***APLICAȚIE MAGAZIN***

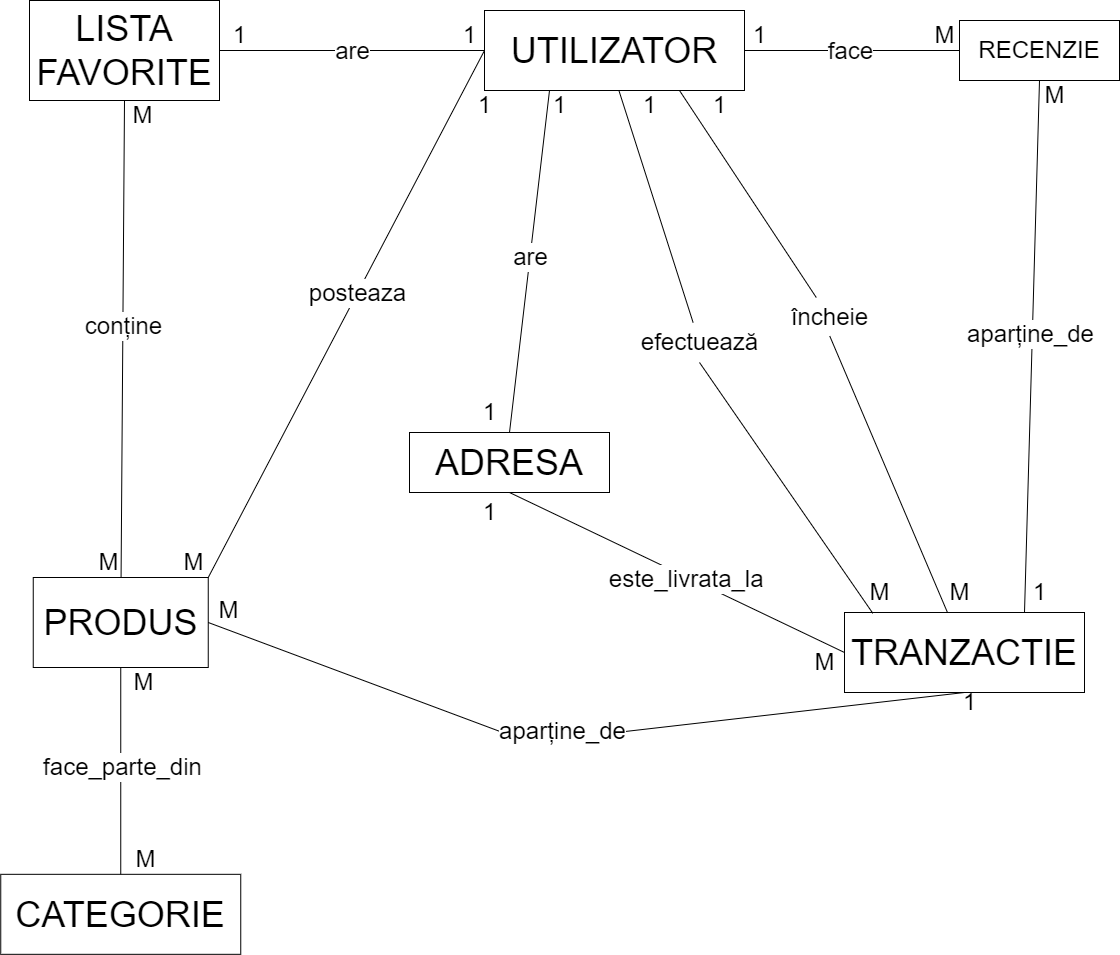
***SECOND-HAND***

***MANIA ELENA – GRUPA 242***

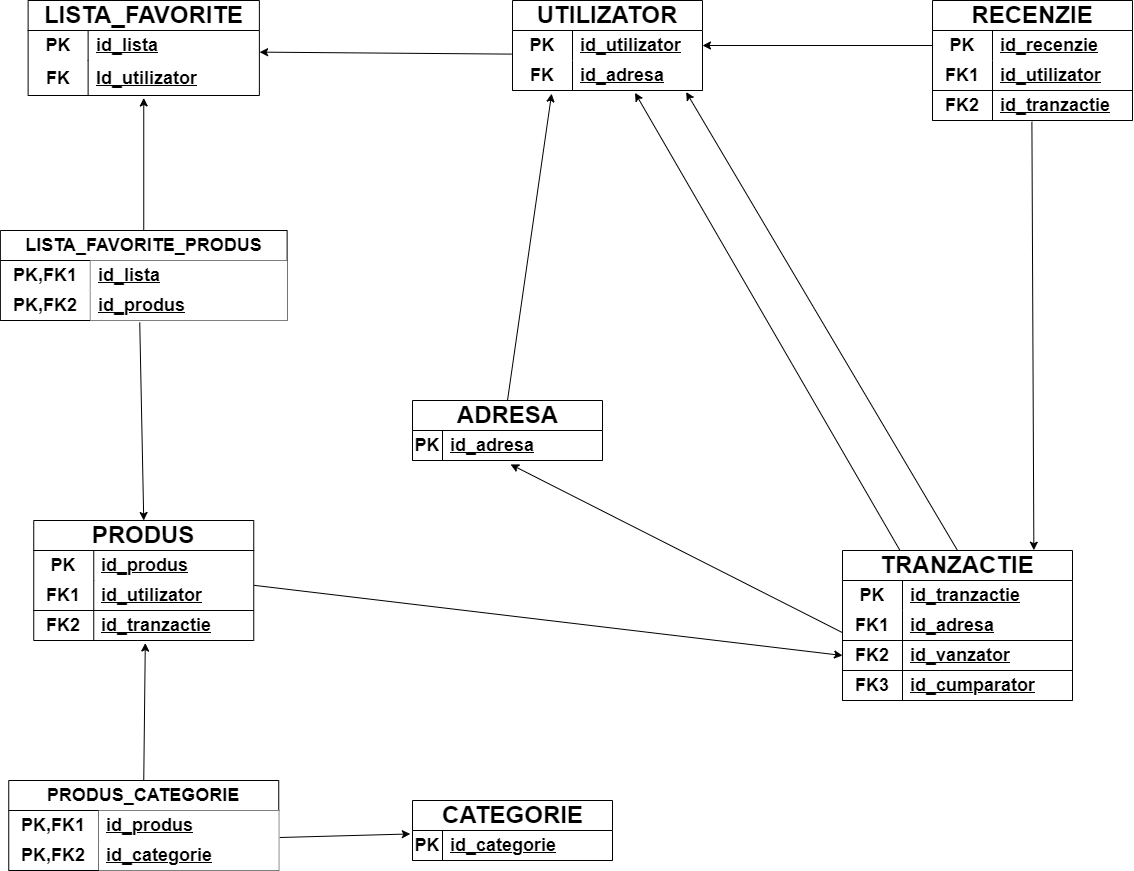
1. Prezentați pe scurt baza de date (utilitatea ei).

Prin intermediul proiectului implementez o bază de date care este concepută pentru a gestiona la nivel conceptual un magazin virtual axat pe vânzarea de haine second-hand. Prin intermediul acestei aplicații utilizatorii au posibilitatea de a vinde și de a cumpăra produse, de a-și crea o listă de produse favorite și de a lăsa recenzii în urma tranzacțiilor efectuate. Recenziile pot fi lăsate de către ambele părți implicate într-o tranzacție, fiind evaluate atât produsele cât și experiența de cumpărare/vânzare cu scopul de a îmbunătăți experiența și nivelul de satisfacție al utilizatorilor. De asemenea, produsele pot fi categorisite în funcție de mai multe criterii, facilitând navigarea și căutarea produselor după preferințe cât mai specifice. Așadar, utilitatea constă în punerea la dispoziție a unui mediu care încurajează sustenabilitatea.

2. Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română (vezi curs SGBD / model de diagrama ERD; nu se va accepta alt format).



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 (chei primare, cheile externe etc).

CREATE TABLE ADRESA (

id\_adresa NUMBER(10) PRIMARY KEY,

tara VARCHAR(50),

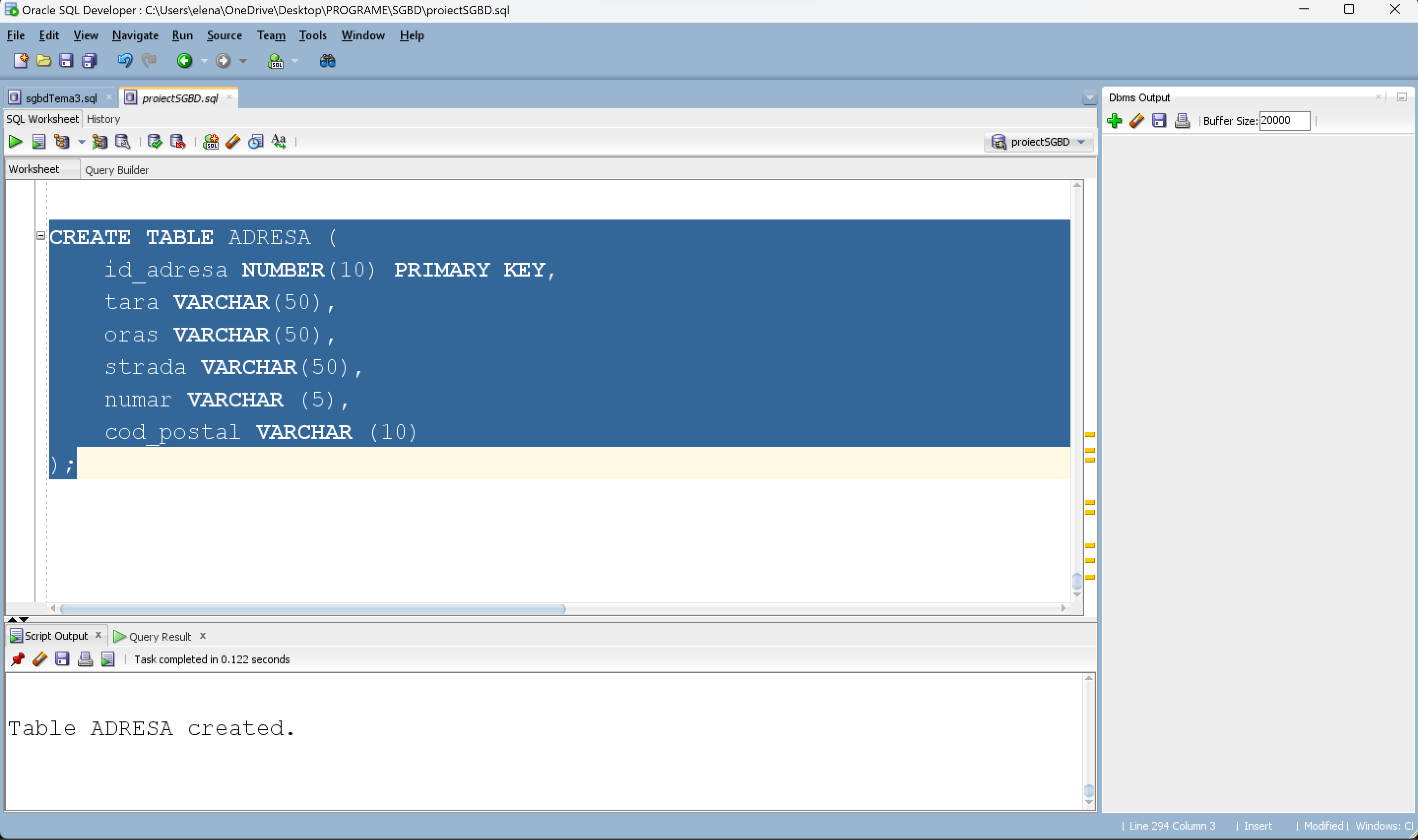
oras VARCHAR(50),

strada VARCHAR(50),

numar VARCHAR (5),

cod\_postal VARCHAR (10)

);



CREATE TABLE UTILIZATOR (

id\_utilizator NUMBER(5) PRIMARY KEY,

nume VARCHAR(50),

prenume VARCHAR(50),

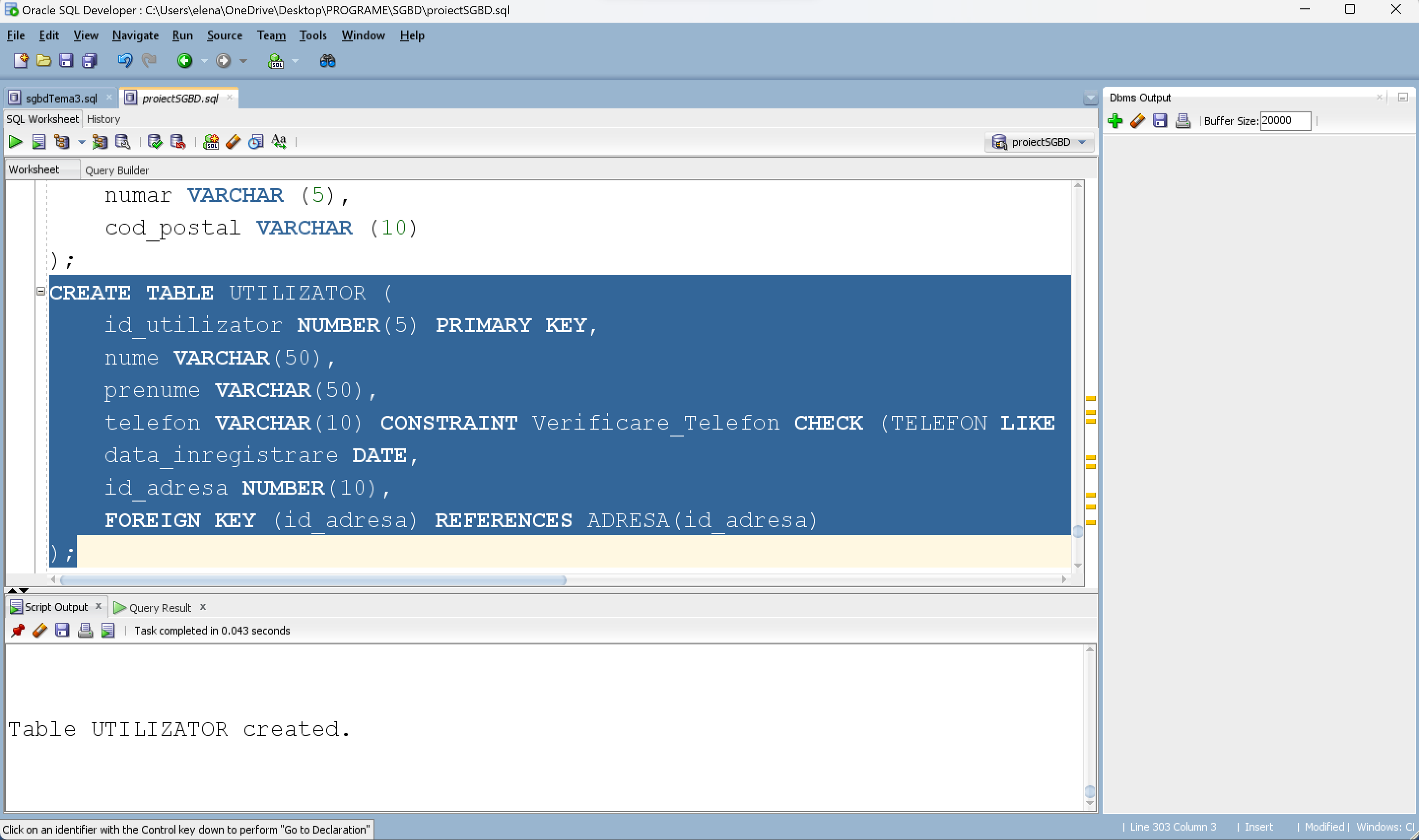
telefon VARCHAR(10) CONSTRAINT Verificare\_Telefon CHECK (TELEFON LIKE '07\_\_\_\_\_\_\_\_'),

data\_inregistrare DATE,

id\_adresa NUMBER(10),

FOREIGN KEY (id\_adresa) REFERENCES ADRESA(id\_adresa)

);



CREATE TABLE TRANZACTIE(

id\_tranzactie NUMBER(5) PRIMARY KEY,

id\_adresa NUMBER(10),

FOREIGN KEY (id\_adresa) REFERENCES ADRESA(id\_adresa),

id\_vanzator NUMBER(5),

FOREIGN KEY (id\_vanzator) REFERENCES UTILIZATOR(id\_utilizator),

id\_cumparator NUMBER(5),

FOREIGN KEY (id\_cumparator) REFERENCES UTILIZATOR(id\_utilizator),

CONSTRAINT Check\_Different\_Ids CHECK (id\_vanzator <> id\_cumparator),

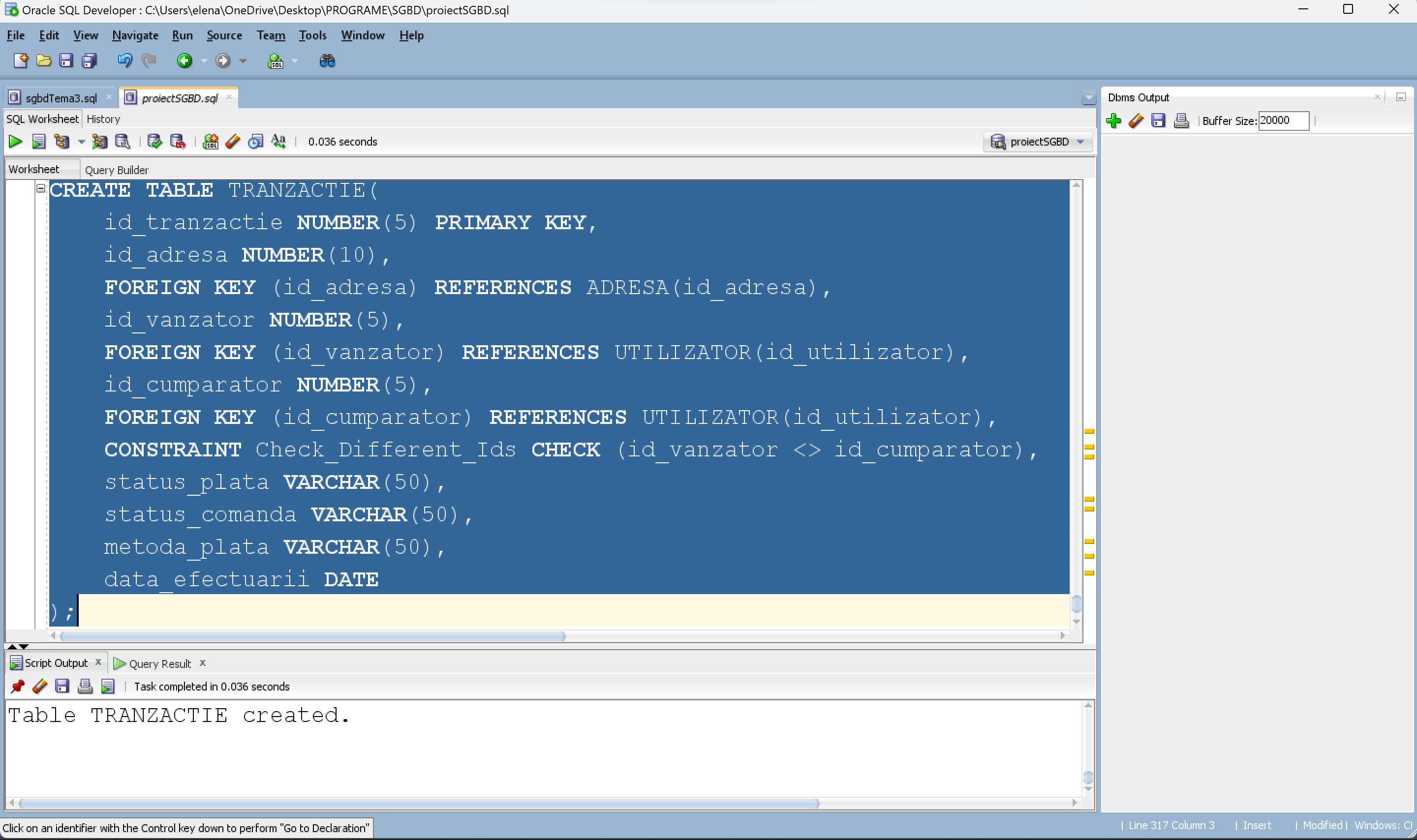
status\_plata VARCHAR(50),

status\_comanda VARCHAR(50),

metoda\_plata VARCHAR(50),

data\_efectuarii DATE

);



