Potcoveanu Florin-Razvan

Grupa 232

Proiect Sisteme de Gestiune a Bazelor de Date

Gestiunea unui lant de librarii

Cuprins

1. Prezentați pe scurt baza de date (utilitatea ei).
2. Realizațidiagramaentitate-relație (ERD).
3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.
4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).
5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).
6. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.
7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.
8. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.
9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat 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.
10. Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul.
11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.
12. Definiți un trigger de tip LDD. Declanșați trigger-ul.
13. Definiți un pachet care să conțină toate obiectele definite în cadrul proiectului.
14. Prezentati pe scurt baza de date (utilitatea ei).

Modelul bazei de date are ca scop ușurarea procesului de gestionare a informațiilor unui lanț de librării cat și ușurarea accesului la diferite informații. Baza de date pune la dispoziția utilizatorului informații despre stocul și prețul cărților în diferite librării, angajații din fiecare librărie cat și despre bonurile eliberate de către casieri.

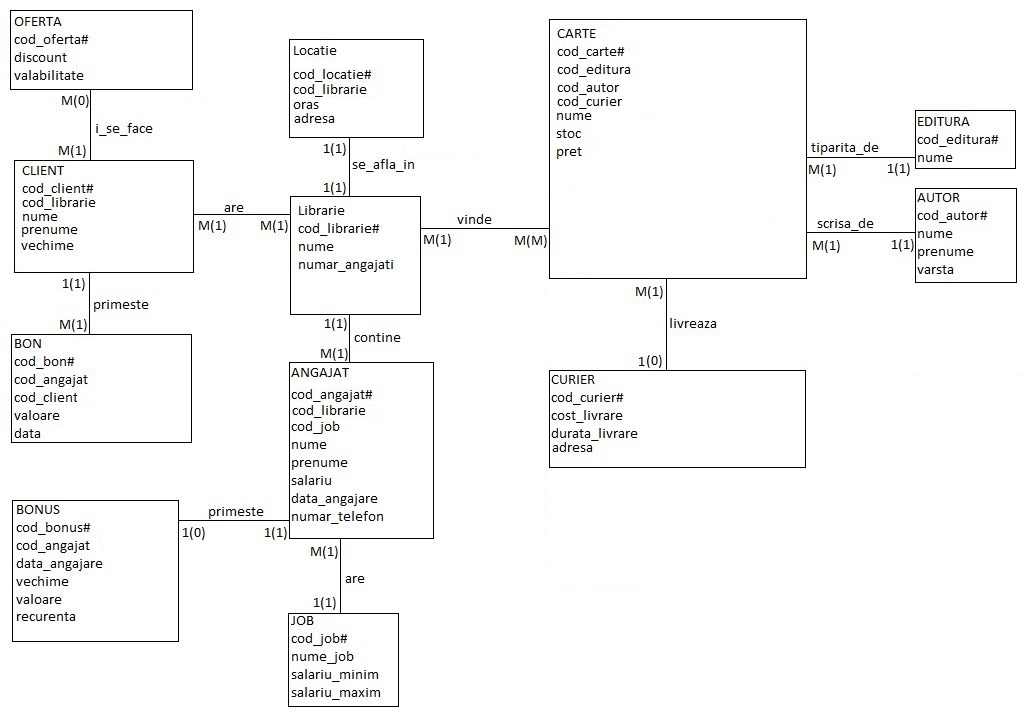
Angajații unei librării pot avea diferite job-uri: șef, casier, îngrijitor, paznic sau gestionar marfă. Clienții primesc un bon pentru fiecare achiziție de una sau mai multe cărți.

Clientul cumpără o carte (care poate fi vândută de mai multe librării) pentru care va primi un bon, eliberat de către un angajat. Un client poate primi mai multe bonuri dacă efectuează mai multe achiziții și i se poate face o ofertă telefonică sau în magazin. Cartea poate fi scrisă de un singur autor, tipărită de o singură editură si repartizată unui singur curier.

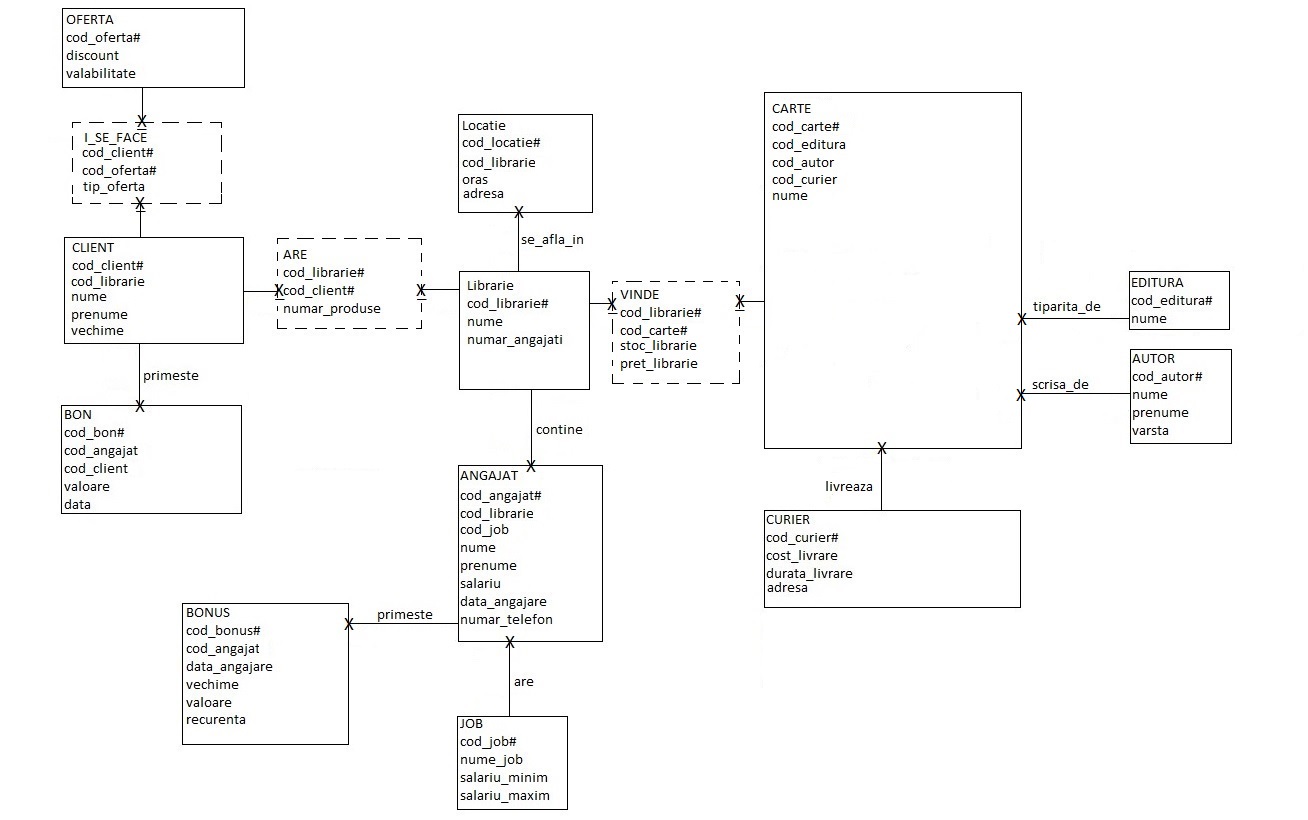
Constrângerile impuse asupra modelului:

