume\_client client.nume%type,nume\_furnizor furnizor.nume%type)

AS

TYPE tabel\_index IS TABLE OF comanda.comanda\_id%type INDEX BY PLS\_INTEGER;

comenzi tabel\_index;

TYPE tabel\_index\_clienti IS TABLE OF client%rowtype INDEX BY PLS\_INTEGER;

clienti tabel\_index\_clienti;

NO\_DATA\_FOUND1 EXCEPTION;

NO\_DATA\_FOUND2 EXCEPTION;

TOO\_MANY\_ROWS1 EXCEPTION;

BEGIN

--selectam clientii cu numele respectiv

SELECT \*

BULK COLLECT INTO clienti

FROM client

WHERE UPPER(nume) = UPPER(nume\_client);

--eroare daca nu il gasim

IF SQL%NOTFOUND THEN

RAISE NO\_DATA\_FOUND1;

END IF;

--eroare daca avem mai multi clienti cu numele respectiv

IF clienti.count >= 2 THEN

RAISE TOO\_MANY\_ROWS1;

END IF;

--selectam id-ul comenzii cu proprietatile respective

SELECT DISTINCT c.comanda\_id

BULK COLLECT INTO comenzi

FROM comanda c

JOIN produse\_comandate pc ON pc.comanda\_id = c.comanda\_id

JOIN inventar i ON i.inventar\_id = pc.inventar\_id

JOIN client cl ON (cl.client\_id = c.client\_id)

JOIN produs p ON p.produs\_id=i.produs\_id

JOIN furnizor f ON f.furnizor\_id=p.furnizor\_id

WHERE UPPER(f.nume) = UPPER(nume\_furnizor) and UPPER(cl.nume)=UPPER(nume\_client);

IF comenzi.count = 0 THEN

RAISE NO\_DATA\_FOUND2;

END IF;

DBMS\_OUTPUT.PUT\_LINE('Comenzile clientului cu numele ' || nume\_client || ' care contin produse de la furnizorul ' || nume\_furnizor|| ' sunt: ');

FOR i IN comenzi.FIRST..comenzi.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE(comenzi(i));

END LOOP;

EXCEPTION

WHEN NO\_DATA\_FOUND1 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista clienti care sa aiba acest nume de familie!');

WHEN NO\_DATA\_FOUND2 THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista comenzi care sa aiba produse ce provin de la acel furnizor!');

WHEN TOO\_MANY\_ROWS1 THEN

DBMS\_OUTPUT.PUT\_LINE('Exista mai mult de un client cu acel nume!');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Codul de eroare: ' || SQLCODE);

DBMS\_OUTPUT.PUT\_LINE('Mesajul erorii: ' || SQLERRM);

END afisare\_clienti\_comenzi\_;

END proiect\_sgbd;

/

-- testare package

EXECUTE proiect\_sgbd.afisare\_produse\_comanda(131);

EXECUTE proiect\_sgbd.afisare\_clienti\_comanda('curier', 200);

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Rezultatul pentru culoarea data este: ' || proiect\_sgbd.afisare\_produse\_culoare(11,'imbracaminte'));

END;

/

EXECUTE proiect\_sgbd.afisare\_clienti\_comenzi\_('horga','XYZ Supplier Ltd.');