CREATE TABLE PRODUS(

id\_produs NUMBER(5) PRIMARY KEY,

pret INT,

marime VARCHAR(10),

culoare VARCHAR(50),

firma VARCHAR(50),

calitate VARCHAR(50),

material VARCHAR(50),

sex VARCHAR(50) CHECK (sex IN ('masculin', 'feminin', 'unisex')),

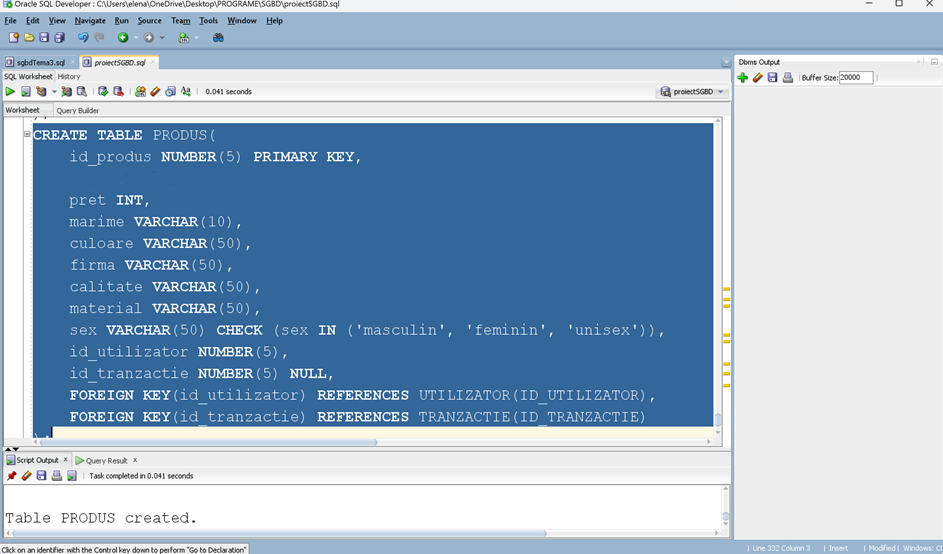
id\_utilizator NUMBER(5),

id\_tranzactie NUMBER(5) NULL,

FOREIGN KEY(id\_utilizator) REFERENCES UTILIZATOR(ID\_UTILIZATOR),

FOREIGN KEY(id\_tranzactie) REFERENCES TRANZACTIE(ID\_TRANZACTIE)

);

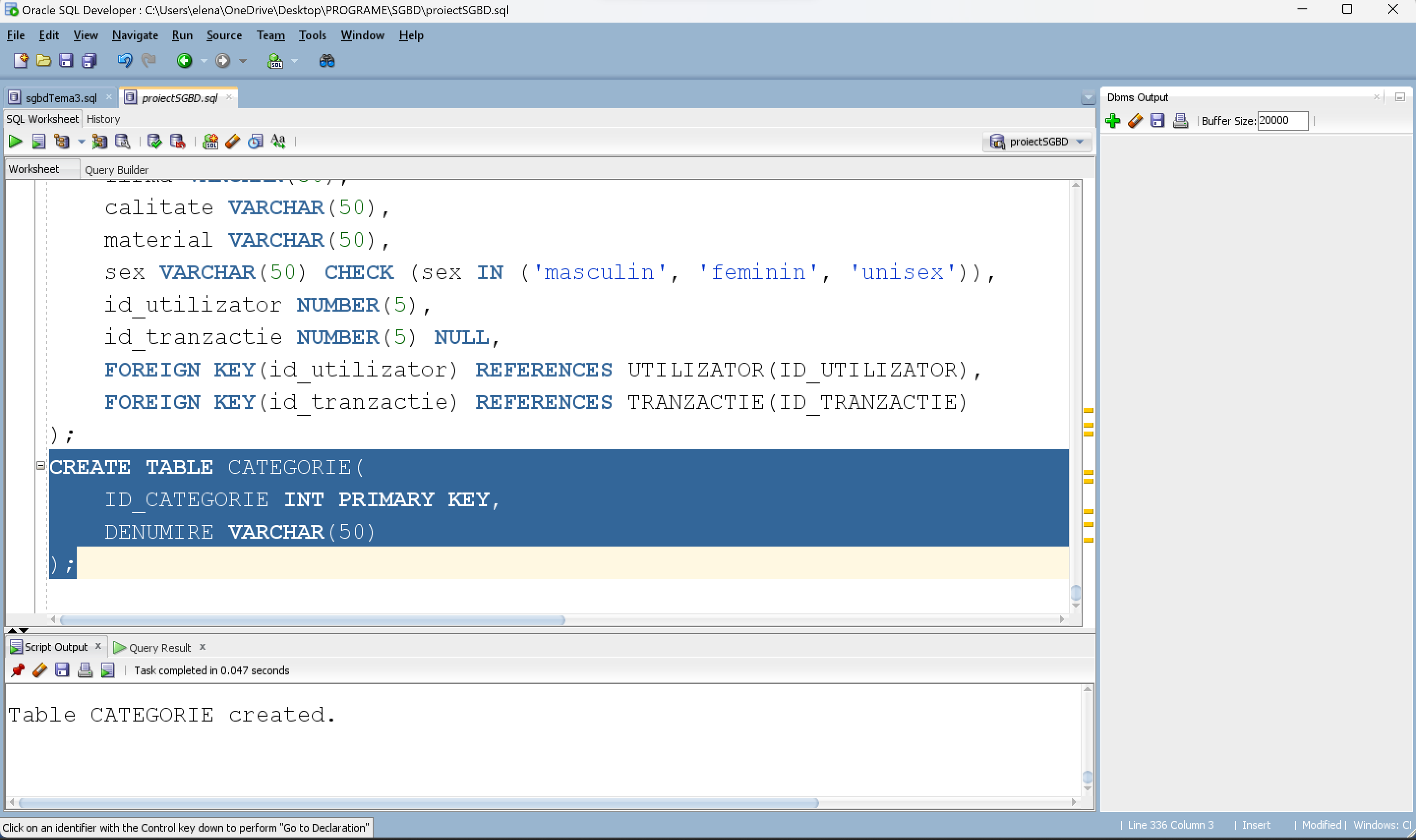


CREATE TABLE CATEGORIE(

ID\_CATEGORIE INT PRIMARY KEY,

DENUMIRE VARCHAR(50)

);



CREATE TABLE RECENZIE(

id\_recenzie NUMBER(5) PRIMARY KEY,

evaluare\_produs VARCHAR(50),

evaluare\_relationare\_utilizatori VARCHAR(100),

rating NUMBER(5) CHECK (rating >= 1 AND rating <= 5),

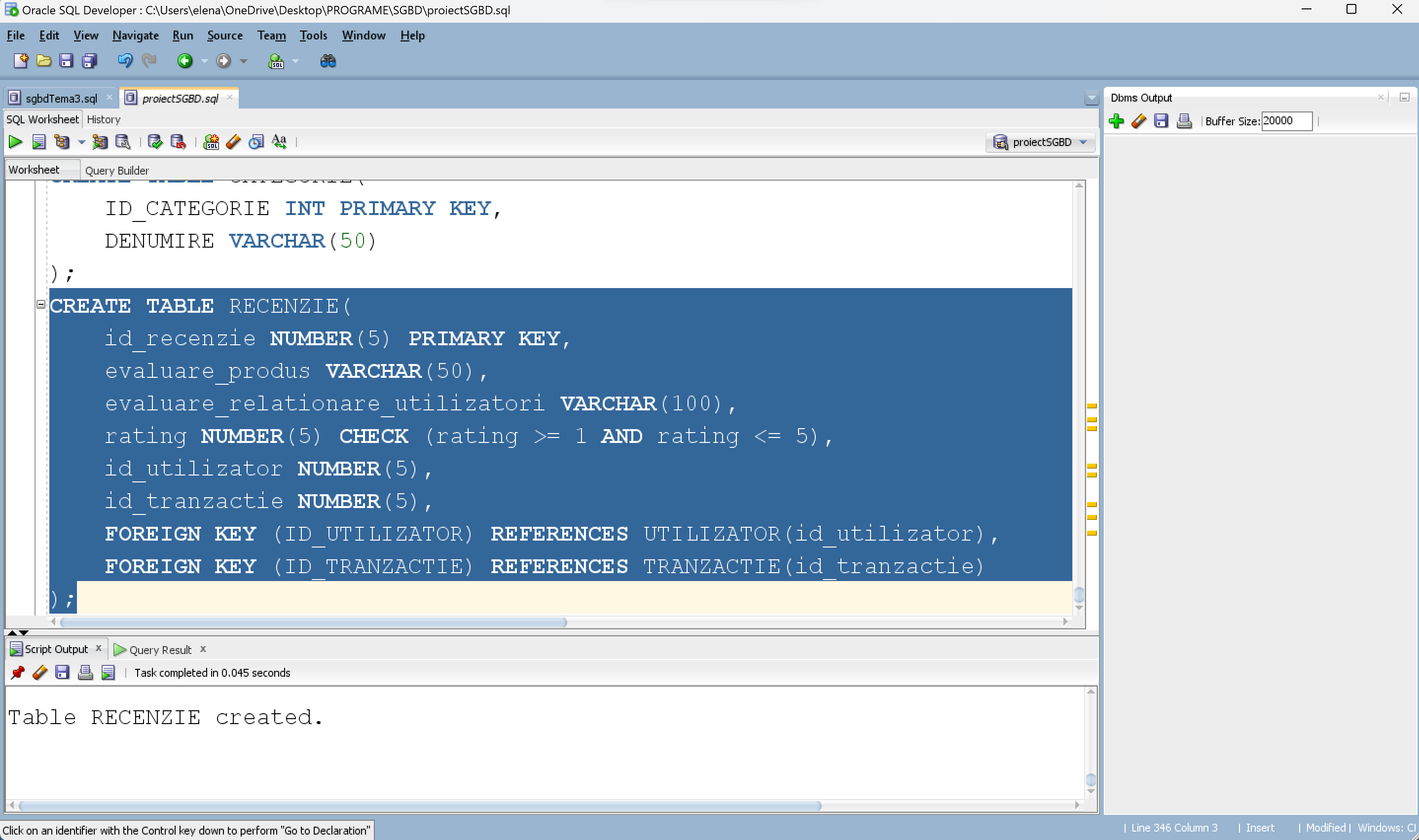
id\_utilizator NUMBER(5),

id\_tranzactie NUMBER(5),

FOREIGN KEY (ID\_UTILIZATOR) REFERENCES UTILIZATOR(id\_utilizator),

FOREIGN KEY (ID\_TRANZACTIE) REFERENCES TRANZACTIE(id\_tranzactie)

);



CREATE TABLE LISTA\_FAVORITE(

id\_lista NUMBER(5) PRIMARY KEY,

id\_utilizator NUMBER(5),

status VARCHAR(10) CHECK ( status IN ('arhivat','sters','activ')),

data\_creare DATE,

FOREIGN KEY (ID\_UTILIZATOR) REFERENCES UTILIZATOR(id\_utilizator)

);



CREATE TABLE CATEGORIE\_PRODUS(

id\_categorie INT,

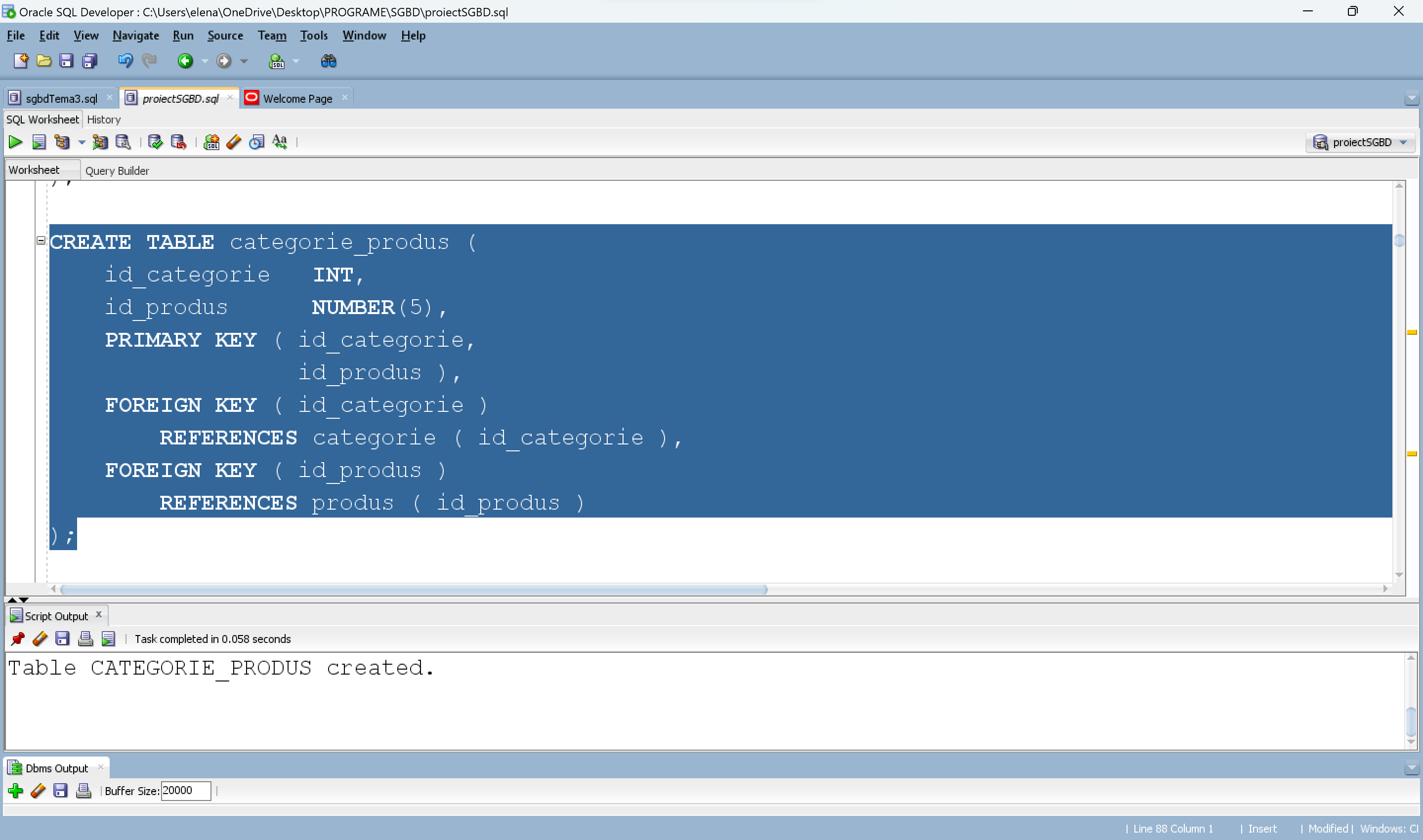
id\_produs NUMBER(5),

PRIMARY KEY(ID\_CATEGORIE,ID\_PRODUS),

FOREIGN KEY (ID\_CATEGORIE) REFERENCES CATEGORIE(ID\_CATEGORIE),

FOREIGN KEY (ID\_PRODUS) REFERENCES PRODUS(ID\_PRODUS)

);



CREATE TABLE LISTA\_FAVORITE\_PRODUS(

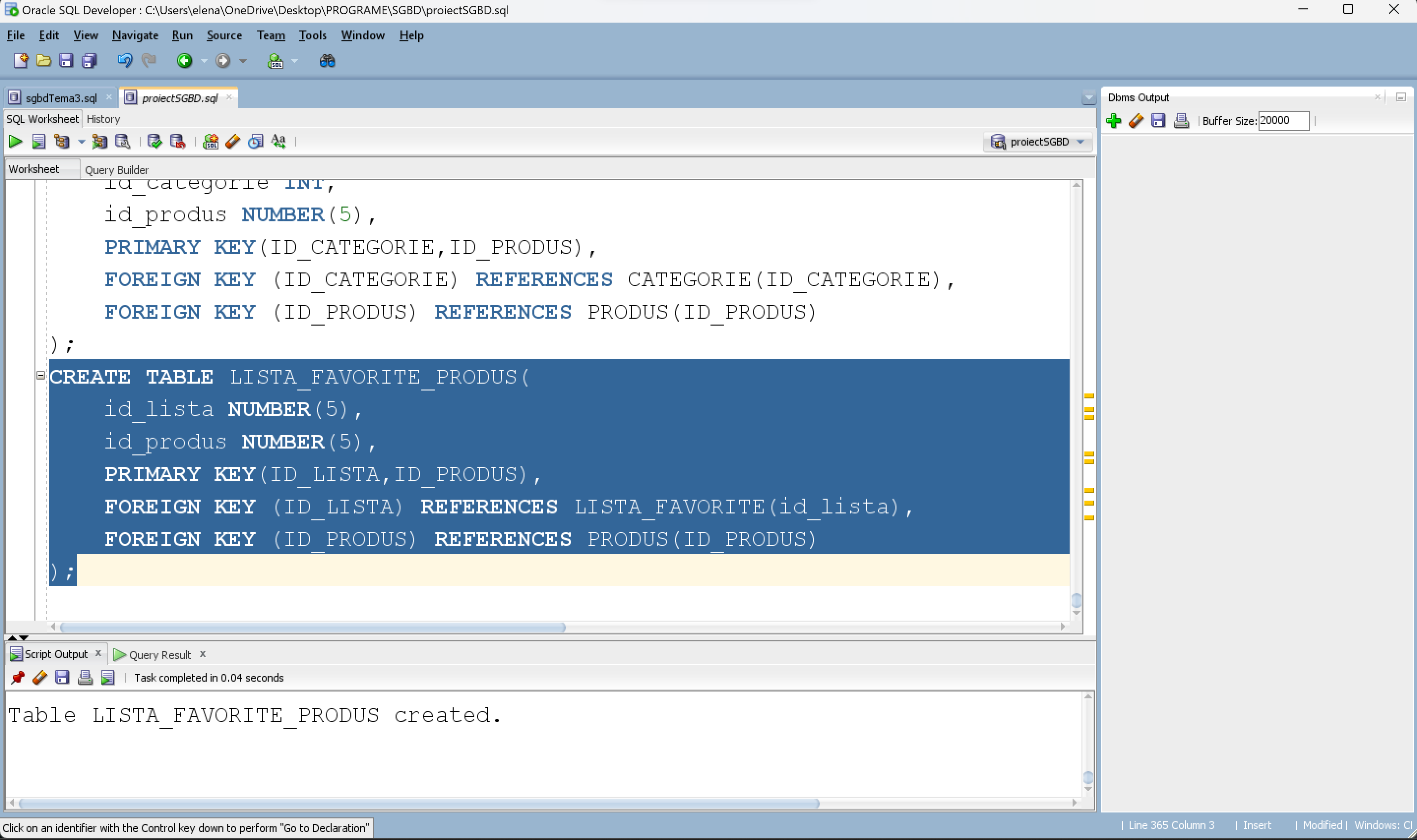
id\_lista NUMBER(5),

id\_produs NUMBER(5),

PRIMARY KEY(ID\_LISTA,ID\_PRODUS),

FOREIGN KEY (ID\_LISTA) REFERENCES LISTA\_FAVORITE(id\_lista),

FOREIGN KEY (ID\_PRODUS) REFERENCES PRODUS(ID\_PRODUS));



5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

INSERT INTO ADRESA VALUES (10,'Romania','Roman','Stefan cel Mare','26','611038');

INSERT INTO ADRESA VALUES (11,'Romania','Bucuresti','Pictor Arthur Verona','12','010314');

INSERT INTO ADRESA VALUES (12,'Romania','Bucuresti','Fizicienilor','6','032064');

INSERT INTO ADRESA VALUES (13,'Romania','Bucuresti','Masina de Paine','37','021125');

INSERT INTO ADRESA VALUES (14,'Romania','Bucuresti','Lirei','1','021421');

INSERT INTO ADRESA VALUES (15,'Romania','Braila','Pacurari','54','700524');

INSERT INTO ADRESA VALUES (16,'Romania','Husi','Schit','3','735100');

INSERT INTO ADRESA VALUES (17,'Romania','Oradea','Horea','10','410080');

INSERT INTO ADRESA VALUES (18,'Romania','Constanta','Decebal','15','900669');

INSERT INTO ADRESA VALUES (19,'Romania','Timisoara','Magura','4','300763');

INSERT INTO ADRESA VALUES (20,'Romania','Sibiu','Independentei','1','550173');

INSERT INTO ADRESA VALUES (21,'Romania','Iasi','Tanasescu','33','700030');