* Librăria are un nume și un număr de angajați, se poate afla într-o singură locație, poate vinde mai multe cărți, poate avea mai mulți clienți și poate avea mai mulți angajați .
* Clientul primește un bon pentru produsele cumpărate (un client poate primi mai multe bonuri). Clientului i se poate face si o ofertă telefonică sau în magazin (un client poate primi mai multe oferte).
* O ofertă conține un discount si un termen de valabilitate. O ofertă poate fi făcută mai multor clienți.
* Un bon poate fi eliberat unui singur client, dar un client poate primi mai multe bonuri.
* Angajatul poate lucra într-o singură librărie, poate primi un singur bonus și poate avea un singur job.
* Bonusul poate fi oferit unui singur angajat.
* Pe un job pot lucra mai mulți angajați.
* O carte poate fi vândută în mai multe librării, poate fi repartizată de un singur curier, poate fi tipărită de o singură editură si poate fi scrisă de un singur autor.
* Curierul poate fi repartizat la mai multe cărți.
* Editura poate tipării mai multe cărți.
* Autorul poate scrie mai multe cărți.

1. Realizati diagrama entitate-relatie (ERD).



1. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.



1. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).
2. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

alter session set NLS\_DATE\_FORMAT='DD-MM-YYYY';

--LIBRARIE

create table LIBRARIE (cod\_librarie number(3), nume varchar(25) not null, numar\_angajati number(3) not null, PRIMARY KEY(cod\_librarie));

insert into LIBRARIE values (1, 'Mihai Eminescu', 5);

insert into LIBRARIE values (2, 'Sophia', 6);

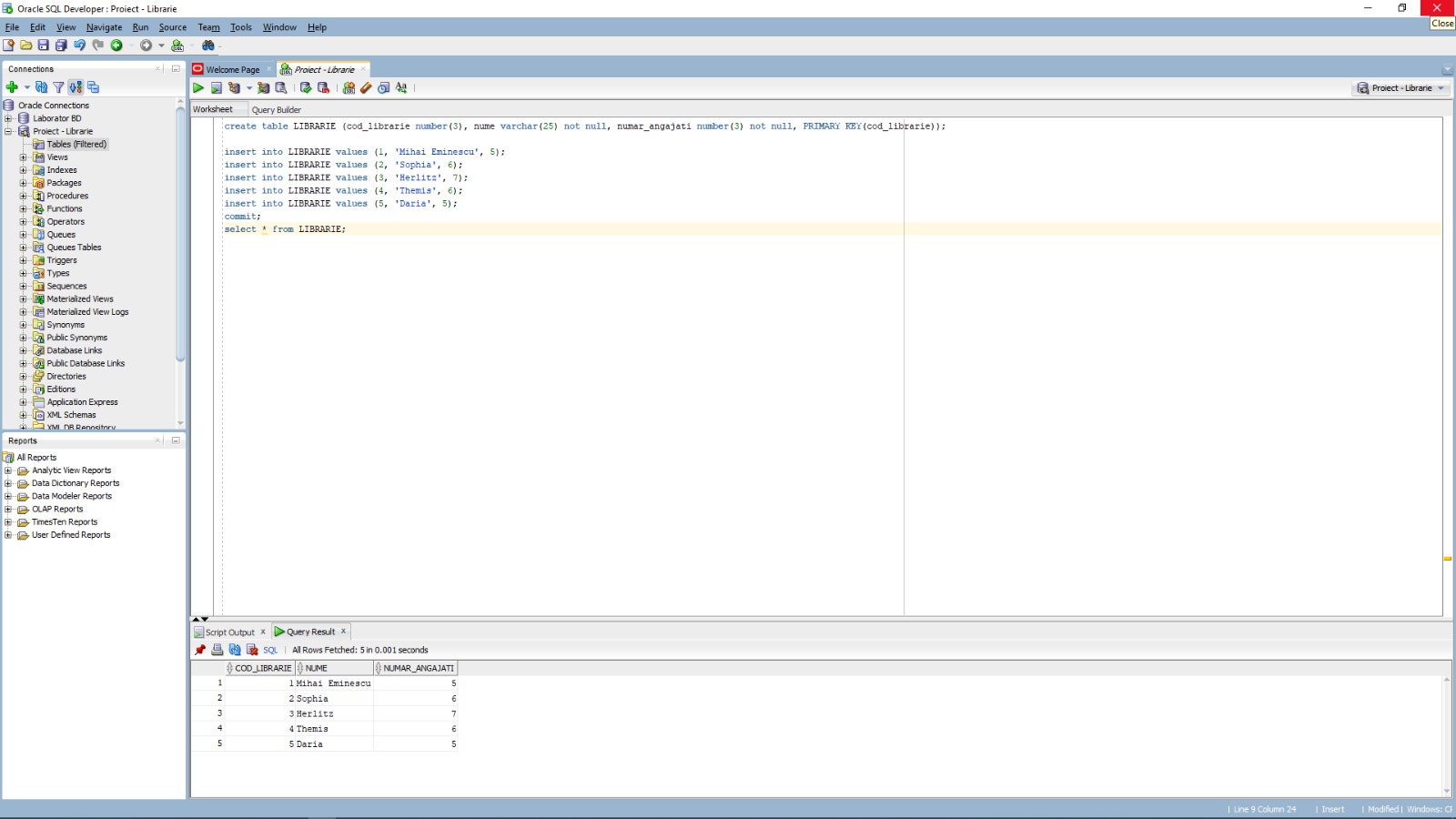
insert into LIBRARIE values (3, 'Herlitz', 7);

insert into LIBRARIE values (4, 'Themis', 6);

insert into LIBRARIE values (5, 'Daria', 5);

commit;

select \* from LIBRARIE;



--LOCATIE

create table LOCATIE (cod\_locatie number(3), cod\_librarie number(3) not null, oras varchar(25) not null, adresa varchar(50) not null, PRIMARY KEY(cod\_locatie), FOREIGN KEY(cod\_librarie) REFERENCES LIBRARIE(cod\_librarie));

insert into LOCATIE values(101, 1, 'Pitesti', 'Strada Victoriei 89');

insert into LOCATIE values(102, 2, 'Pitesti', 'Bulevardul I. C. Bratianu, Bloc B5');

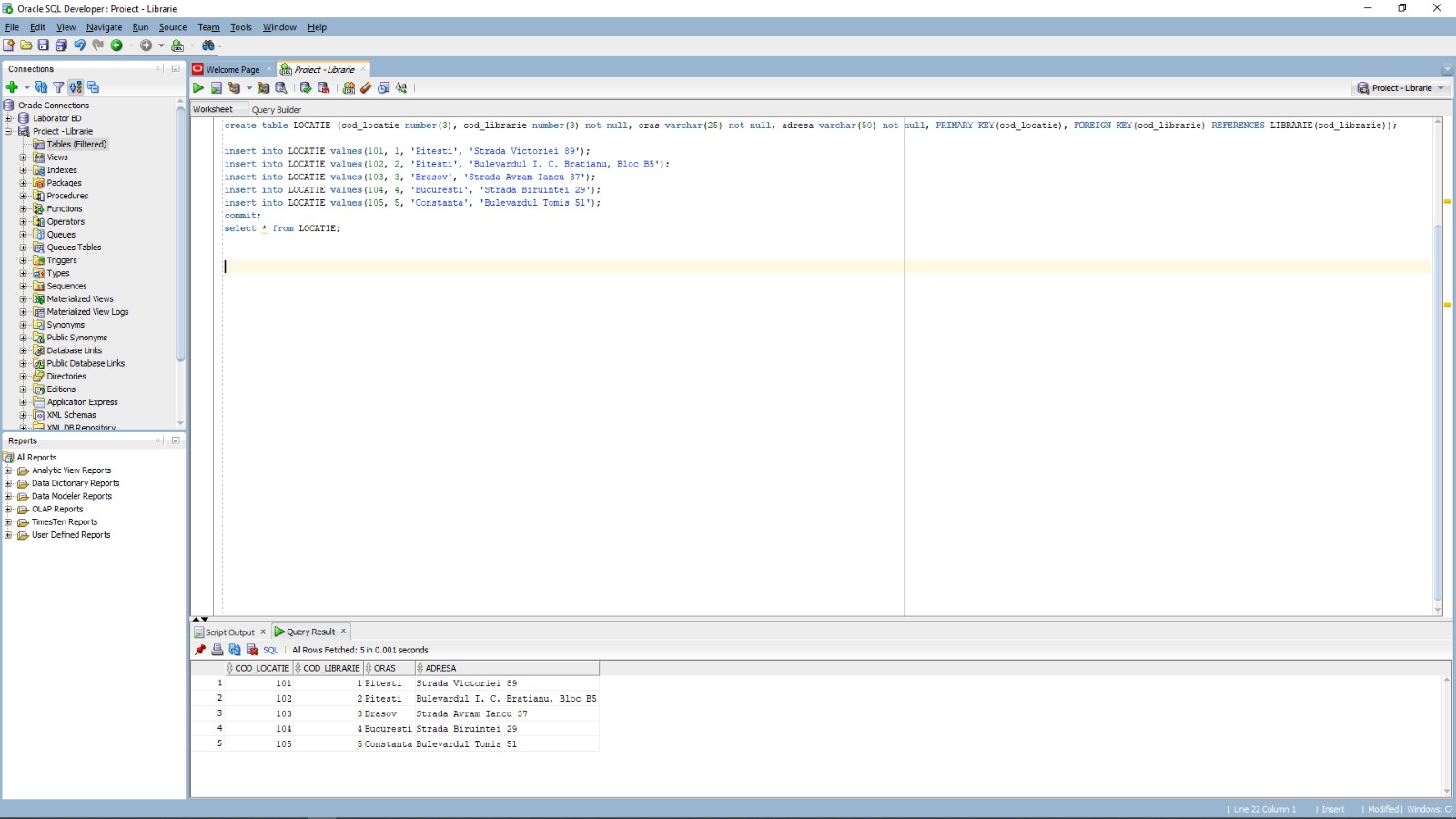
insert into LOCATIE values(103, 3, 'Brasov', 'Strada Avram Iancu 37');

insert into LOCATIE values(104, 4, 'Bucuresti', 'Strada Biruintei 29');

insert into LOCATIE values(105, 5, 'Constanta', 'Bulevardul Tomis 51');

commit;

select \* from LOCATIE;



--CLIENT

create table CLIENT (cod\_client number(3), cod\_librarie number(3) not null, nume varchar(25) not null, prenume varchar(25) not null, vechime date not null, PRIMARY KEY(cod\_client),

FOREIGN KEY(cod\_librarie) REFERENCES LIBRARIE(cod\_librarie));

insert into CLIENT values(100, 1, 'Potcoveanu', 'Florin-Razvan', '20-12-2020');

insert into CLIENT values(101, 1, 'Nidelea', 'Gabriela-Andreea', '03-01-2021');

insert into CLIENT values(102, 1, 'Popescu', 'Mihnea-Valentin', '15-08-2020');

insert into CLIENT values(103, 2, 'Popa', 'Andrei-Marian', '07-04-2020');

insert into CLIENT values(104, 2, 'Ionescu', 'Stefan-Laurentiu', '04-04-2021');

insert into CLIENT values(105, 2, 'Stroe', 'Adrian', '13-12-2020');

insert into CLIENT values(106, 3, 'Iancu', 'Andrei-Cosmin', '17-11-2020');

insert into CLIENT values(107, 3, 'Velcea', 'Radu-Constantin', '29-11-2020');

insert into CLIENT values(108, 3, 'Dumitru', 'Ana-Maria', '05-11-2020');

insert into CLIENT values(109, 4, 'Stancu', 'Madalin-George', '28-10-2020');

insert into CLIENT values(110, 4, 'Comanescu', 'Ioana-Alexandra', '25-09-2020');

insert into CLIENT values(111, 4, 'Ghiordanescu', 'Raluca-Georgiana', '09-10-2020');