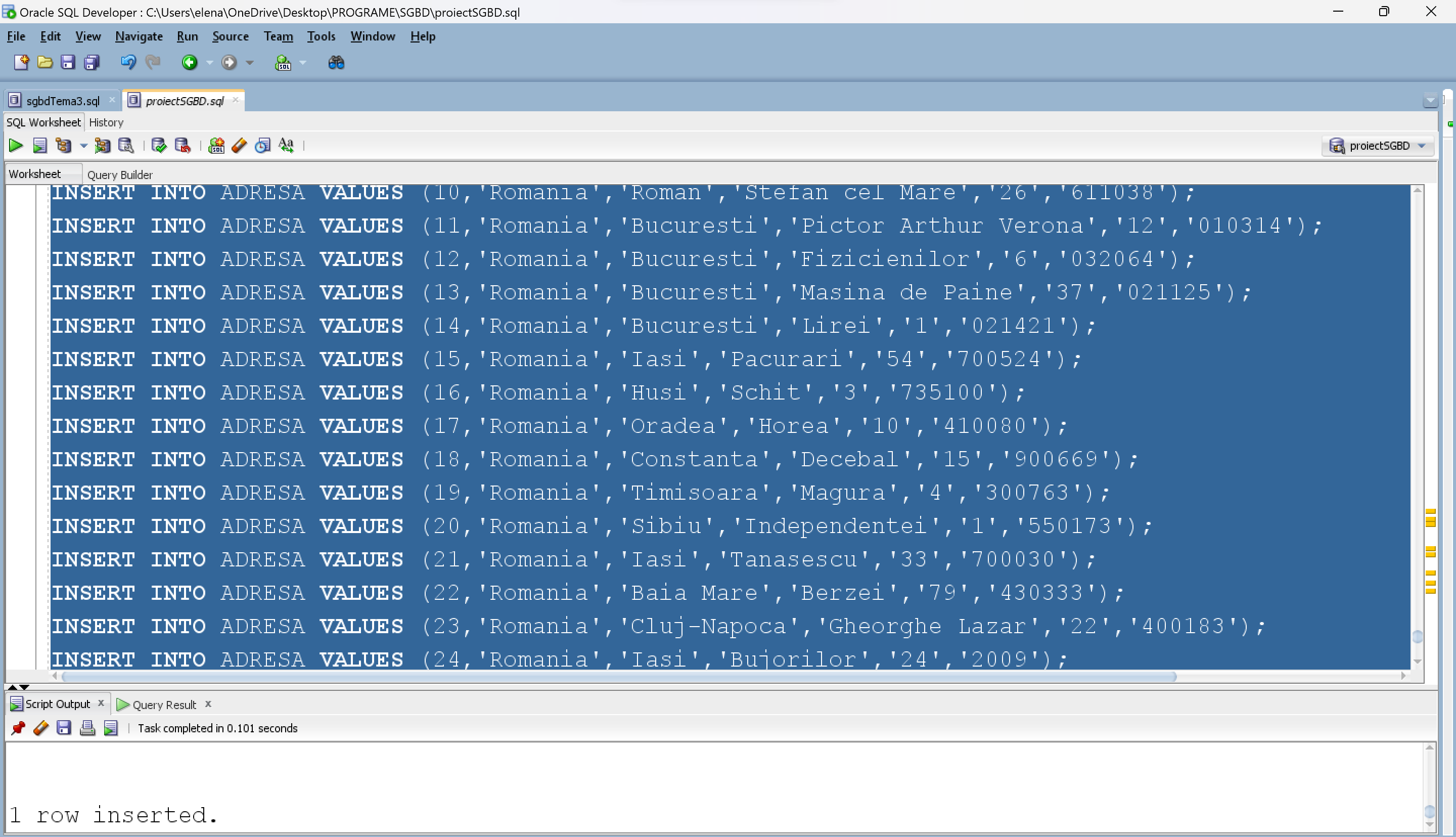
INSERT INTO ADRESA VALUES (22,'Romania','Baia Mare','Berzei','79','430333');

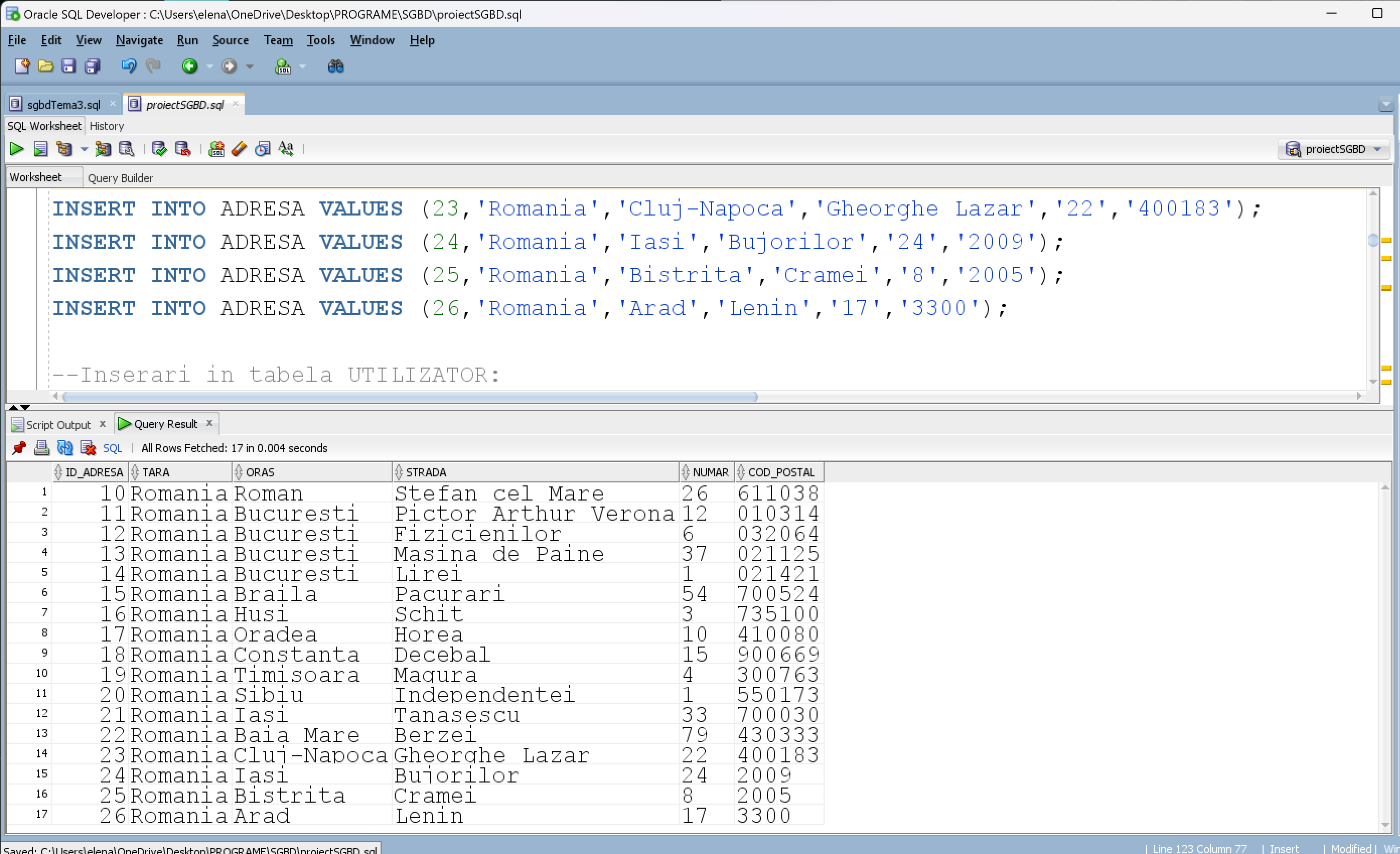
INSERT INTO ADRESA VALUES (23,'Romania','Cluj-Napoca','Gheorghe Lazar','22','400183');

INSERT INTO ADRESA VALUES (24,'Romania','Iasi,'Bujorilor','24','2009');

INSERT INTO ADRESA VALUES (25,'Romania','Bistrita','Cramei','8','2005');

INSERT INTO ADRESA VALUES (26,'Romania','Arad','Lenin','17','3300');





INSERT INTO UTILIZATOR VALUES (10006, 'Popescu','Andreea','0712345678',TO\_DATE('2023-06-01','YYYY-MM-DD'),11);

INSERT INTO UTILIZATOR VALUES (10002, 'Andreescu','Ion','0712346678',TO\_DATE('2021-10-12','YYYY-MM-DD'),12);

INSERT INTO UTILIZATOR VALUES (10003, 'Calinescu','Marina','0745645678',TO\_DATE('2022-03-12','YYYY-MM-DD'),15);

INSERT INTO UTILIZATOR VALUES (10001, 'Popescu','Andreea','0716377778', TO\_DATE('2020-05-15','YYYY-MM-DD'),22);

INSERT INTO UTILIZATOR VALUES (10004, 'Vasilache','Cristiana','0794365678', TO\_DATE('2022-06-11','YYYY-MM-DD'),18);

INSERT INTO UTILIZATOR VALUES (10007, 'Copaceanu','Mihai','0744045078',TO\_DATE('2021-09-24','YYYY-MM-DD'),23);

INSERT INTO UTILIZATOR VALUES (10008, 'Iliescu', 'Maria','0742066079', TO\_DATE('2023-12-04','YYYY-MM-DD'),10);

INSERT INTO UTILIZATOR VALUES (10005, 'Marinescu','Iulian','0745045076', TO\_DATE('2023-03-24','YYYY-MM-DD'),16);

INSERT INTO UTILIZATOR VALUES (10009, 'Romascu', 'Sergiu', '0767380297', TO\_DATE('2023-06-25','YYYY-MM-DD'),13);

INSERT INTO UTILIZATOR VALUES (10010, 'Gosav','Doina','0765432780',TO\_DATE('2023-06-25','YYYY-MM-DD'),14);

INSERT INTO UTILIZATOR VALUES (10011, 'Oprea','Alexandru','0794365321',TO\_DATE('2021-09-20','YYYY-MM-DD'),17);

INSERT INTO UTILIZATOR VALUES (10012, 'Caliniuc', 'Ramona','0712040608',TO\_DATE('2021-10-12','YYYY-MM-DD'),19);

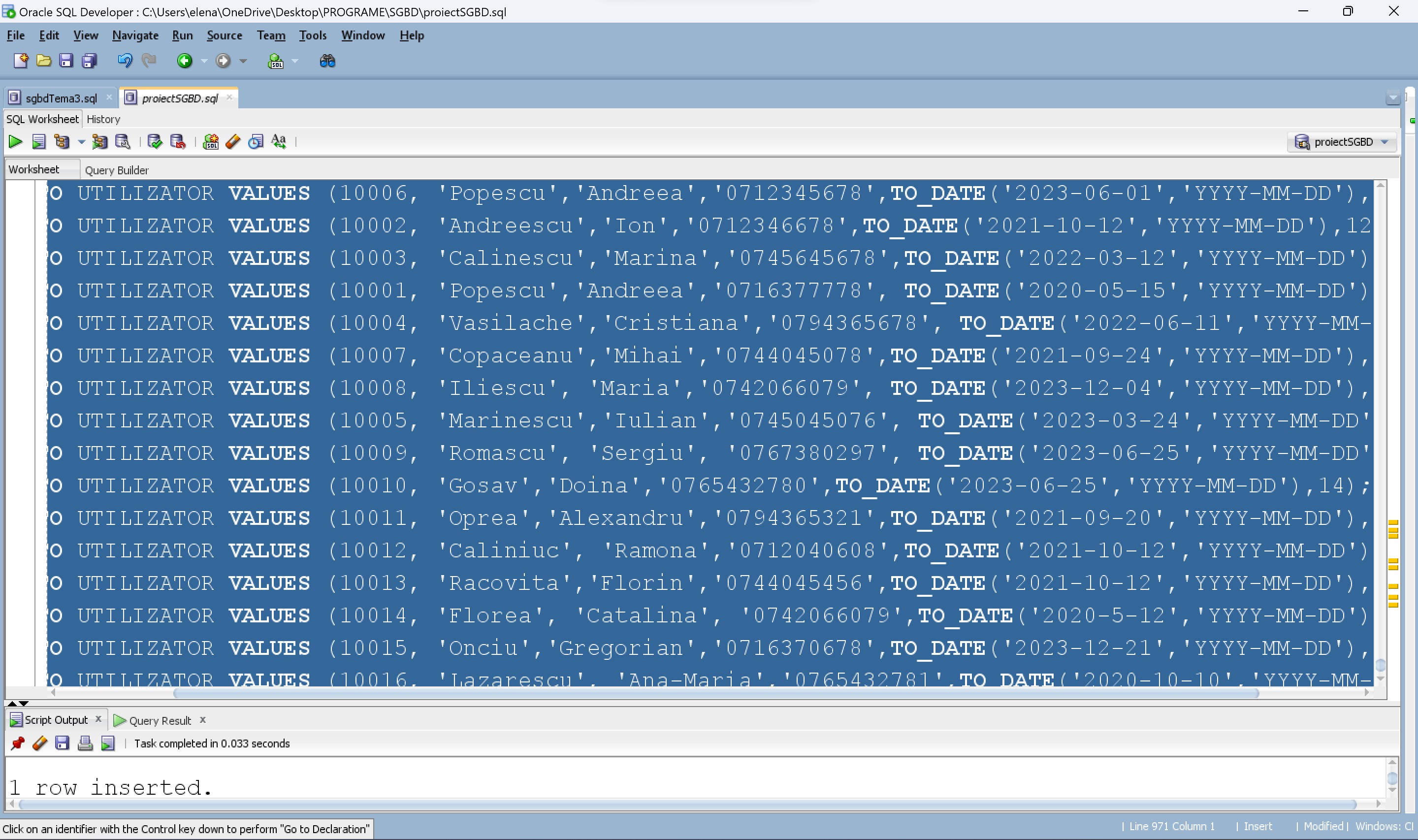
INSERT INTO UTILIZATOR VALUES (10013, 'Racovita','Florin','0744045456',TO\_DATE('2021-10-12','YYYY-MM-DD'),20);

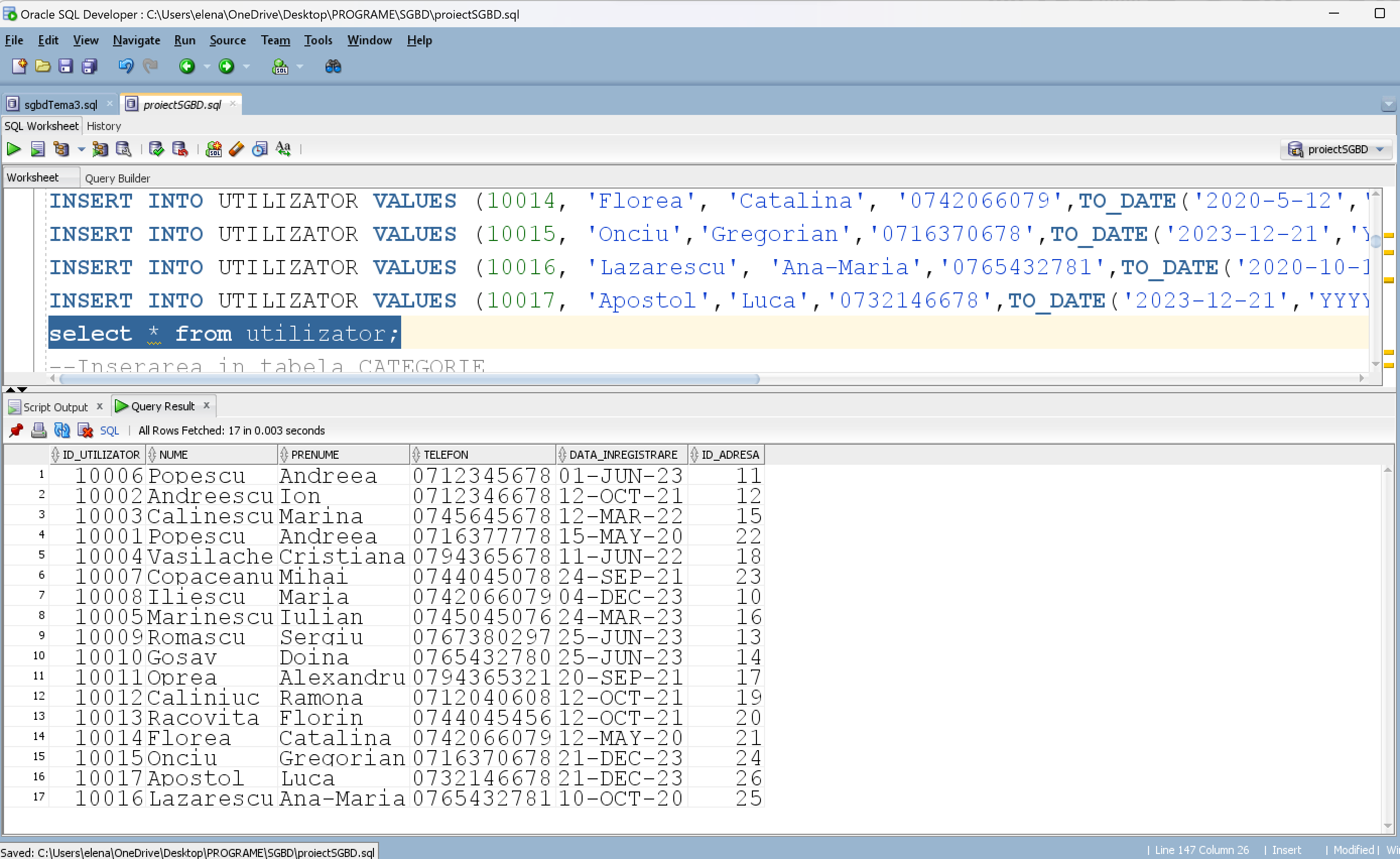
INSERT INTO UTILIZATOR VALUES (10014, 'Florea', 'Catalina', '0742066079',TO\_DATE('2020-5-12','YYYY-MM-DD'), 21);

INSERT INTO UTILIZATOR VALUES (10015, 'Onciu','Gregorian','0716370678',TO\_DATE('2023-12-21','YYYY-MM-DD'),24);

INSERT INTO UTILIZATOR VALUES (10016, 'Lazarescu', 'Ana-Maria','0765432781',TO\_DATE('2020-10-10','YYYY-MM-DD'),25);

INSERT INTO UTILIZATOR VALUES (10017, 'Apostol','Luca','0732146678',TO\_DATE('2023-12-21','YYYY-MM-DD'),26);





INSERT INTO CATEGORIE VALUES (1,'Imbracaminte exterioara');

INSERT INTO CATEGORIE VALUES (2,'Palton');

INSERT INTO CATEGORIE VALUES (3,'Geaca');

INSERT INTO CATEGORIE VALUES (4,'Pelerina');

INSERT INTO CATEGORIE VALUES (5,'Jacheta');

INSERT INTO CATEGORIE VALUES (6,'Pulover');

INSERT INTO CATEGORIE VALUES (7,'Hanorac');

INSERT INTO CATEGORIE VALUES (8,'Bluze');

INSERT INTO CATEGORIE VALUES (9,'Tricou');

INSERT INTO CATEGORIE VALUES (10,'Maleta');

INSERT INTO CATEGORIE VALUES (11,'Camasa');

INSERT INTO CATEGORIE VALUES (12,'Sacou');

INSERT INTO CATEGORIE VALUES (13,'Fusta');

INSERT INTO CATEGORIE VALUES (14,'Rochie');

INSERT INTO CATEGORIE VALUES (15,'Pantaloni');

INSERT INTO CATEGORIE VALUES (16,'Incaltaminte');

INSERT INTO CATEGORIE VALUES (17,'Imbracaminte Sport');

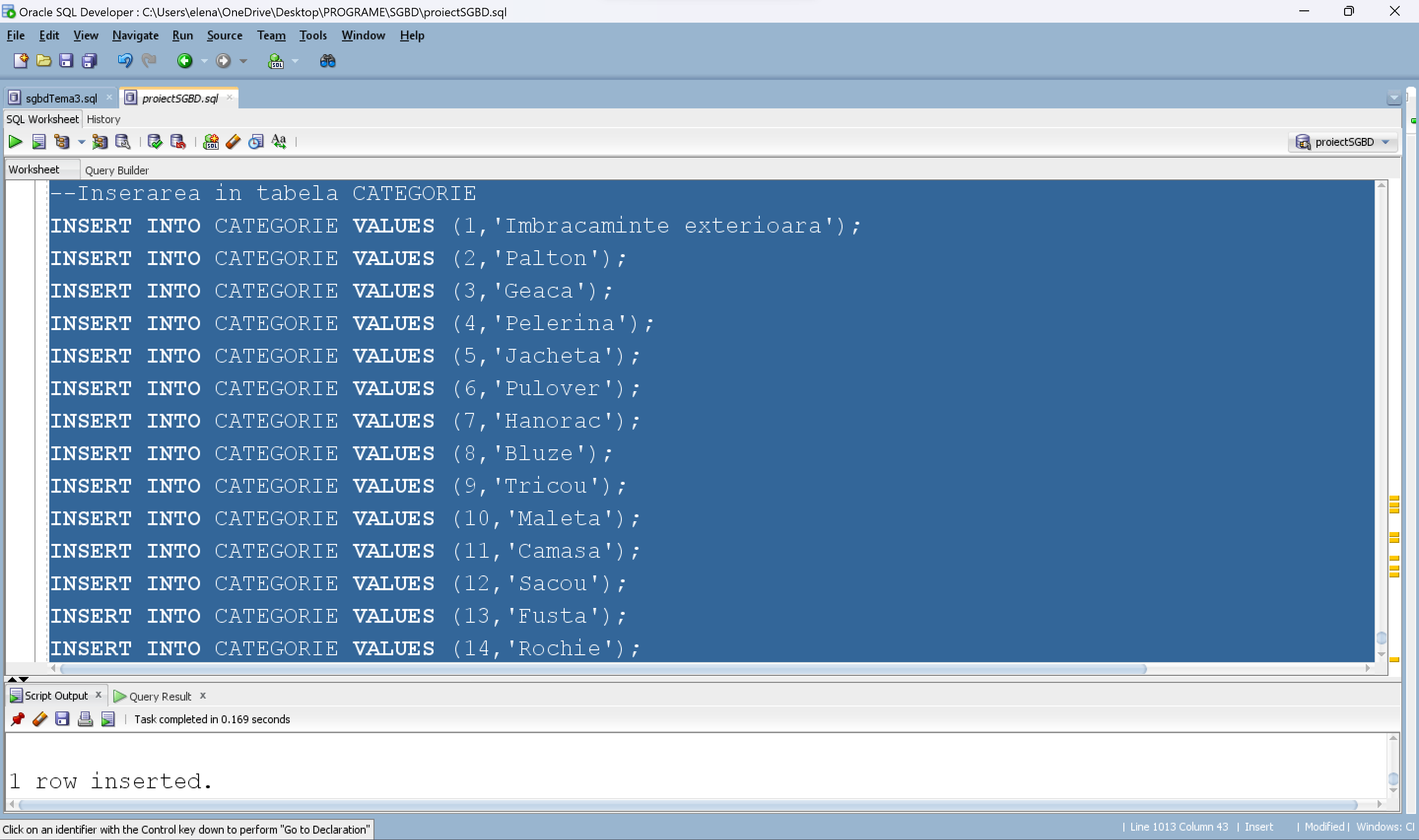
INSERT INTO CATEGORIE VALUES (18,'Imbracaminte Formala');