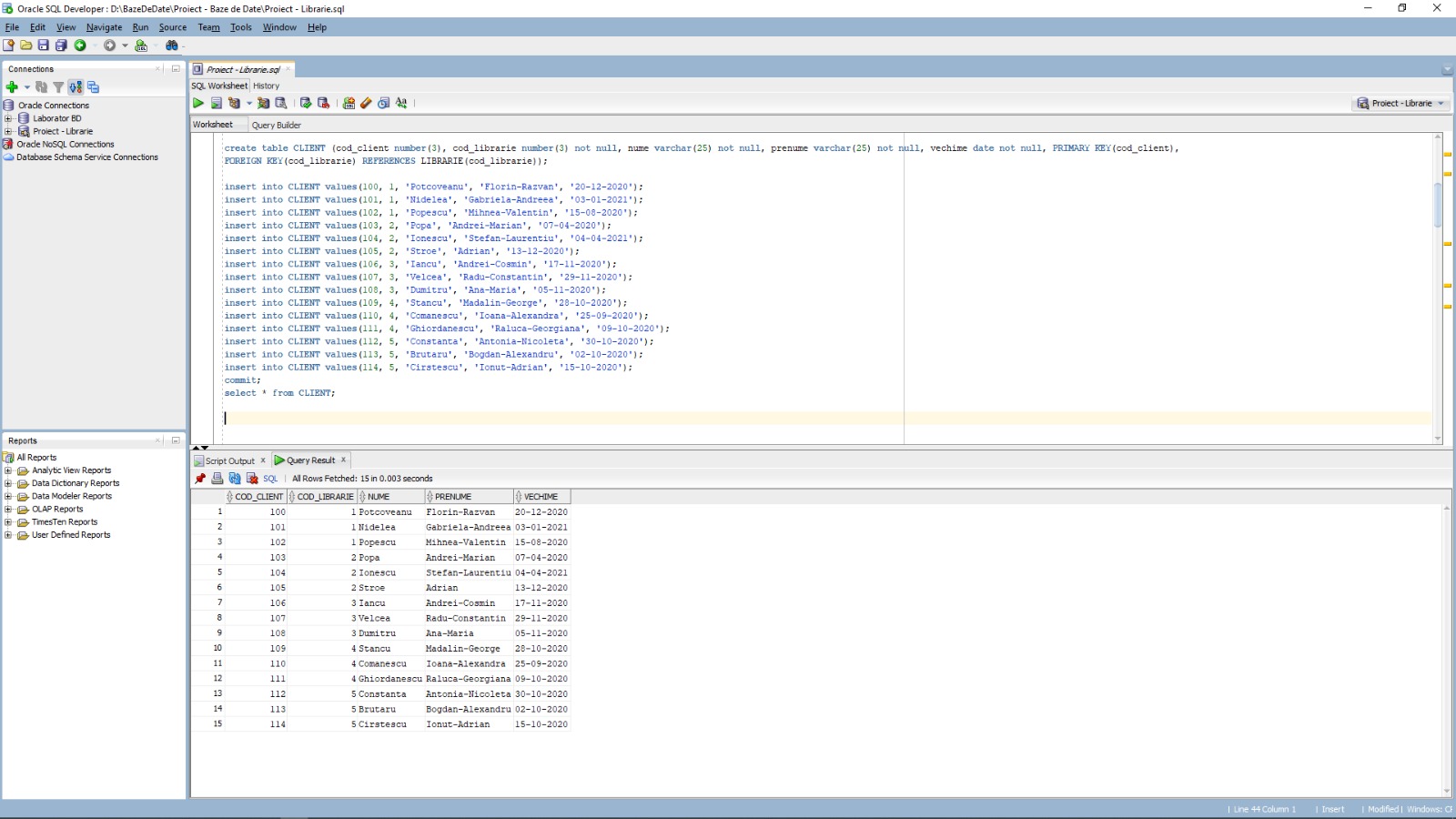
insert into CLIENT values(112, 5, 'Constanta', 'Antonia-Nicoleta', '30-10-2020');

insert into CLIENT values(113, 5, 'Brutaru', 'Bogdan-Alexandru', '02-10-2020');

insert into CLIENT values(114, 5, 'Cirstescu', 'Ionut-Adrian', '15-10-2020');

commit;

select \* from CLIENT;



--JOB

create table JOB (cod\_job number(3), nume\_job varchar(15) not null, salaiu\_minim number(7) not null, salariu\_maxim number(7) not null, PRIMARY KEY(cod\_job));

insert into JOB values(200, 'Sef', 10000, 15000);

insert into JOB values(201, 'Casier', 2000, 3500);

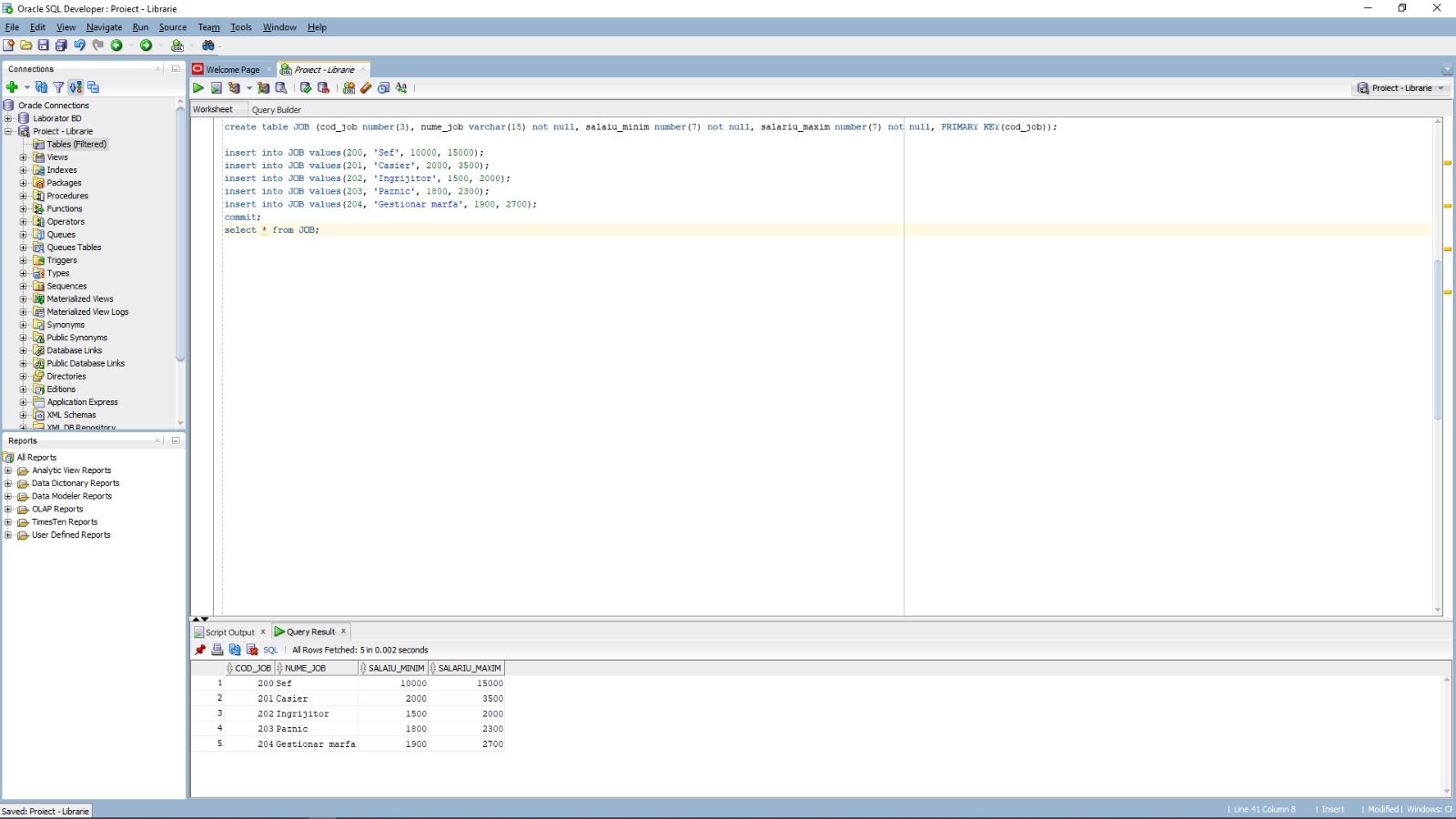
insert into JOB values(202, 'Ingrijitor', 1500, 2000);