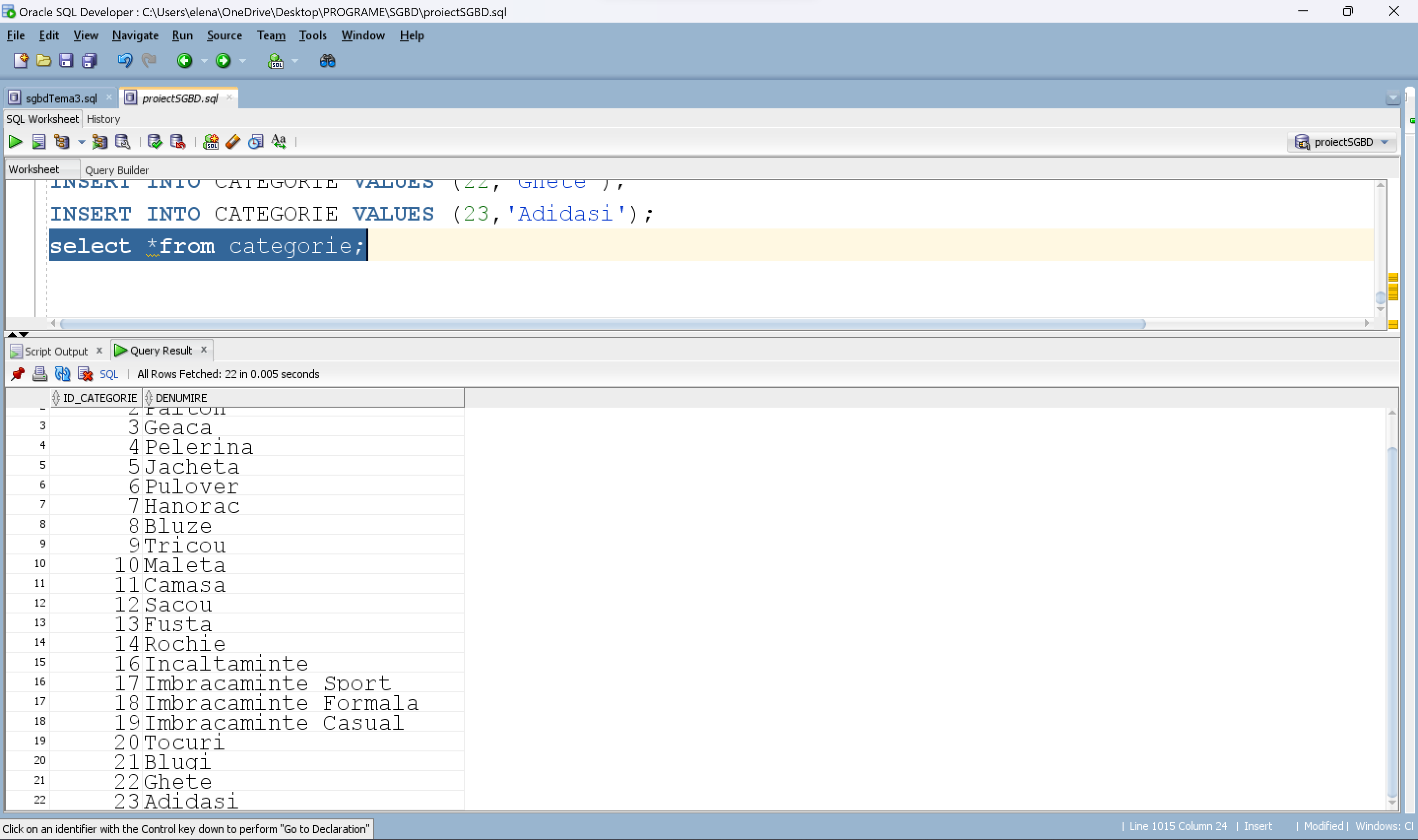
INSERT INTO CATEGORIE VALUES (19,'Imbracaminte Casual');

INSERT INTO CATEGORIE VALUES (20,'Tocuri');

INSERT INTO CATEGORIE VALUES (21,'Blugi');

INSERT INTO CATEGORIE VALUES (22,'Ghete');

INSERT INTO CATEGORIE VALUES (23,'Adidasi'); 



INSERT INTO TRANZACTIE VALUES (12345,18,10003,10004,'Platit', 'Livrat', 'Card',TO\_DATE('2022-11-20','YYYY-MM-DD'));

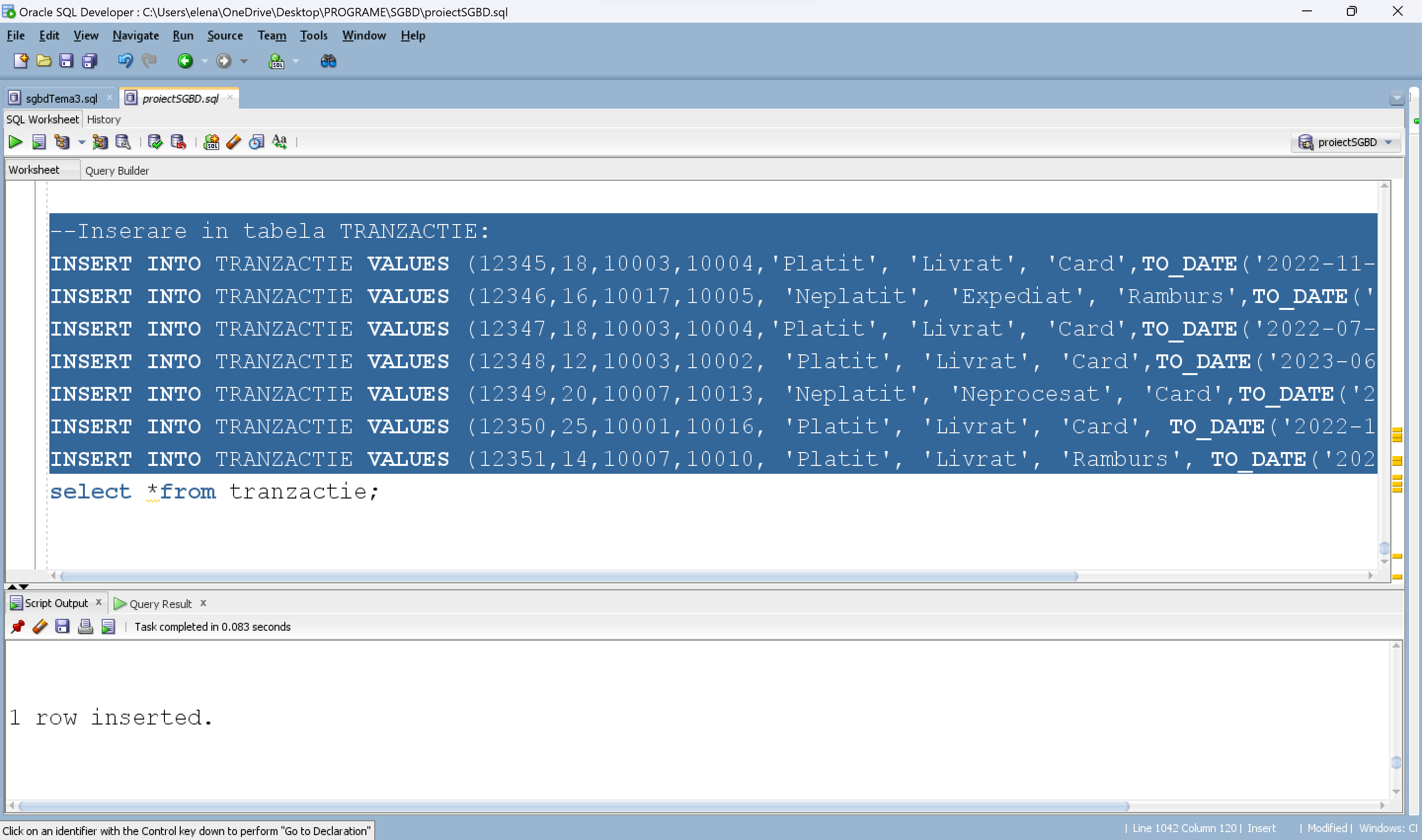
INSERT INTO TRANZACTIE VALUES (12346,16,10017,10005, 'Neplatit', 'Expediat', 'Ramburs',TO\_DATE('2023-12-30','YYYY-MM-DD'));

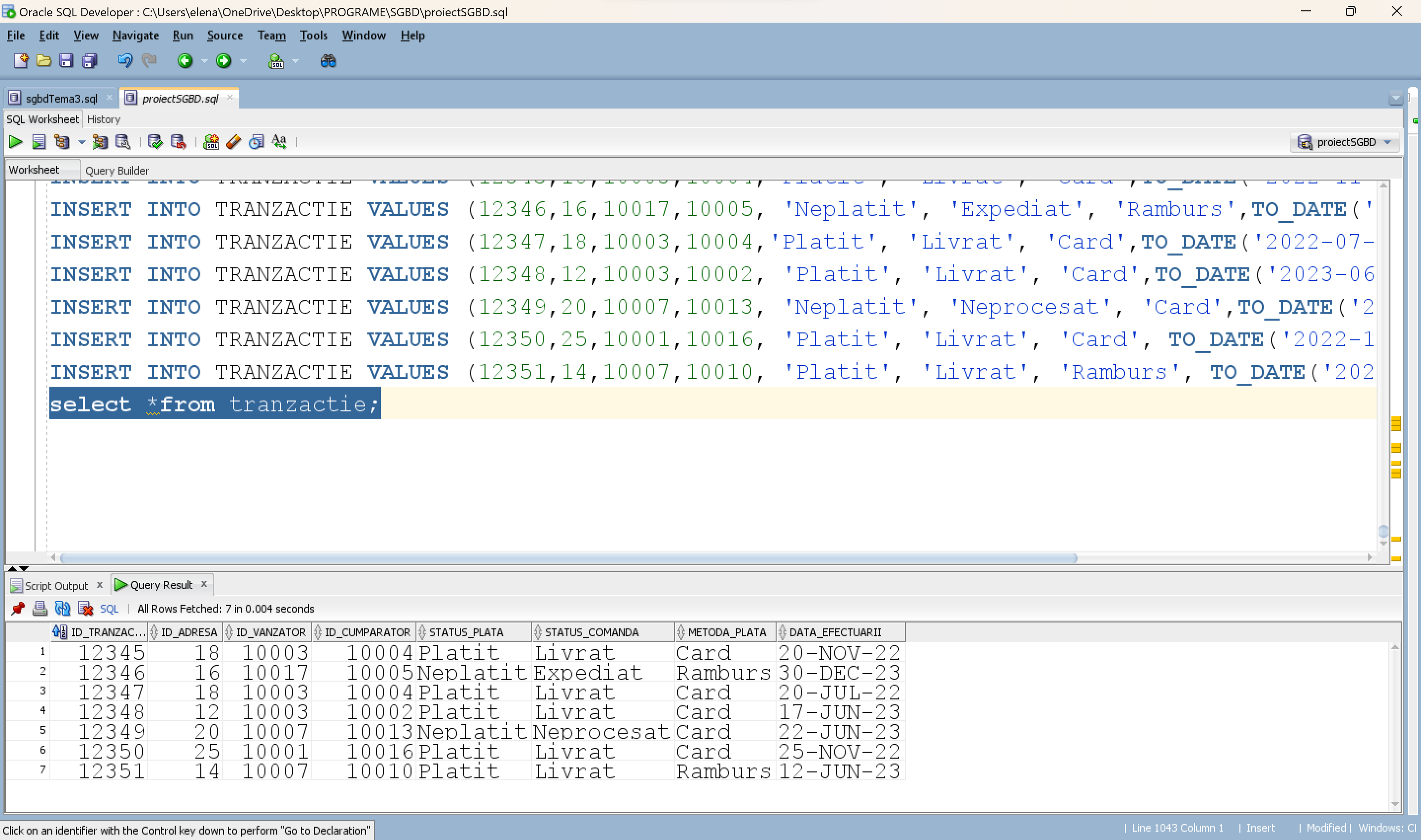
INSERT INTO TRANZACTIE VALUES (12347,18,10003,10004,'Platit', 'Livrat', 'Card',TO\_DATE('2022-07-20','YYYY-MM-DD'));

INSERT INTO TRANZACTIE VALUES (12348,12,10003,10002, 'Platit', 'Livrat', 'Card',TO\_DATE('2023-06-17','YYYY-MM-DD'));

INSERT INTO TRANZACTIE VALUES (12349,20,10007,10013, 'Neplatit', 'Neprocesat', 'Card',TO\_DATE('2023-06-22','YYYY-MM-DD'));

INSERT INTO TRANZACTIE VALUES (12350,25,10001,10016, 'Platit', 'Livrat', 'Card', TO\_DATE('2022-11-25','YYYY-MM-DD'));

INSERT INTO TRANZACTIE VALUES (12351,14,10007,10010, 'Platit', 'Livrat', 'Ramburs', TO\_DATE('2023-6-12','YYYY-MM-DD')); 



INSERT INTO PRODUS VALUES (1,50,'Mediu','Bej','H/M','Purtat','Bumbac','feminin', 10003,12345);

INSERT INTO PRODUS VALUES (2,500,'Small','Alb','Ralph-Lauren','Uzat','Tricot','feminin', 10003,12345);

INSERT INTO PRODUS VALUES (3,45,'Large','Negru','H/M','Nepurtat','Stofa','masculin', 10003,12347);

INSERT INTO PRODUS VALUES (4,120,'Large','Roz','Ralph-Lauren','Nepurtat','Poliester','feminin', 10003, NULL);

INSERT INTO PRODUS VALUES (5,250,'45','Alb','Nike','Nepurtat','Panza','masculin',10007,12349);

INSERT INTO PRODUS VALUES (6,37,'Large','Albastru deschis','Bershka','Purtat','Poliester','masculin', 10002, NULL);

INSERT INTO PRODUS VALUES (7,44,'Small','Multicolor','NoName','Nepurtat','Bumbac','unisex', 10001,12350);

INSERT INTO PRODUS VALUES (8,50,'Mediu','Bej','H/M','Purtat','Bumbac','feminin', 10003,12348);

INSERT INTO PRODUS VALUES (9,13,'Large','Alb','NoName','Uzat','Tricot','feminin', 10009, NULL);

INSERT INTO PRODUS VALUES (10,45,'36','Negru','H/M','Nepurtat','Stofa','masculin', 10007, 12351);

INSERT INTO PRODUS VALUES (11,5,'Large','Grena','NoName','Nepurtat','Poliester','feminin', 10015,NULL);

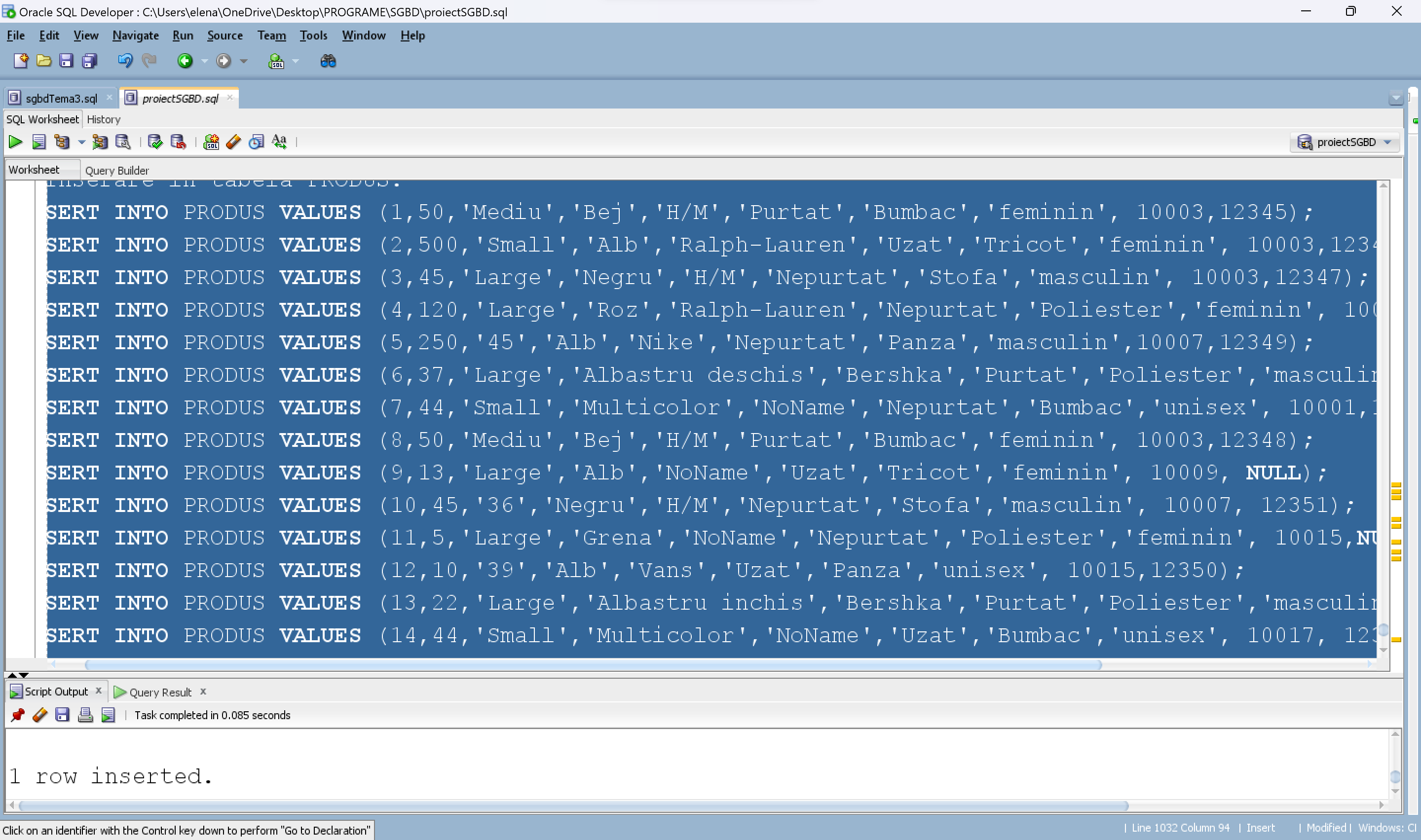
INSERT INTO PRODUS VALUES (12,10,'39','Alb','Vans','Uzat','Panza','unisex', 10015,12350);

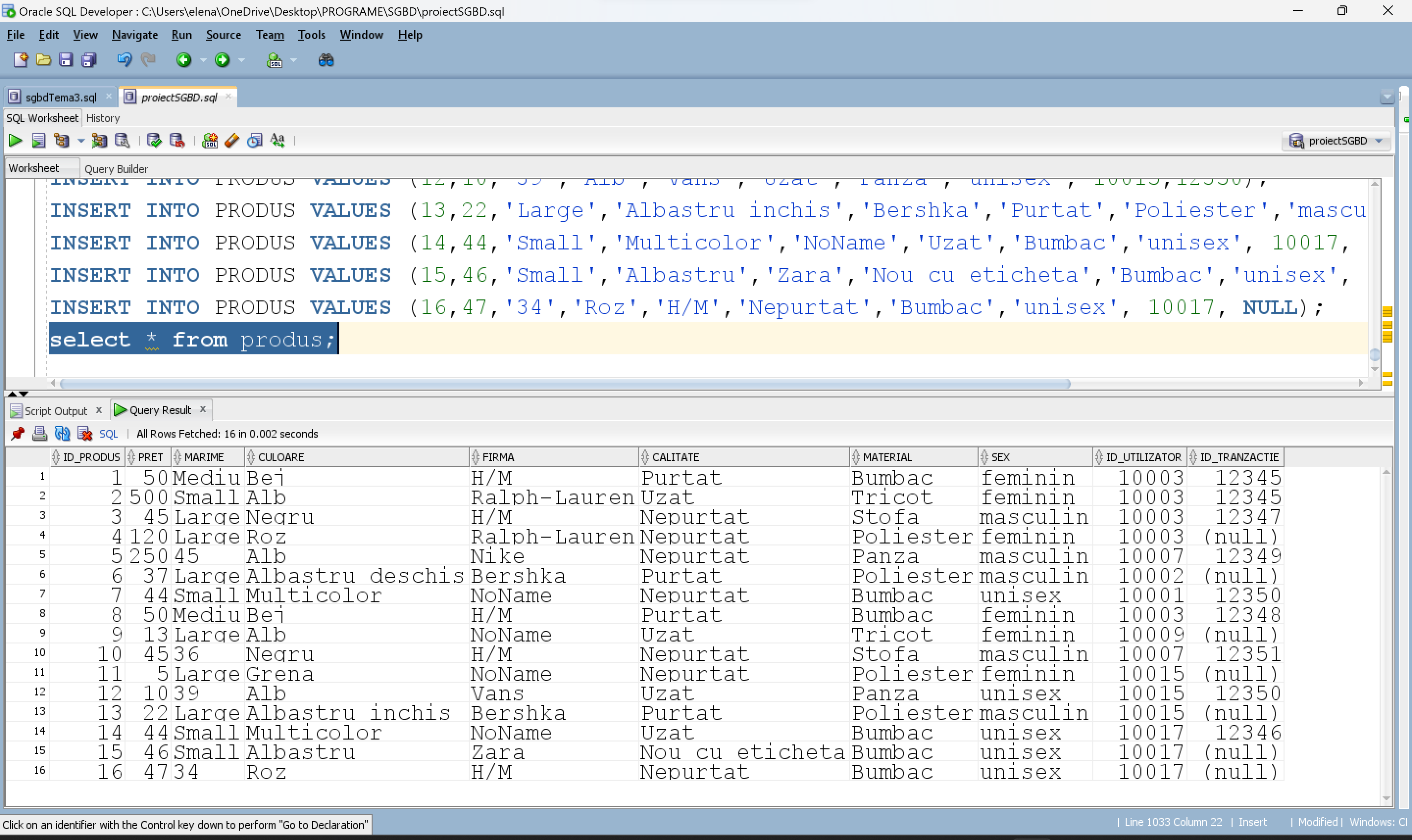
INSERT INTO PRODUS VALUES (13,22,'Large','Albastru inchis','Bershka','Purtat','Poliester','masculin', 10015, NULL);

INSERT INTO PRODUS VALUES (14,44,'Small','Multicolor','NoName','Uzat','Bumbac','unisex', 10017, 12346);

INSERT INTO PRODUS VALUES (15,46,'Small','Albastru','Zara','Nou cu eticheta','Bumbac','unisex', 10017, NULL);

INSERT INTO PRODUS VALUES (16,47,'34','Roz','H/M','Nepurtat','Bumbac','unisex', 10017, NULL);





INSERT INTO RECENZIE VALUES(10101,'Excelent','Comunicare promta', 5, 10004, 12345 );

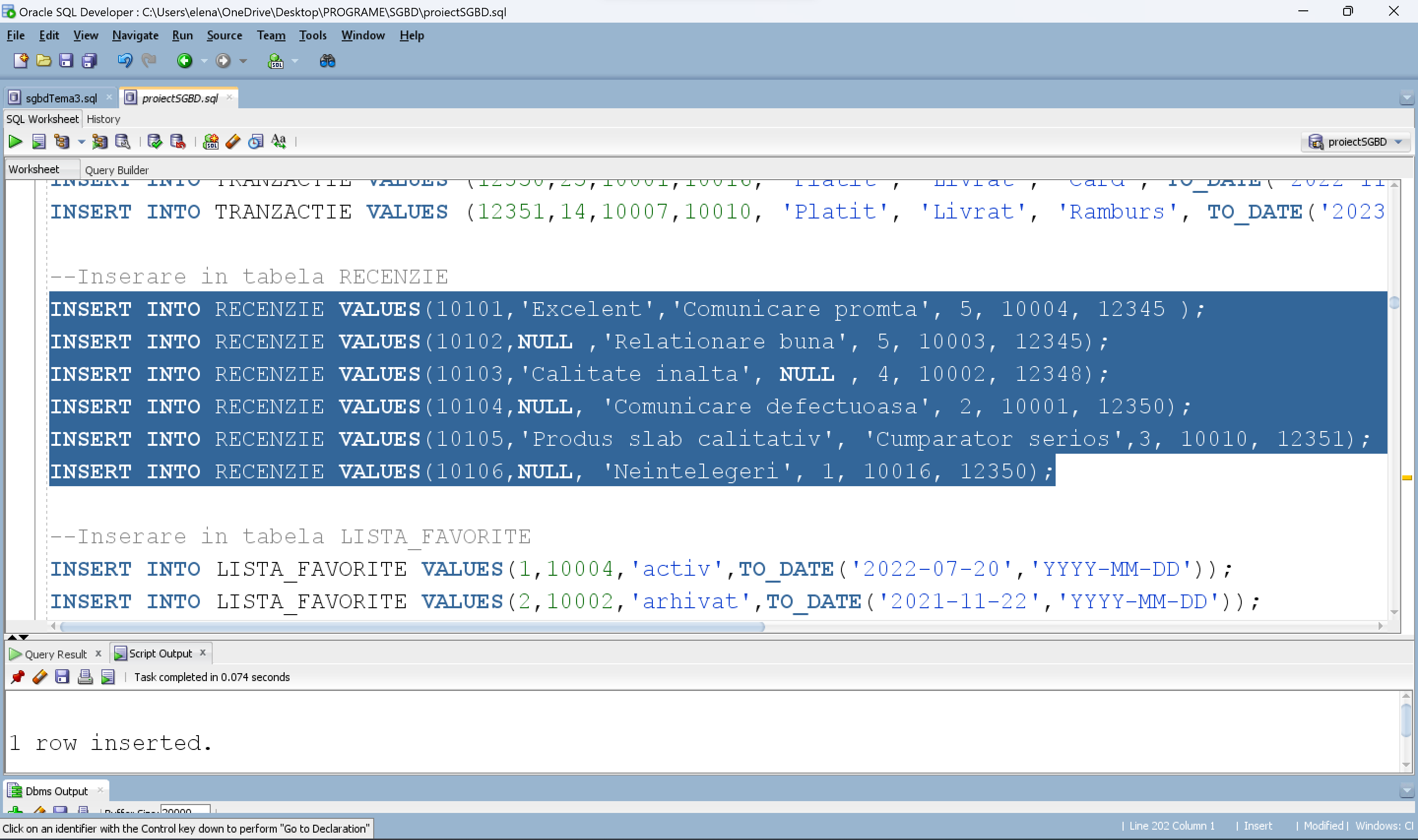
INSERT INTO RECENZIE VALUES(10102,NULL ,'Relationare buna', 5, 10003, 12345);

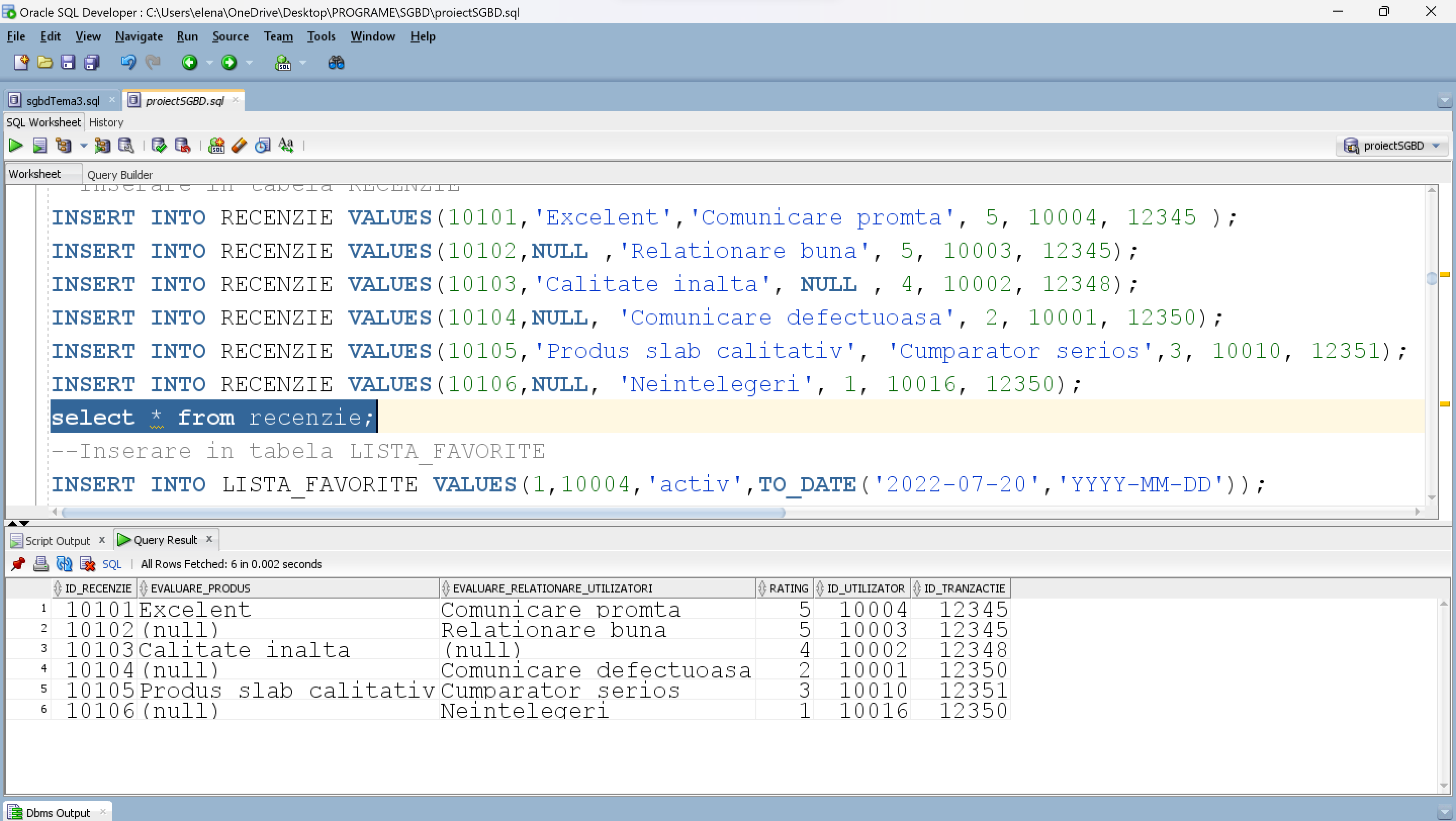
INSERT INTO RECENZIE VALUES(10103,'Calitate inalta', NULL , 4, 10002, 12348);

INSERT INTO RECENZIE VALUES(10104,NULL, 'Comunicare defectuoasa', 2, 10001, 12350);

INSERT INTO RECENZIE VALUES(10105,'Produs slab calitativ', 'Cumparator serios',3, 10010, 12351);

INSERT INTO RECENZIE VALUES(10106,NULL, 'Neintelegeri', 1, 10016, 12350);





INSERT INTO LISTA\_FAVORITE VALUES(1,10004,'activ',TO\_DATE('2022-07-20','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(2,10002,'arhivat',TO\_DATE('2021-11-22','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(3,10006,'activ',TO\_DATE('2023-11-20','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(4,10013,'activ',TO\_DATE('2021-11-20','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(5,10016,'sters',TO\_DATE('2020-10-25','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(6,10005,'arhivat',TO\_DATE('2022-11-20','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(7,10010,'arhivat',TO\_DATE('2023-03-20','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(8,10017,'sters',TO\_DATE('2023-12-21','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(9,10003,'activ',TO\_DATE('2022-03-12','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(10,10001,'arhivat',TO\_DATE('2020-11-20','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(11,10007,'activ',TO\_DATE('2021-09-30','YYYY-MM-DD'));

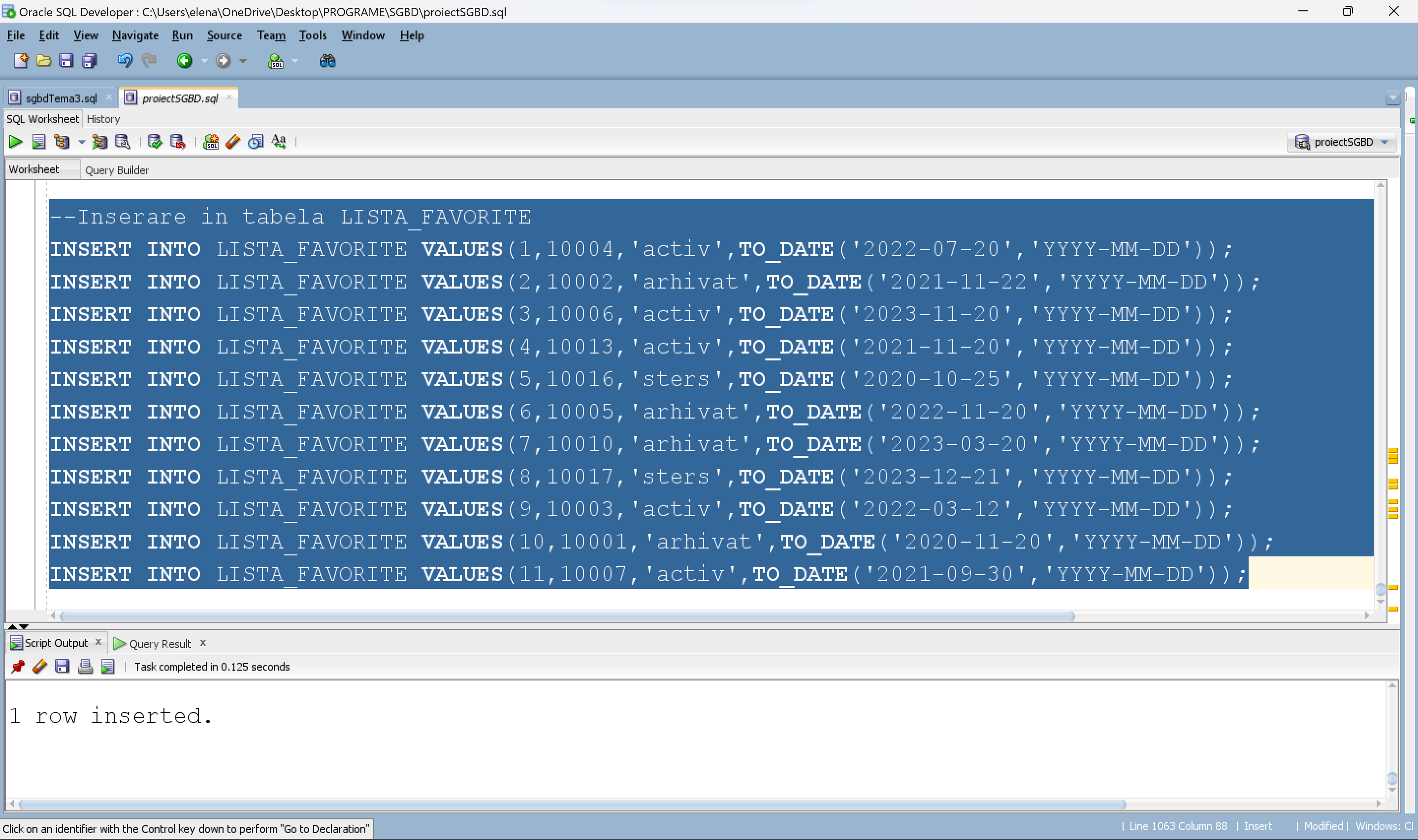
INSERT INTO LISTA\_FAVORITE VALUES(12,10008,'arhivat',TO\_DATE('2023-12-04','YYYY-MM-DD'));

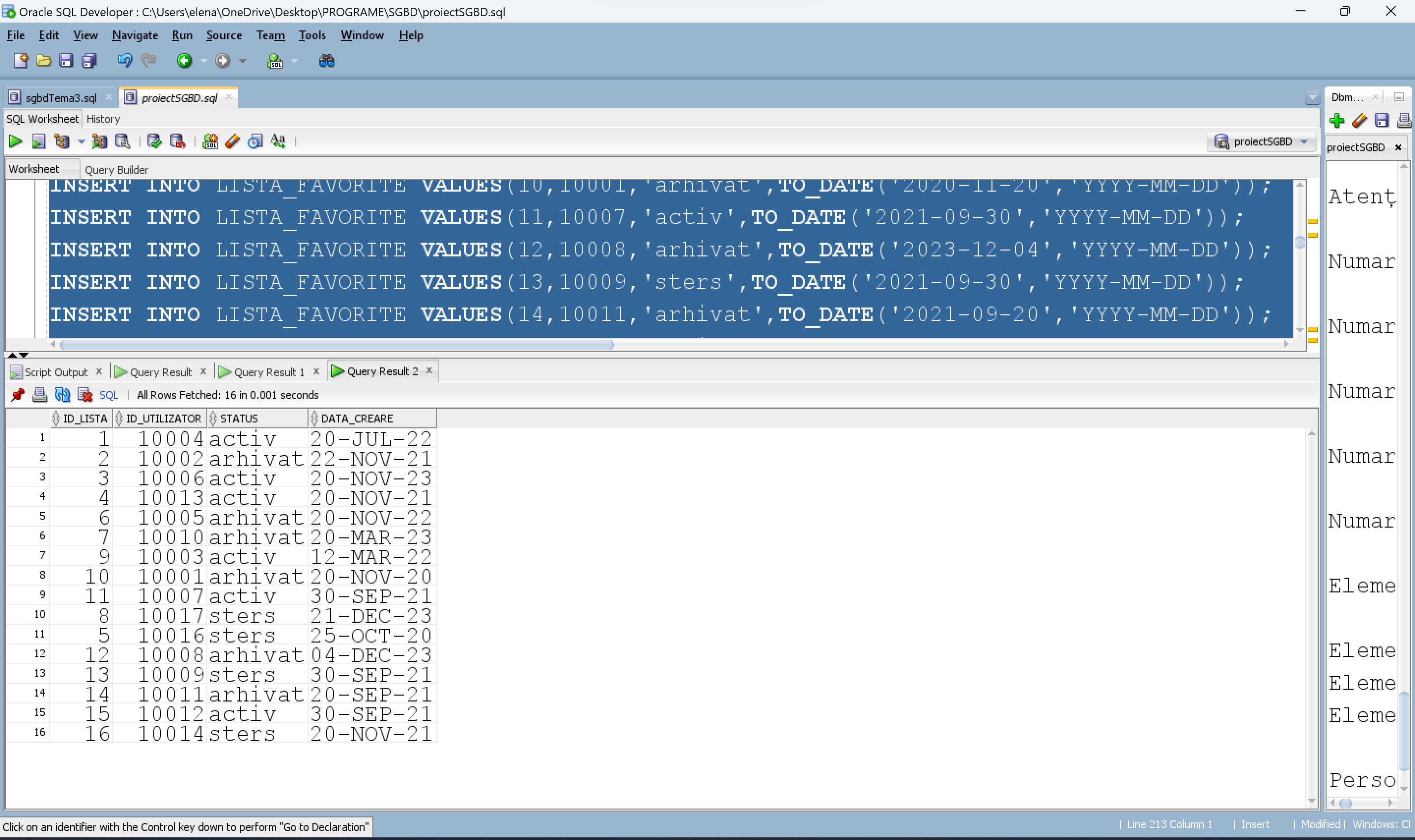
INSERT INTO LISTA\_FAVORITE VALUES(13,10009,'sters',TO\_DATE('2021-09-30','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(14,10011,'arhivat',TO\_DATE('2021-09-20','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(15,10012,'activ',TO\_DATE('2021-09-30','YYYY-MM-DD'));

INSERT INTO LISTA\_FAVORITE VALUES(16,10014,'sters',TO\_DATE('2021-11-20','YYYY-MM-DD'));





INSERT INTO CATEGORIE\_PRODUS VALUES (11,1);

INSERT INTO CATEGORIE\_PRODUS VALUES (19,1);

INSERT INTO CATEGORIE\_PRODUS VALUES (9,2);

INSERT INTO CATEGORIE\_PRODUS VALUES (12,3);

INSERT INTO CATEGORIE\_PRODUS VALUES (18,3);

INSERT INTO CATEGORIE\_PRODUS VALUES (14,4);

INSERT INTO CATEGORIE\_PRODUS VALUES (18,4);

INSERT INTO CATEGORIE\_PRODUS VALUES (16,5);

INSERT INTO CATEGORIE\_PRODUS VALUES (23,5);

INSERT INTO CATEGORIE\_PRODUS VALUES (17,5);

INSERT INTO CATEGORIE\_PRODUS VALUES (1,6);

INSERT INTO CATEGORIE\_PRODUS VALUES (4,6);

INSERT INTO CATEGORIE\_PRODUS VALUES (9,7);

INSERT INTO CATEGORIE\_PRODUS VALUES (5,8);

INSERT INTO CATEGORIE\_PRODUS VALUES (1,8);

INSERT INTO CATEGORIE\_PRODUS VALUES (19,8);

INSERT INTO CATEGORIE\_PRODUS VALUES (10,9);

INSERT INTO CATEGORIE\_PRODUS VALUES (15,10);

INSERT INTO CATEGORIE\_PRODUS VALUES (18,10);

INSERT INTO CATEGORIE\_PRODUS VALUES (15,11);

INSERT INTO CATEGORIE\_PRODUS VALUES (16,12);

INSERT INTO CATEGORIE\_PRODUS VALUES (18,12);

INSERT INTO CATEGORIE\_PRODUS VALUES (3,13);