insert into JOB values(203, 'Paznic', 1800, 2300);

insert into JOB values(204, 'Gestionar marfa', 1900, 2700);

commit;

select \* from JOB;



--ANGAJAT

create table ANGAJAT (cod\_angajat number(3), cod\_librarie number(3), cod\_job number(3), nume varchar(25), prenume varchar(25), salariu number(7), data\_angajare date, numar\_telefon varchar(10), PRIMARY KEY(cod\_angajat),

FOREIGN KEY(cod\_librarie) REFERENCES LIBRARIE(cod\_librarie), FOREIGN KEY(cod\_job) REFERENCES JOB(cod\_job));

insert into ANGAJAT values(100, 1, 200, 'Dumitrescu', 'Florin-Andrei', 12500, '10-10-2019', '0728356891');

insert into ANGAJAT values(101, 1, 201, 'Mocanu', 'Matei', 2100, '17-12-2019', '0761827164');

insert into ANGAJAT values(102, 1, 203, 'Enache', 'Sebastian-Ionut', 2000, '04-01-2020', '0734823178');

insert into ANGAJAT values(103, 2, 200, 'Borcea', 'Antonio-Valentin', 15000, '02-07-2019', '0767524632');

insert into ANGAJAT values(104, 2, 201, 'Dumitru', 'Marian', 2700, '23-08-2019', NULL);

insert into ANGAJAT values(105, 2, 202, 'Arsene', 'Ramona', 1501, '11-09-2019', '0721387258');

insert into ANGAJAT values(106, 3, 200, 'Stanescu', 'Atodiresei', 13000, '27-02-2019', '0738726129');

insert into ANGAJAT values(107, 3, 201, 'Florea', 'Gabita-Maria', 3100, '21-03-2019', NULL);

insert into ANGAJAT values(108, 3, 204, 'Pasca', 'Gheorge', 2300, '24-03-2019', '0736296152');

insert into ANGAJAT values(109, 4, 200, 'Constantinescu', 'Eduard', 14200, '08-04-2019', '0743231252');

insert into ANGAJAT values(110, 4, 201, 'Mihaescu', 'Mihai', 3420, '10-04-2019', NULL);

insert into ANGAJAT values(111, 4, 202, 'Stan', 'Adda', 1600, '12-04-2019', '0775867898');

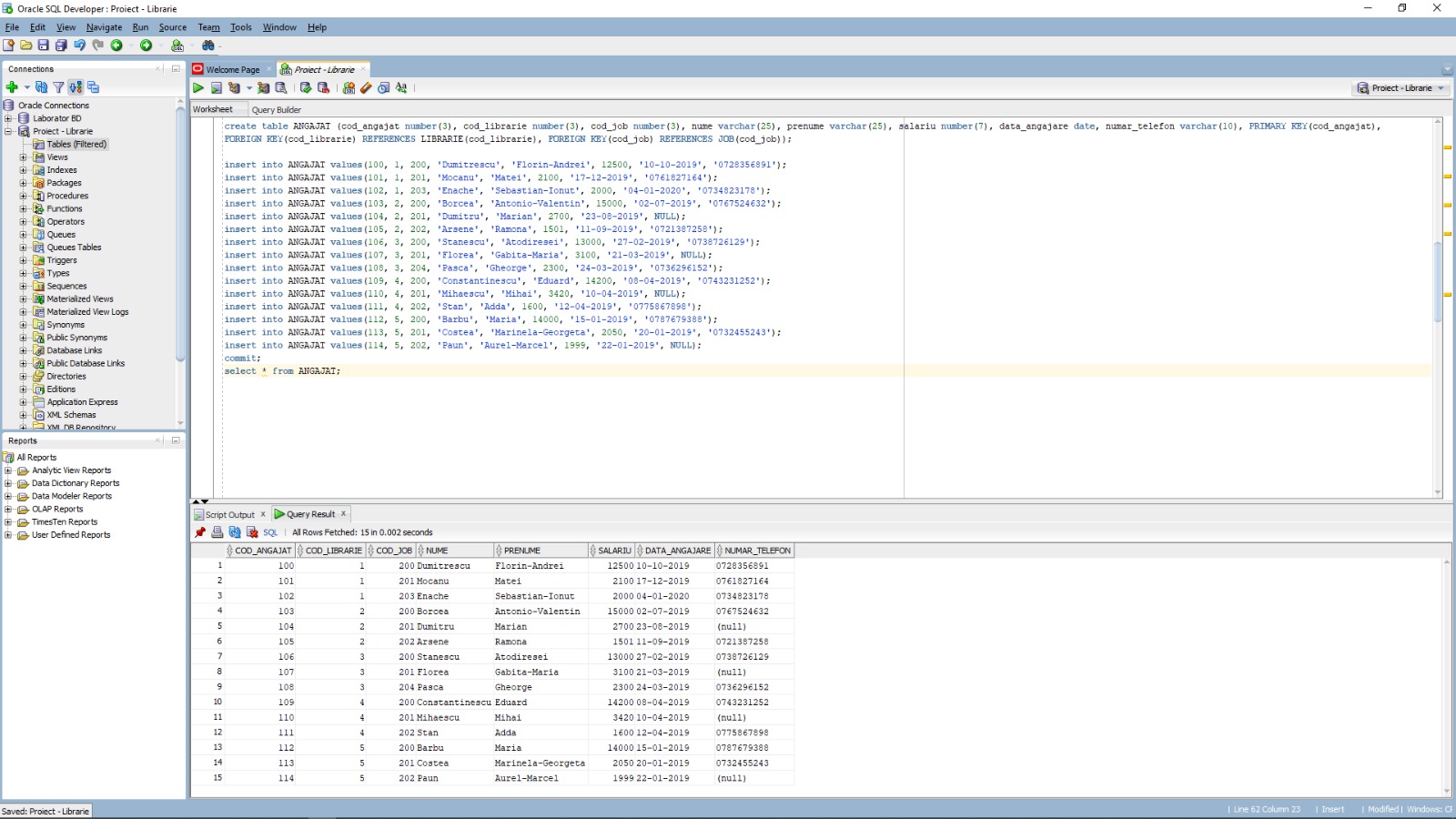
insert into ANGAJAT values(112, 5, 200, 'Barbu', 'Maria', 14000, '15-01-2019', '0787679388');

insert into ANGAJAT values(113, 5, 201, 'Costea', 'Marinela-Georgeta', 2050, '20-01-2019', '0732455243');

insert into ANGAJAT values(114, 5, 202, 'Paun', 'Aurel-Marcel', 1999, '22-01-2019', NULL);

commit;

select \* from ANGAJAT;