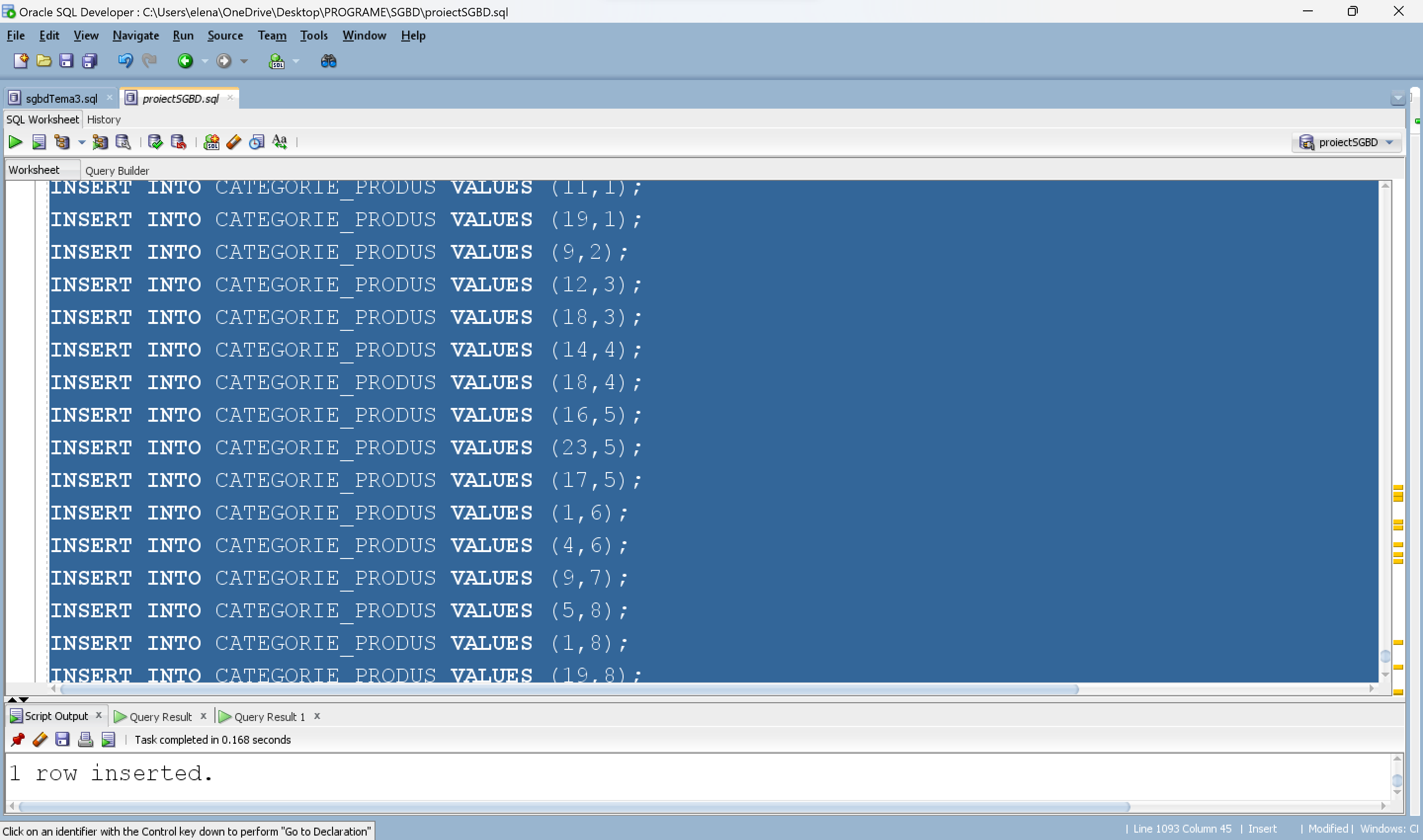
INSERT INTO CATEGORIE\_PRODUS VALUES (9,14);

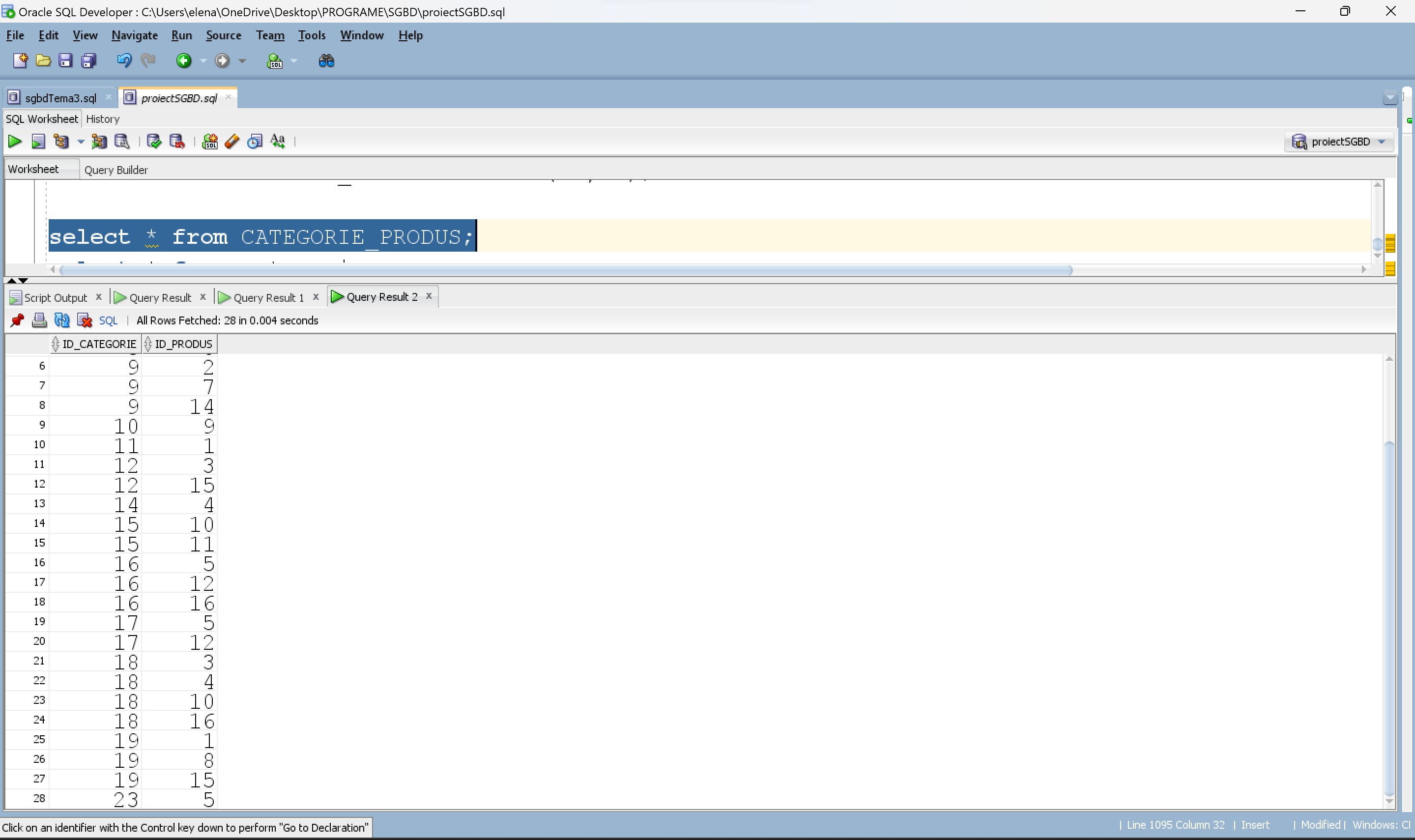
INSERT INTO CATEGORIE\_PRODUS VALUES (12,15);

INSERT INTO CATEGORIE\_PRODUS VALUES (19,15);

INSERT INTO CATEGORIE\_PRODUS VALUES (16,16);

INSERT INTO CATEGORIE\_PRODUS VALUES (18,16);





INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (1,1);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (1,2);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (1,3);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (1,4);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (2,1);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (3,5);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (3,9);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (4,6);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (5,15);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (5,16);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (6,9);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (7,6);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (8,1);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (9,1);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (10,12);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (10,11);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (10,13);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (11,12);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (12,9);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (12,14);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (12,15);

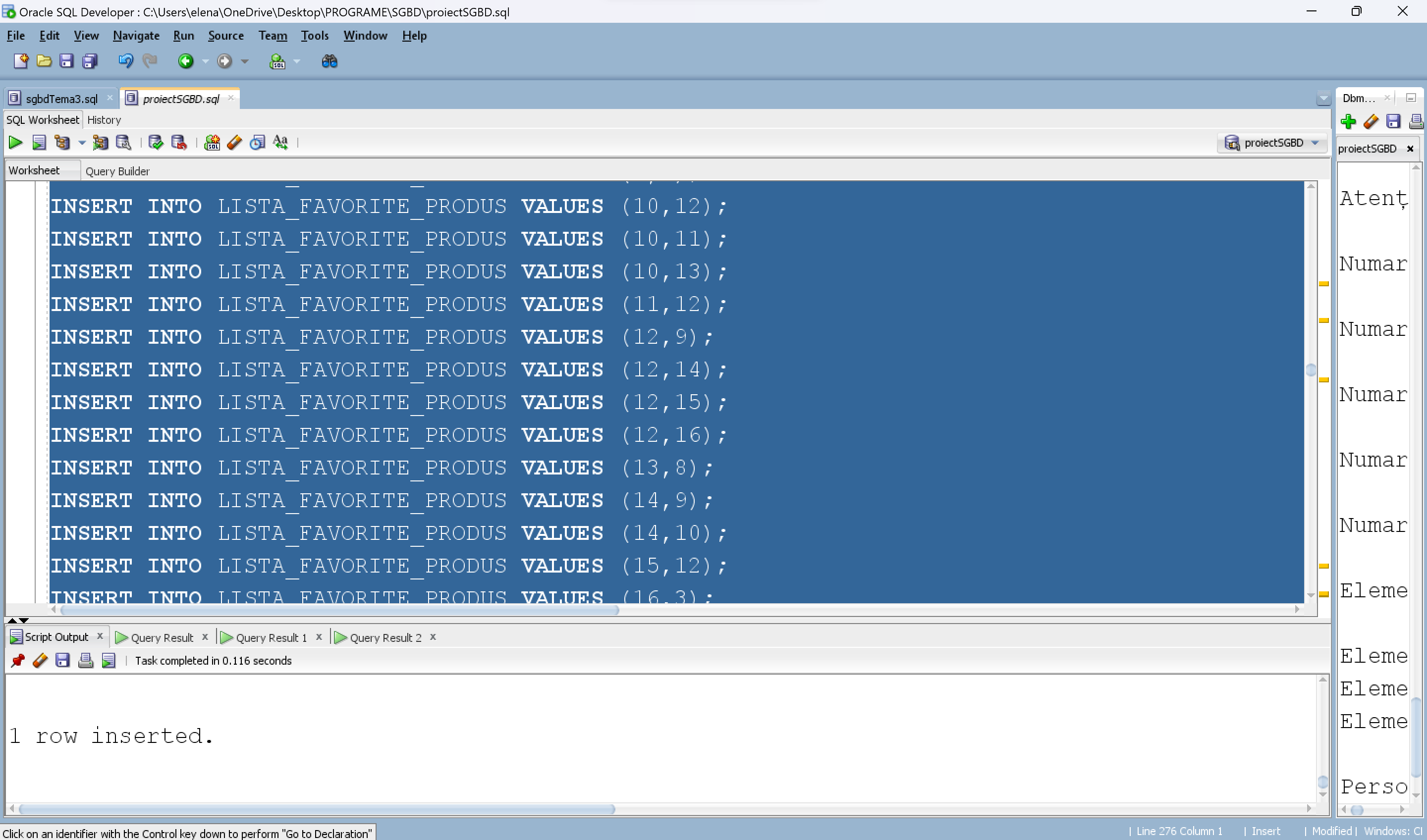
INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (12,16);

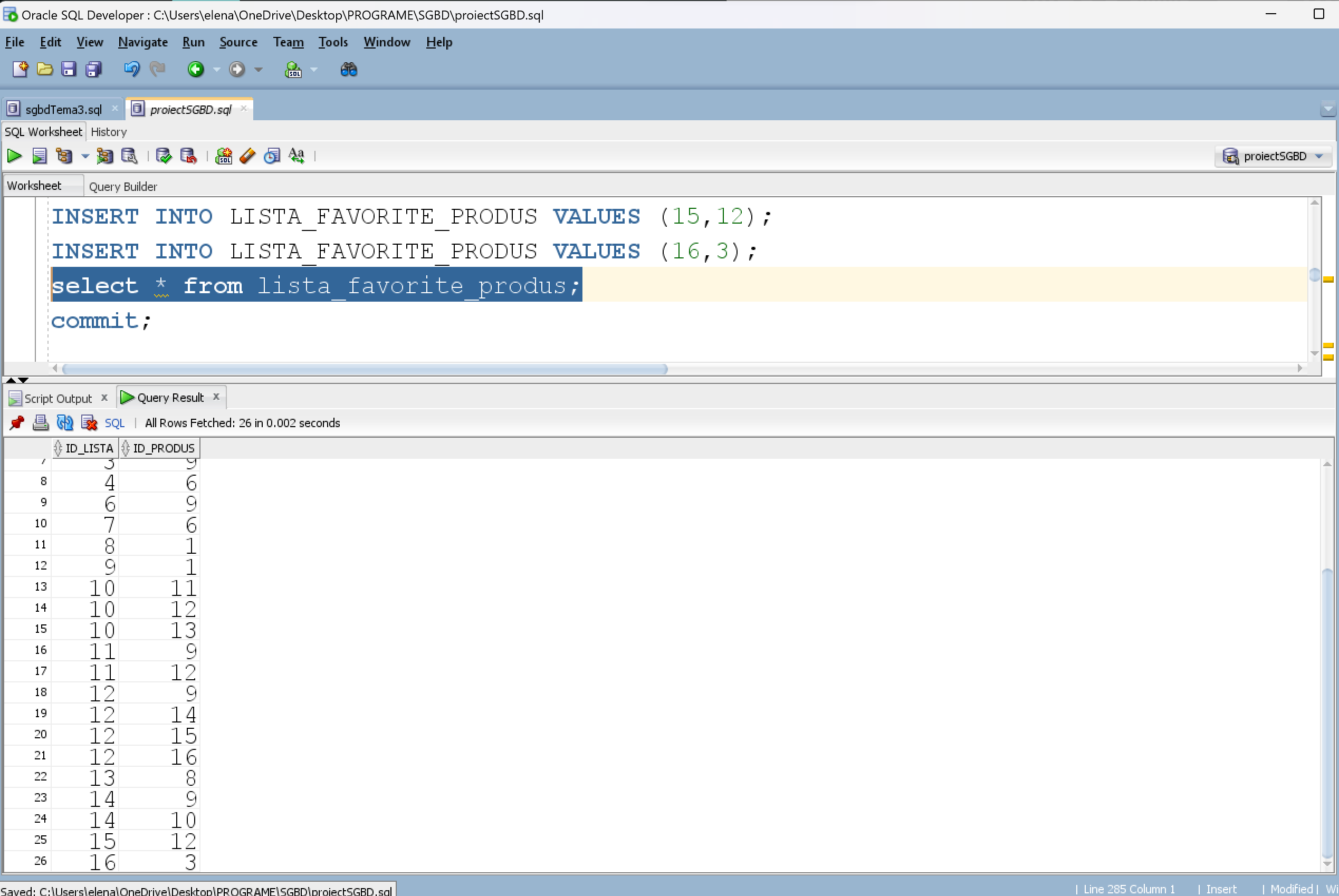
INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (13,8);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (14,9);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (14,10);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (15,12);

INSERT INTO LISTA\_FAVORITE\_PRODUS VALUES (16,3); 



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.

Definiți un subprogram stocat independent de tip procedură prin care să obțineți și să afișați următoarele informații:

- însumarea prețurilor produselor din lista de favorite ale utilizatorilor care și-au creat lista de favorite în aceeași zi în care și-au înregistrat contul in aplicație

-pentru utilizatorul cu cele mai puține produse vândute, afișați informații despre produsele sale

CREATE OR REPLACE PROCEDURE exercitiul\_6 IS

TYPE tabel\_produse IS TABLE OF VARCHAR2(50);

TYPE coduri\_utilizator IS VARRAY(50) OF utilizator.id\_utilizator%TYPE;

TYPE tabel\_pret\_produse IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

produse\_nevandute tabel\_produse := tabel\_produse();

cod\_utilizatori coduri\_utilizator := coduri\_utilizator();

suma\_produse\_fav tabel\_pret\_produse;

utilizator\_fara\_vanzari utilizator.id\_utilizator%TYPE;

ct int := 1;

BEGIN

SELECT id\_utilizator INTO utilizator\_fara\_vanzari

FROM (

SELECT id\_utilizator, COUNT(id\_produs) AS numar\_produse

FROM produs

WHERE id\_tranzactie IS NULL

GROUP BY id\_utilizator

ORDER BY numar\_produse DESC

)

WHERE ROWNUM = 1;

FOR cod IN (SELECT l.id\_utilizator

FROM lista\_favorite l

JOIN utilizator u ON l.id\_utilizator = u.id\_utilizator

WHERE l.data\_creare = u.data\_inregistrare) LOOP

cod\_utilizatori.EXTEND;

cod\_utilizatori(ct) := cod.id\_utilizator;

ct := ct + 1;

END LOOP;

--stiu ca cod\_utilizatori si suma+produse\_fav au acelasi nr de elem, nu mai am nev de ct

FOR i IN cod\_utilizatori.FIRST..cod\_utilizatori.LAST LOOP

SELECT SUM(p.pret) INTO suma\_produse\_fav(i)

FROM produs p

JOIN lista\_favorite\_produs lf ON lf.id\_produs = p.id\_produs

JOIN lista\_favorite l ON lf.id\_lista = l.id\_lista

WHERE l.id\_utilizator = cod\_utilizatori(i);

END LOOP;

FOR i IN suma\_produse\_fav.FIRST..suma\_produse\_fav.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Suma preturilor pentru Id-ul '|| COD\_UTILIZATORI(i) || ' este ' || suma\_produse\_fav(i));

END LOOP;

FOR produs IN (SELECT firma, marime, culoare, calitate, sex

FROM produs

WHERE id\_utilizator = utilizator\_fara\_vanzari) LOOP

produse\_nevandute.EXTEND;

produse\_nevandute(produse\_nevandute.LAST) := produs.firma || ', ' || produs.marime || ', ' ||

produs.culoare || ', ' || produs.calitate || ', ' ||

produs.sex;

END LOOP;

FOR i IN produse\_nevandute.FIRST..produse\_nevandute.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Informatii produs: ');

DBMS\_OUTPUT.PUT\_LINE(produse\_nevandute(i));

END LOOP;

END exercitiul\_6;

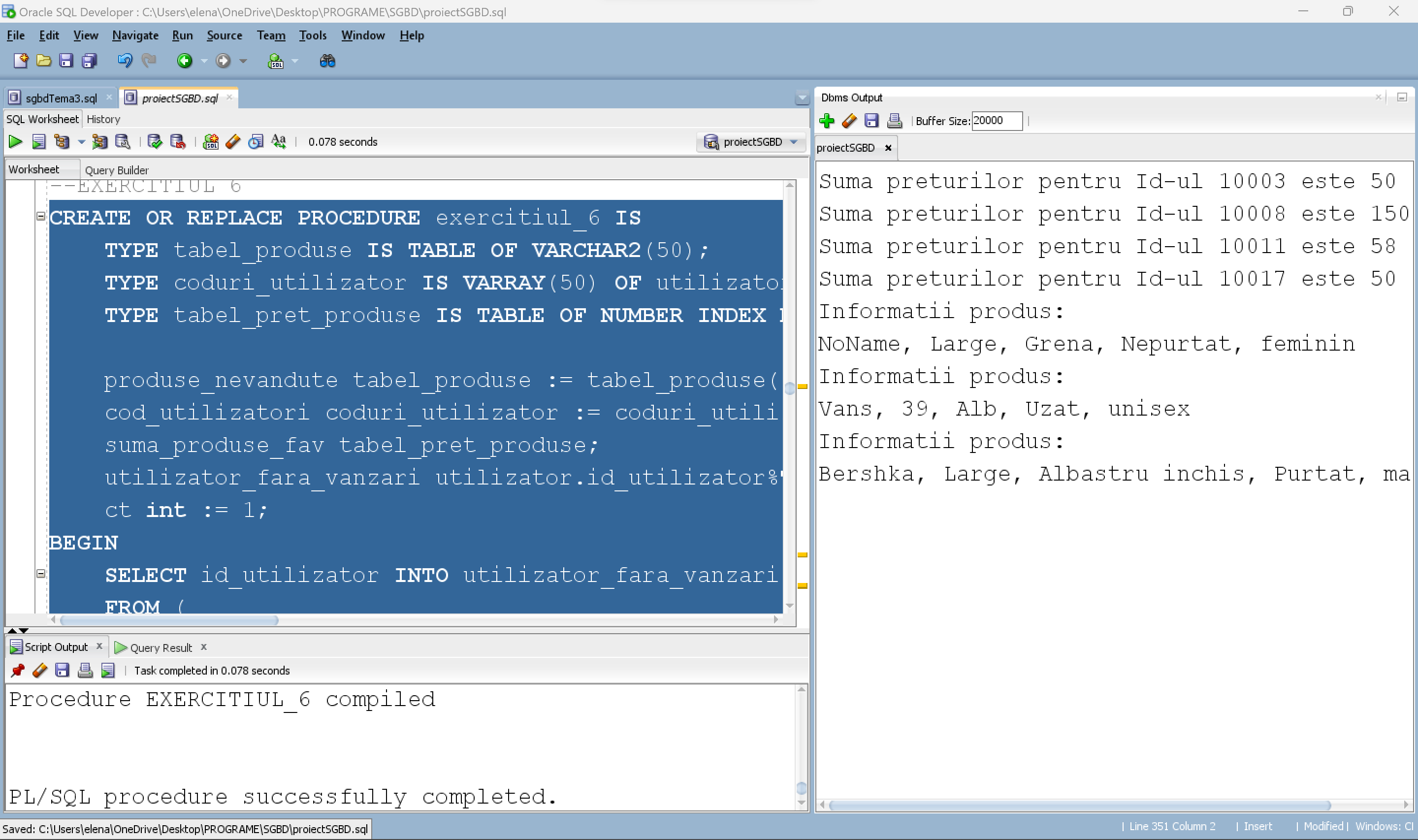
/

BEGIN

exercitiul\_6;

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.