--BONUS

create table BONUS (cod\_bonus number(3), cod\_angajat number(3), data\_angajare date, vechime varchar(15), valoare number(5), recurenta varchar(15), PRIMARY KEY(cod\_bonus),

FOREIGN KEY(cod\_angajat) REFERENCES ANGAJAT(cod\_angajat));

insert into BONUS values(900, 102, '04-01-2020', '1 an+', 300, '4 luni');

insert into BONUS values(901, 104, '23-08-2019', '1 an+', 300, '3 luni');

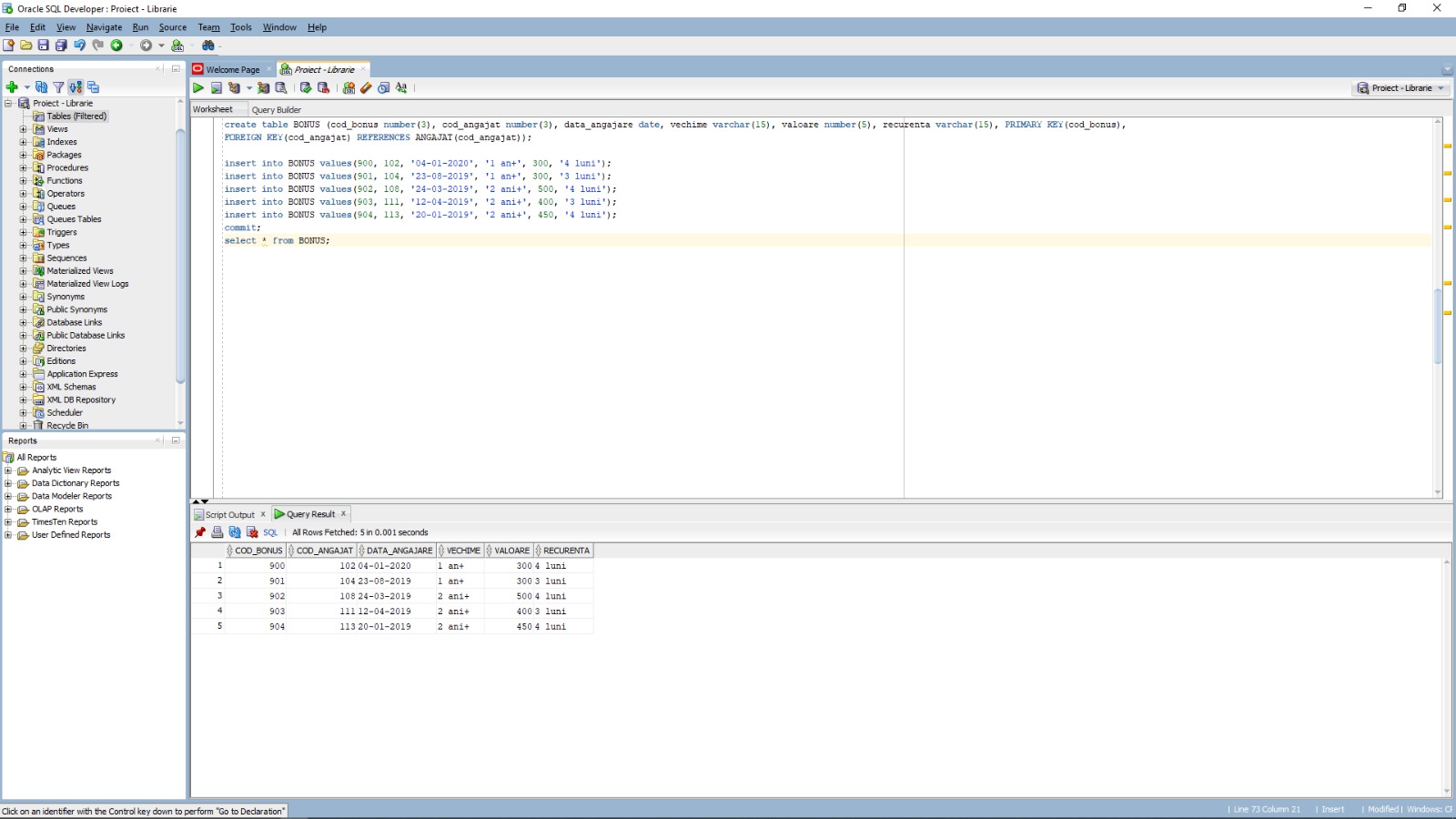
insert into BONUS values(902, 108, '24-03-2019', '2 ani+', 500, '4 luni');

insert into BONUS values(903, 111, '12-04-2019', '2 ani+', 400, '3 luni');

insert into BONUS values(904, 113, '20-01-2019', '2 ani+', 450, '4 luni');

commit;

select \* from BONUS;



--OFERTA

create table OFERTA(cod\_oferta number(3), dicount number(2) not null, valabilitate date not null, PRIMARY KEY(cod\_oferta));

insert into OFERTA values(800, 10, '01-09-2021');

insert into OFERTA values(801, 20, '15-08-2021');

insert into OFERTA values(802, 25, '01-08-2021');

insert into OFERTA values(803, 30, '15-07-2021');

insert into OFERTA values(804, 40, '01-07-2021');

commit;

select \* from OFERTA;



--BON

create table BON(cod\_bon number(3), cod\_angajat number(3) not null, cod\_client number(3) not null, valoare number(5) not null, data date not null, PRIMARY KEY(cod\_bon),

FOREIGN KEY(cod\_angajat) REFERENCES ANGAJAT(cod\_angajat), FOREIGN KEY(cod\_client) REFERENCES CLIENT(cod\_client));

insert into BON values(1, 101, 100, 1500, '01-01-2021');

insert into BON values(2, 104, 105, 750, '23-12-2020');