Definiți un subprogram stocat independent de tip procedură care afișează numele, prenumele și sumarul recenziei fiecărui utilizator care a lăsat o recenzie în urma unei tranzacții în calitate de cumpărător.

CREATE OR REPLACE PROCEDURE Exercitiul\_7 IS

-- Cursor pentru selectarea utilizatorilor care au lasat recenzii ca cumparatori

CURSOR c\_utilizatori\_cu\_recenzii IS

SELECT DISTINCT u.id\_utilizator, u.nume, u.prenume

FROM utilizator u

JOIN tranzactie t ON u.id\_utilizator = t.id\_cumparator

JOIN recenzie r ON t.id\_tranzactie = r.id\_tranzactie;

-- Cursor parametrizat pentru selectarea recenziilor bazate pe id-ul utilizatorului

CURSOR c\_recenzii (p\_id\_utilizator NUMBER) IS

SELECT r.evaluare\_produs, r.evaluare\_relationare\_utilizatori, r.rating

FROM recenzie r

JOIN tranzactie t ON r.id\_tranzactie = t.id\_tranzactie

WHERE t.id\_cumparator = p\_id\_utilizator AND r.id\_utilizator = p\_id\_utilizator;

v\_utilizator c\_utilizatori\_cu\_recenzii%ROWTYPE;

v\_recenzie c\_recenzii%ROWTYPE;

BEGIN

OPEN c\_utilizatori\_cu\_recenzii;

LOOP

FETCH c\_utilizatori\_cu\_recenzii INTO v\_utilizator;

EXIT WHEN c\_utilizatori\_cu\_recenzii%NOTFOUND;

-- Afișarea informațiilor utilizatorului

DBMS\_OUTPUT.PUT\_LINE('Utilizatorul: ' || v\_utilizator.nume || ' ' || v\_utilizator.prenume);

-- Deschiderea cursorului parametrizat pentru recenzii

OPEN c\_recenzii(v\_utilizator.id\_utilizator);

LOOP

FETCH c\_recenzii INTO v\_recenzie;

EXIT WHEN c\_recenzii%NOTFOUND;

-- Afișarea recenziilor utilizatorului

DBMS\_OUTPUT.PUT\_LINE(' Recenzie: Evaluare Produs - ' || v\_recenzie.evaluare\_produs ||

'; Evaluare Relationare - ' || v\_recenzie.evaluare\_relationare\_utilizatori ||

'; Rating - ' || v\_recenzie.rating);

END LOOP;

CLOSE c\_recenzii;

END LOOP;

CLOSE c\_utilizatori\_cu\_recenzii;

END Exercitiul\_7;

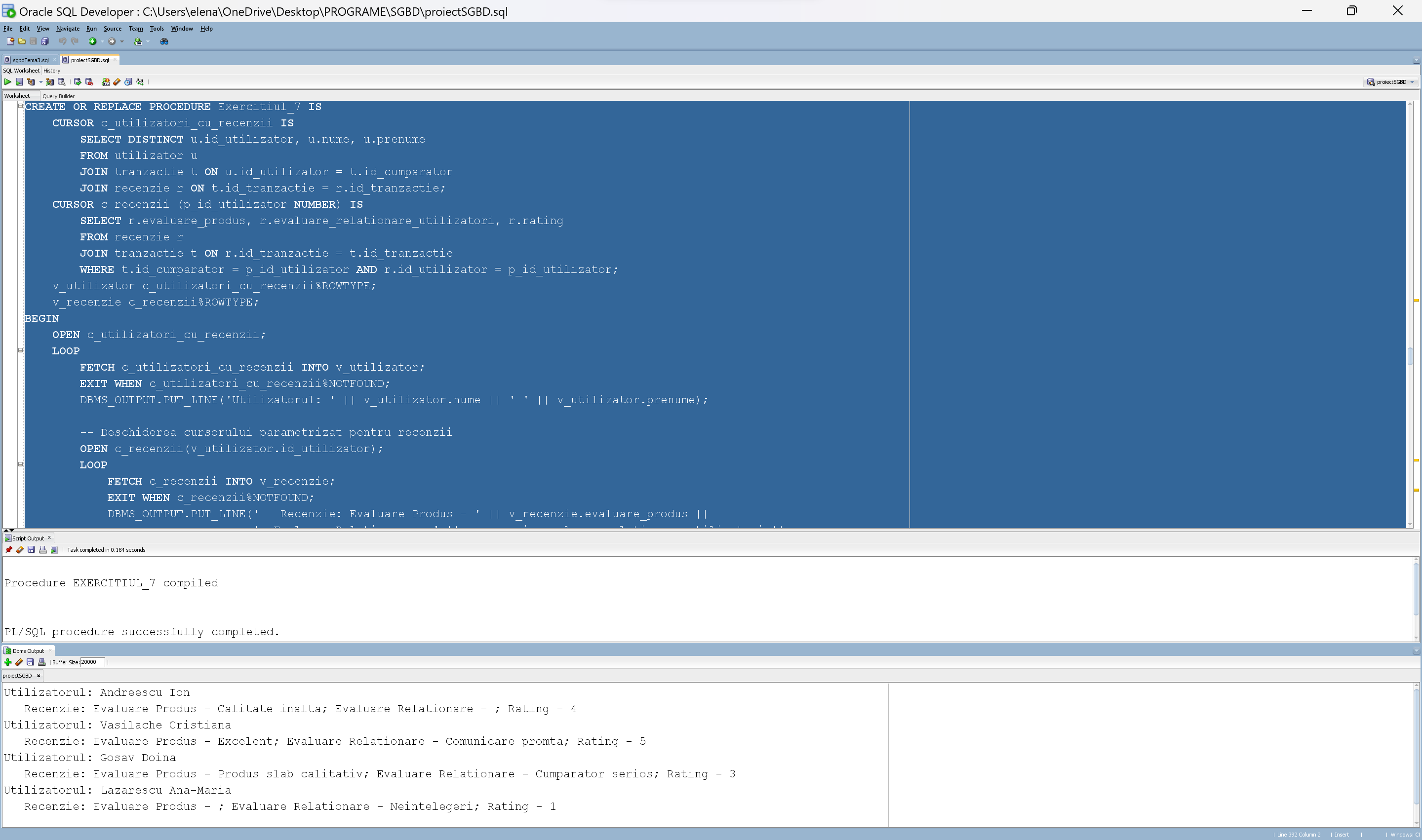
/

BEGIN

Exercitiul\_7;

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.

Definiți o funcție stocată independent care afișează denumirea categoriei din care au fost vândute cele mai multe produse într-un anumit interval de timp, nu mai mare de un an. În cazul în care există mai multe categorii, va fi afișată prima alfabetic. Tratați cazurile în care nu este corect input-ul de intervale de timp și cazul în care intervalul este mai mare de un an.

CREATE OR REPLACE FUNCTION Exercitiul\_8(p\_data\_inceput DATE, p\_data\_sfarsit DATE)

RETURN VARCHAR2 IS

denumire\_categorie VARCHAR2(50);

max\_vanzari INT;

data\_invalida EXCEPTION;

interval\_prea\_mare EXCEPTION;

BEGIN

IF p\_data\_inceput > p\_data\_sfarsit THEN

RAISE data\_invalida;

END IF;

IF p\_data\_sfarsit - p\_data\_inceput > 365 THEN

RAISE interval\_prea\_mare;

END IF;

SELECT c.denumire INTO denumire\_categorie

FROM categorie c

JOIN categorie\_produs cp ON c.id\_categorie = cp.id\_categorie

JOIN produs p ON cp.id\_produs = p.id\_produs

JOIN tranzactie t ON p.id\_tranzactie = t.id\_tranzactie

WHERE t.data\_efectuarii BETWEEN p\_data\_inceput AND p\_data\_sfarsit

GROUP BY c.denumire

ORDER BY COUNT(\*) DESC, c.denumire

FETCH FIRST 1 ROW ONLY;

RETURN denumire\_categorie;

EXCEPTION

WHEN data\_invalida THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Data de inceput este mai mare decât data de sfarsit.');

RETURN NULL;

WHEN interval\_prea\_mare THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Intervalul de timp specificat este prea mare.');

RETURN NULL;

END Exercitiul\_8;

/

DECLARE

test1 VARCHAR2(50);

test2 VARCHAR2(50);

test3 VARCHAR2(50);

BEGIN

--input-ul nu este corect

test1:= Exercitiul\_8(TO\_DATE('2023-10-01', 'YYYY-MM-DD'), TO\_DATE('2020-03-01', 'YYYY-MM-DD'));

--diferenta de timp este prea mare

test2:= Exercitiul\_8(TO\_DATE('2020-01-01', 'YYYY-MM-DD'), TO\_DATE('2024-01-01', 'YYYY-MM-DD'));

--inputul este corect

test3:= Exercitiul\_8(TO\_DATE('2022-01-01', 'YYYY-MM-DD'), TO\_DATE('2023-01-01', 'YYYY-MM-DD'));

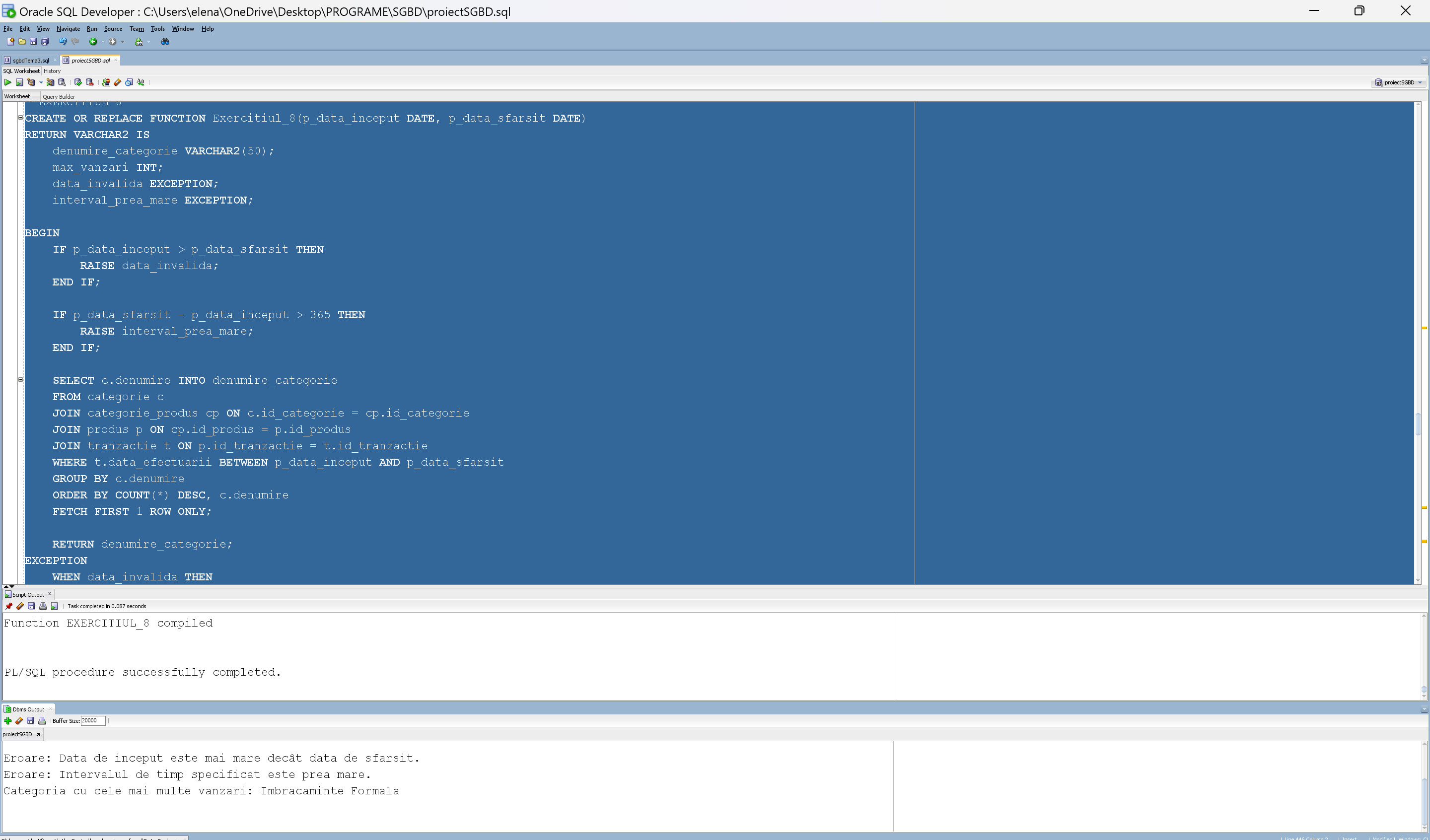
IF test3 IS NOT NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Categoria cu cele mai multe vanzari: ' || test3);

END IF;

END;

/



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.

Definiți o procedură stocată care afișează informații despre unicul utilizator dintr-un oraș dat ca parametru, care a vândut cel puțin două produse cu o medie a recenziilor mai mare decât 2. Gestionați excepțiile NO\_DATA\_FOUND și TOO\_MANY\_ROWS.

CREATE OR REPLACE PROCEDURE Exercitiul\_9(

p\_oras IN VARCHAR2

) IS

nume\_utilizator VARCHAR2(100);

prenume\_utilizator VARCHAR2(100);

castig\_total NUMBER;

utilizator\_fara\_vanzari EXCEPTION;

verif\_oras INT;

medie\_recenzii NUMBER;

BEGIN

-- verificam daca orasul este unic: daca codul nu e corect se duce pe eroarea too many rows

SELECT id\_adresa INTO verif\_oras

FROM adresa

WHERE oras = p\_oras;

-- selectam utilizatorul care indeplineste conditiile

SELECT u.nume, u.prenume, SUM(p.pret), AVG(r.rating)

INTO nume\_utilizator, prenume\_utilizator, castig\_total, medie\_recenzii

FROM utilizator u

JOIN adresa a ON u.id\_adresa = a.id\_adresa

JOIN tranzactie t ON u.id\_utilizator = t.id\_vanzator

LEFT JOIN produs p ON t.id\_tranzactie = p.id\_tranzactie

LEFT JOIN recenzie r ON t.id\_tranzactie = r.id\_tranzactie AND t.id\_cumparator = r.id\_utilizator

WHERE a.oras = p\_oras

GROUP BY u.nume, u.prenume

HAVING COUNT(p.id\_produs) >= 2 AND AVG(r.rating)> 2

FETCH FIRST 1 ROW ONLY;

DBMS\_OUTPUT.PUT\_LINE('Nume: ' || nume\_utilizator || ', Prenume: ' || prenume\_utilizator || ', Castig total: ' || castig\_total || ', Medie recenzii: ' || medie\_recenzii);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

IF verif\_oras IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Nu exista orasul ' || p\_oras || ' in baza de date.');

ELSE DBMS\_OUTPUT.PUT\_LINE('Eroare: Nu exista utilizator in orasul ' || p\_oras || ' care sa indeplineasca conditiile.');

END IF;

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Exista mai multi utilizatori care locuiesc in orasul ' || p\_oras || '.');

END Exercitiul\_9;

/

BEGIN

Exercitiul\_9('Bucuresti');

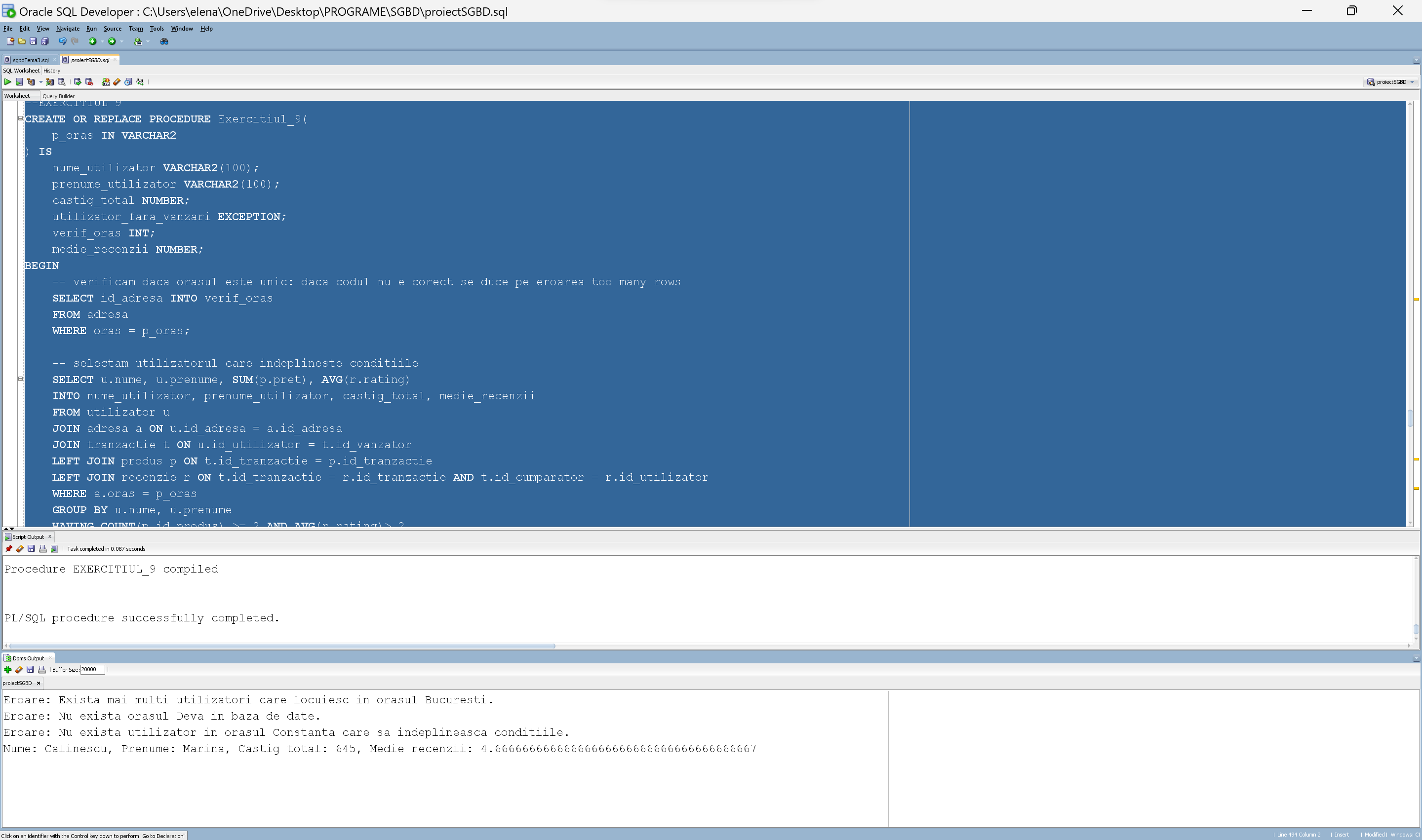
Exercitiul\_9('Deva');

Exercitiul\_9('Constanta');

Exercitiul\_9('Braila');

END;

/



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

Creați un trigger care afișează numărul produselor care au prețurile cuprinse intre 50 și 100 după ce un produs este adăugat, șters, sau editat.