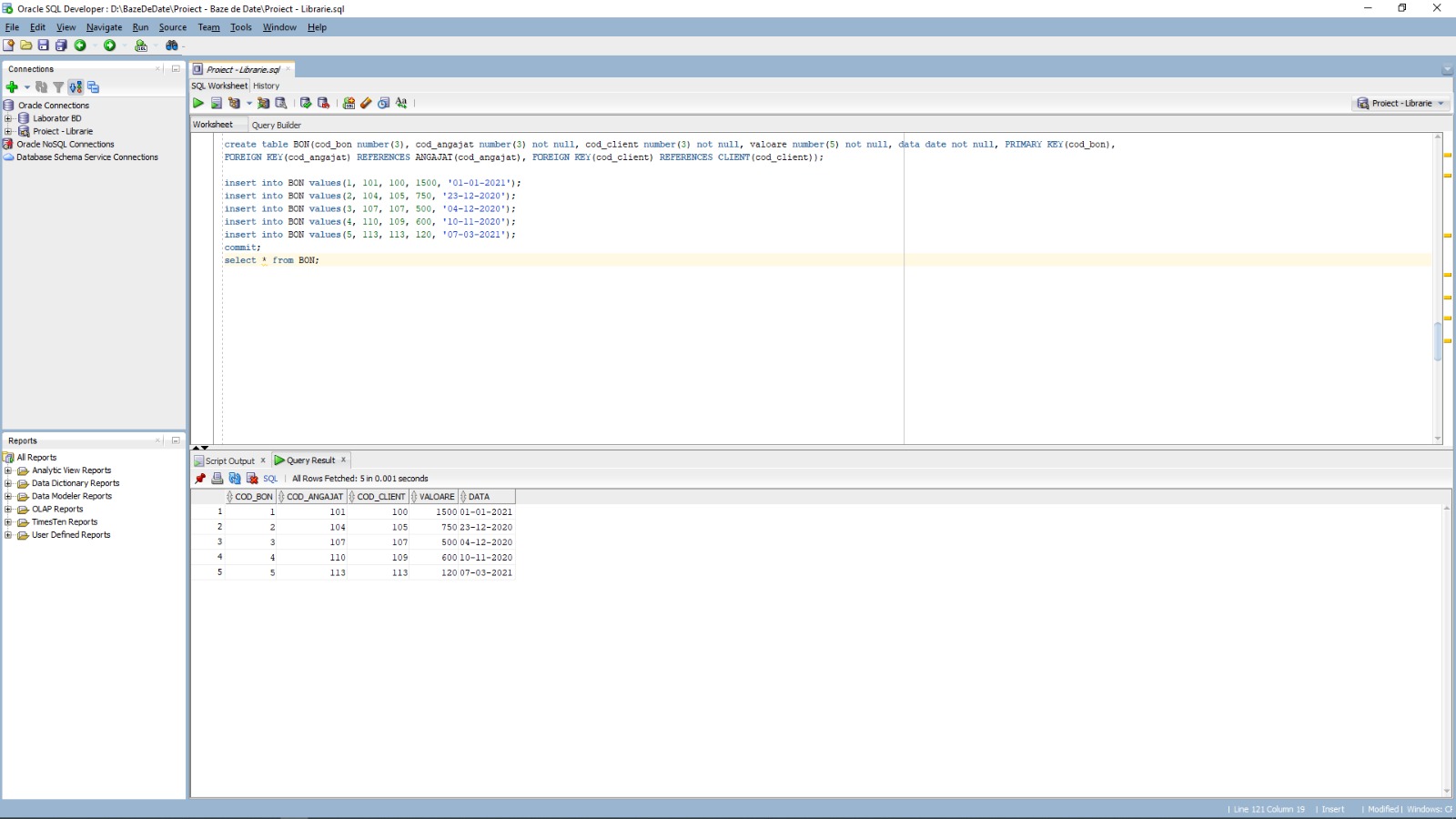
insert into BON values(3, 107, 107, 500, '04-12-2020');

insert into BON values(4, 110, 109, 600, '10-11-2020');

insert into BON values(5, 113, 113, 120, '07-03-2021');

commit;

select \* from BON;



--CURIER

create table CURIER(cod\_curier number(3), cost\_livrare number(4), durata\_livrare number(2) not null, adresa varchar(50) not null, PRIMARY KEY(cod\_curier));

insert into CURIER values(500, 25, 2, 'Calea Dragasani 4');

insert into CURIER values(501, 15, 3, 'Strada Acvariului 20');

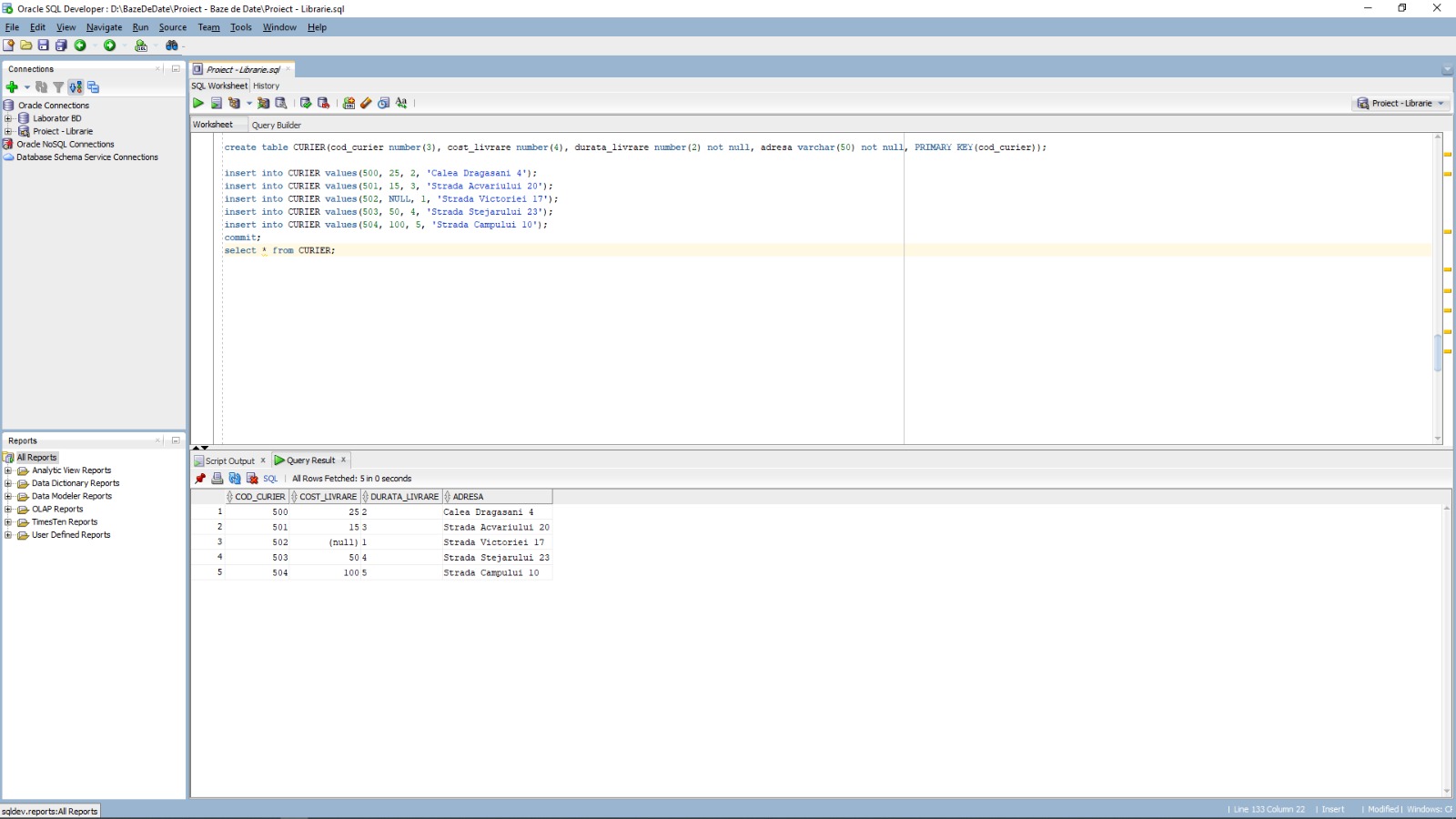
insert into CURIER values(502, NULL, 1, 'Strada Victoriei 17');