CREATE OR REPLACE TRIGGER Exercitiul\_10

AFTER INSERT OR DELETE OR UPDATE ON produs

DECLARE

total\_produse NUMBER:=0;

BEGIN

select Count(\*) into total\_produse

from produs

where pret>=50 and pret<=100;

IF total\_produse<>0 then

DBMS\_OUTPUT.PUT\_LINE('Numarul de produse este: '||total\_produse);

ELSE

DBMS\_OUTPUT.PUT\_LINE('Nu exista produse cuprinse in intervalul respectiv de pret.');

END IF;

END;

/

--adaugarea unui produs

INSERT INTO PRODUS VALUES (17,53,'Small','Crem','NoName','Nepurtat','Bumbac','unisex', 10017, NULL);

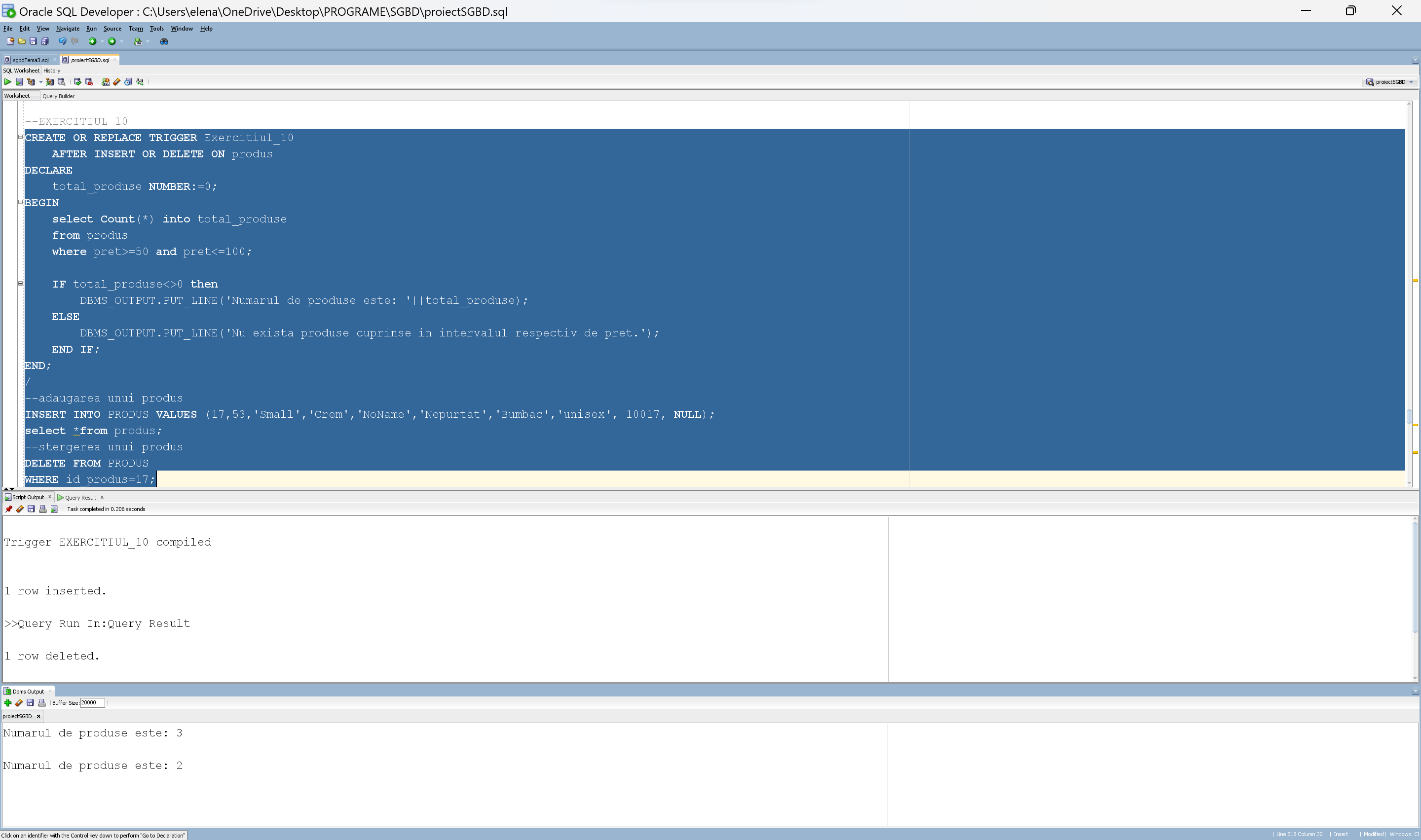
select \*from produs;

--stergerea unui produs

DELETE FROM PRODUS

WHERE id\_produs=17;

DROP TRIGGER Exercitiul\_10;



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

Creați un trigger care șterge liste de favorite alte utilizatorilor numai dacă statusul acestora este marcat drept „șters”, altfel aruncă o eroare, totodată afișând numărul de ștergeri efectuate și modificând tabelul lista\_favorite\_produs în funcție de listele șterse.

CREATE OR REPLACE TRIGGER Exercitiul\_11

FOR DELETE ON lista\_favorite

COMPOUND TRIGGER

total NUMBER :=0;

numar\_sters NUMBER :=0;

numar\_ramas NUMBER:=0;

BEFORE STATEMENT IS

BEGIN

SELECT COUNT(\*) INTO total

FROM lista\_favorite;

END BEFORE STATEMENT;

BEFORE EACH ROW IS

BEGIN

IF :OLD.status <> 'sters' THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Informatia nu poate fi stearsa');

END IF;

DELETE FROM lista\_favorite\_produs l

WHERE l.id\_lista = :OLD.id\_lista;

END BEFORE EACH ROW;

AFTER STATEMENT IS

BEGIN

SELECT COUNT(\*) INTO numar\_ramas

FROM lista\_favorite;

numar\_sters:=total-numar\_ramas;

DBMS\_OUTPUT.PUT\_LINE('Numarul de liste sterse este: '|| numar\_sters);

END AFTER STATEMENT;

END Exercitiul\_11;

/

--Nu poate sterge, elementul nu e marcat ca fiind sters

DELETE FROM lista\_favorite

where id\_lista=1;

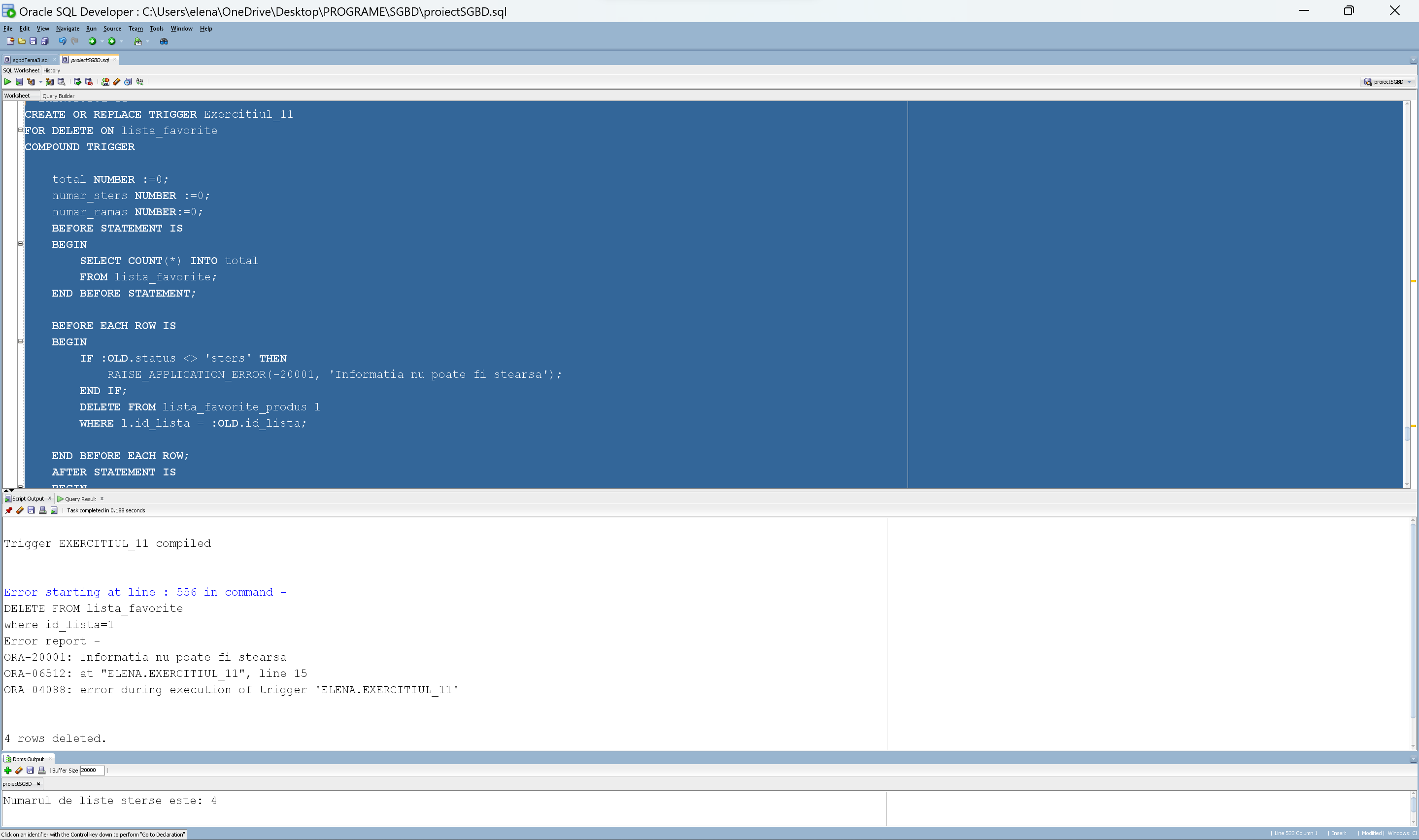
--stergerea se efectueaza cu succes

DELETE FROM lista\_favorite

where status='sters';

rollback;

DROP TRIGGER Exercitiul\_11;



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

Creați un trigger care la orice ștergere sau creare făcută contorizează numărul tabelelor din baza de date și afișează un avertisment atunci când numărul de tabele atinge un anumit prag (10 tabele).

CREATE OR REPLACE TRIGGER Exercitiul\_12

AFTER CREATE OR DROP ON SCHEMA

DECLARE

nr\_tabele NUMBER;

BEGIN

SELECT COUNT(\*)

INTO nr\_tabele

FROM user\_tables;

IF nr\_tabele > 10 THEN

DBMS\_OUTPUT.PUT\_LINE('Atentie: Numarul total de tabele este peste 10.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Numarul de tabele este: '|| nr\_tabele);

END IF;

END Exercitiul\_12;

/

CREATE TABLE tabel\_test(id INT);

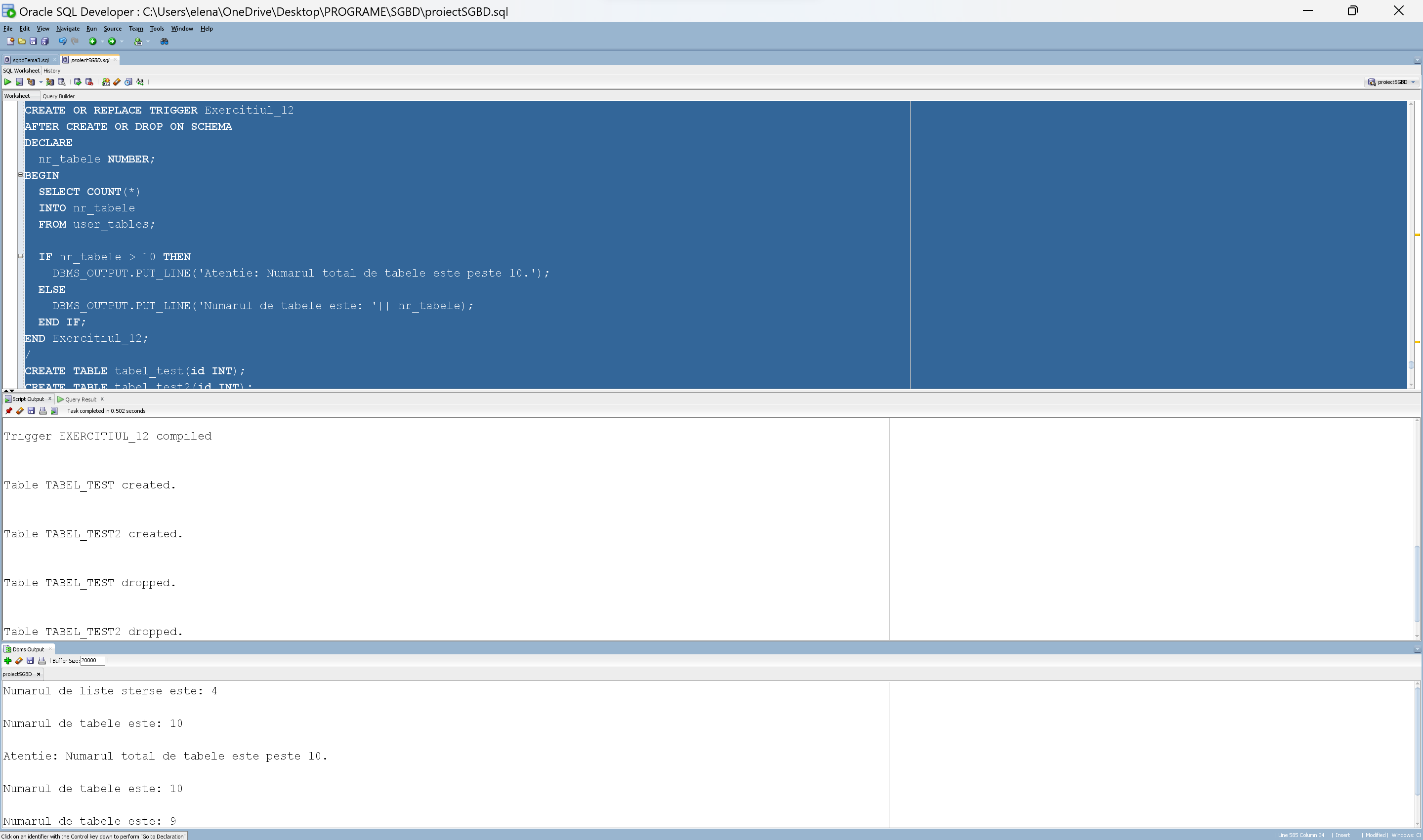
CREATE TABLE tabel\_test2(id INT);

DROP TABLE tabel\_test;

DROP TABLE tabel\_test2;

rollback;

drop trigger exercitiul\_12;



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

CREATE OR REPLACE PACKAGE pachet\_ex\_13

AS

PROCEDURE Exercitiul\_6;

PROCEDURE Exercitiul\_7;

FUNCTION Exercitiul\_8(p\_data\_inceput DATE, p\_data\_sfarsit DATE) RETURN VARCHAR2;

PROCEDURE Exercitiul\_9(p\_oras IN VARCHAR2);

END pachet\_ex\_13;

/

CREATE OR REPLACE PACKAGE BODY pachet\_ex\_13 AS

--EX6 PACHET

PROCEDURE exercitiul\_6 IS

TYPE tabel\_produse IS TABLE OF VARCHAR2(50);

TYPE coduri\_utilizator IS VARRAY(50) OF utilizator.id\_utilizator%TYPE;

TYPE tabel\_pret\_produse IS TABLE OF NUMBER INDEX BY PLS\_INTEGER;

produse\_nevandute tabel\_produse := tabel\_produse();

cod\_utilizatori coduri\_utilizator := coduri\_utilizator();

suma\_produse\_fav tabel\_pret\_produse;

utilizator\_fara\_vanzari utilizator.id\_utilizator%TYPE;

ct int := 1;

BEGIN

SELECT id\_utilizator INTO utilizator\_fara\_vanzari

FROM (

SELECT id\_utilizator, COUNT(id\_produs) AS numar\_produse

FROM produs

WHERE id\_tranzactie IS NULL

GROUP BY id\_utilizator

ORDER BY numar\_produse DESC

)

WHERE ROWNUM = 1;

FOR cod IN (SELECT l.id\_utilizator

FROM lista\_favorite l

JOIN utilizator u ON l.id\_utilizator = u.id\_utilizator

WHERE l.data\_creare = u.data\_inregistrare) LOOP

cod\_utilizatori.EXTEND;

cod\_utilizatori(ct) := cod.id\_utilizator;

ct := ct + 1;

END LOOP;

--stiu ca cod\_utilizatori si suma+produse\_fav au acelasi nr de elem, nu mai am nev de ct

FOR i IN cod\_utilizatori.FIRST..cod\_utilizatori.LAST LOOP

SELECT SUM(p.pret) INTO suma\_produse\_fav(i)

FROM produs p

JOIN lista\_favorite\_produs lf ON lf.id\_produs = p.id\_produs

JOIN lista\_favorite l ON lf.id\_lista = l.id\_lista

WHERE l.id\_utilizator = cod\_utilizatori(i);

END LOOP;

FOR i IN suma\_produse\_fav.FIRST..suma\_produse\_fav.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Suma preturilor pentru Id-ul '|| COD\_UTILIZATORI(i) || ' este ' || suma\_produse\_fav(i));

END LOOP;

FOR produs IN (SELECT firma, marime, culoare, calitate, sex

FROM produs

WHERE id\_utilizator = utilizator\_fara\_vanzari) LOOP

produse\_nevandute.EXTEND;

produse\_nevandute(produse\_nevandute.LAST) := produs.firma || ', ' || produs.marime || ', ' ||

produs.culoare || ', ' || produs.calitate || ', ' ||

produs.sex;

END LOOP;

FOR i IN produse\_nevandute.FIRST..produse\_nevandute.LAST LOOP

DBMS\_OUTPUT.PUT\_LINE('Informatii produs: ');

DBMS\_OUTPUT.PUT\_LINE(produse\_nevandute(i));

END LOOP;

END exercitiul\_6;

--EX 7 PACHET

PROCEDURE Exercitiul\_7 IS

CURSOR c\_utilizatori\_cu\_recenzii IS

SELECT DISTINCT u.id\_utilizator, u.nume, u.prenume

FROM utilizator u

JOIN tranzactie t ON u.id\_utilizator = t.id\_cumparator

JOIN recenzie r ON t.id\_tranzactie = r.id\_tranzactie;

CURSOR c\_recenzii (p\_id\_utilizator NUMBER) IS

SELECT r.evaluare\_produs, r.evaluare\_relationare\_utilizatori, r.rating

FROM recenzie r

JOIN tranzactie t ON r.id\_tranzactie = t.id\_tranzactie

WHERE t.id\_cumparator = p\_id\_utilizator AND r.id\_utilizator = p\_id\_utilizator;

v\_utilizator c\_utilizatori\_cu\_recenzii%ROWTYPE;

v\_recenzie c\_recenzii%ROWTYPE;

BEGIN

OPEN c\_utilizatori\_cu\_recenzii;

LOOP

FETCH c\_utilizatori\_cu\_recenzii INTO v\_utilizator;

EXIT WHEN c\_utilizatori\_cu\_recenzii%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Utilizatorul: ' || v\_utilizator.nume || ' ' || v\_utilizator.prenume);

-- Deschiderea cursorului parametrizat pentru recenzii

OPEN c\_recenzii(v\_utilizator.id\_utilizator);

LOOP

FETCH c\_recenzii INTO v\_recenzie;

EXIT WHEN c\_recenzii%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(' Recenzie: Evaluare Produs - ' || v\_recenzie.evaluare\_produs ||

'; Evaluare Relationare - ' || v\_recenzie.evaluare\_relationare\_utilizatori ||

'; Rating - ' || v\_recenzie.rating);

END LOOP;

CLOSE c\_recenzii;

END LOOP;

CLOSE c\_utilizatori\_cu\_recenzii;

END Exercitiul\_7;

--EX8 PACHET

FUNCTION Exercitiul\_8(p\_data\_inceput DATE, p\_data\_sfarsit DATE)

RETURN VARCHAR2 IS

denumire\_categorie VARCHAR2(50);

max\_vanzari INT;

data\_invalida EXCEPTION;

interval\_prea\_mare EXCEPTION;

BEGIN

IF p\_data\_inceput > p\_data\_sfarsit THEN

RAISE data\_invalida;

END IF;

IF p\_data\_sfarsit - p\_data\_inceput > 365 THEN

RAISE interval\_prea\_mare;

END IF;

SELECT c.denumire INTO denumire\_categorie

FROM categorie c

JOIN categorie\_produs cp ON c.id\_categorie = cp.id\_categorie

JOIN produs p ON cp.id\_produs = p.id\_produs

JOIN tranzactie t ON p.id\_tranzactie = t.id\_tranzactie

WHERE t.data\_efectuarii BETWEEN p\_data\_inceput AND p\_data\_sfarsit

GROUP BY c.denumire

ORDER BY COUNT(\*) DESC, c.denumire

FETCH FIRST 1 ROW ONLY;

RETURN denumire\_categorie;

EXCEPTION

WHEN data\_invalida THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Data de inceput este mai mare decât data de sfarsit.');

RETURN NULL;

WHEN interval\_prea\_mare THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Intervalul de timp specificat este prea mare.');

RETURN NULL;

END Exercitiul\_8;

--EX 9 PACHET

PROCEDURE Exercitiul\_9(

p\_oras IN VARCHAR2

) IS

nume\_utilizator VARCHAR2(100);

prenume\_utilizator VARCHAR2(100);

castig\_total NUMBER;

utilizator\_fara\_vanzari EXCEPTION;

verif\_oras INT;

medie\_recenzii NUMBER;

BEGIN

-- verificam daca orasul este unic: daca codul nu e corect se duce pe eroarea too many rows

SELECT id\_adresa INTO verif\_oras

FROM adresa

WHERE oras = p\_oras;

-- selectam utilizatorul care indeplineste conditiile

SELECT u.nume, u.prenume, SUM(p.pret), AVG(r.rating)

INTO nume\_utilizator, prenume\_utilizator, castig\_total, medie\_recenzii

FROM utilizator u

JOIN adresa a ON u.id\_adresa = a.id\_adresa

JOIN tranzactie t ON u.id\_utilizator = t.id\_vanzator

LEFT JOIN produs p ON t.id\_tranzactie = p.id\_tranzactie

LEFT JOIN recenzie r ON t.id\_tranzactie = r.id\_tranzactie AND t.id\_cumparator = r.id\_utilizator

WHERE a.oras = p\_oras

GROUP BY u.nume, u.prenume

HAVING COUNT(p.id\_produs) >= 2 AND AVG(r.rating)> 2

FETCH FIRST 1 ROW ONLY;

DBMS\_OUTPUT.PUT\_LINE('Nume: ' || nume\_utilizator || ', Prenume: ' || prenume\_utilizator || ', Castig total: ' || castig\_total || ', Medie recenzii: ' || medie\_recenzii);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

IF verif\_oras IS NULL THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Nu exista orasul ' || p\_oras || ' in baza de date.');

ELSE DBMS\_OUTPUT.PUT\_LINE('Eroare: Nu exista utilizator in orasul ' || p\_oras || ' care sa indeplineasca conditiile.');

END IF;

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Eroare: Exista mai multi utilizatori care locuiesc in orasul ' || p\_oras || '.');

END Exercitiul\_9;

END;

/