insert into CURIER values(503, 50, 4, 'Strada Stejarului 23');

insert into CURIER values(504, 100, 5, 'Strada Campului 10');

commit;

select \* from CURIER;



--EDITURA

create table EDITURA(cod\_editura number(3), nume varchar(25) not null, PRIMARY KEY(cod\_editura));

insert into EDITURA values(700, 'Corint');

insert into EDITURA values(701, 'Carminis');

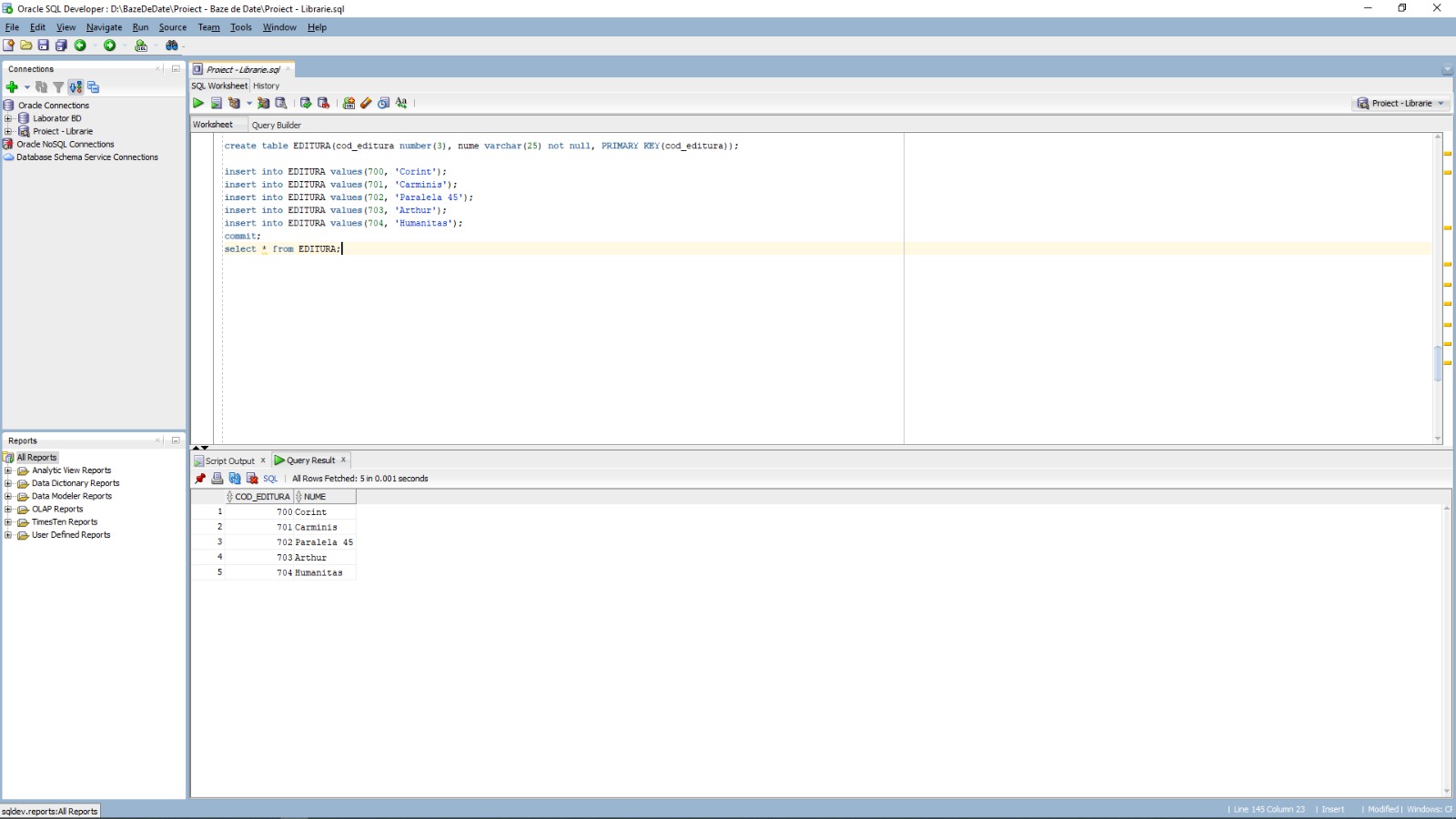
insert into EDITURA values(702, 'Paralela 45');

insert into EDITURA values(703, 'Arthur');

insert into EDITURA values(704, 'Humanitas');

commit;

select \* from EDITURA;



--AUTOR

create table AUTOR(cod\_autor number(3), nume varchar(25) not null, prenume varchar(25) not null, varsta number(2), PRIMARY KEY(cod\_autor));

insert into AUTOR values(300, 'Bishop', 'Gary John', NULL);

insert into AUTOR values(301, 'Kiyosaki', 'Robert Toru', 74);

insert into AUTOR values(302, 'Manson', 'Mark', 37);

insert into AUTOR values(303, 'Voss', 'Christopher', 79);

insert into AUTOR values(304, 'Dutton', 'Kevin', 54);

commit;

select \* from AUTOR;



--CARTE

create table CARTE(cod\_carte number(3), cod\_editura number(3) not null, cod\_autor number(3) not null, cod\_curier number(3), nume varchar(25) not null,

PRIMARY KEY(cod\_carte), FOREIGN KEY (cod\_editura) REFERENCES EDITURA(cod\_editura), FOREIGN KEY (cod\_autor) REFERENCES AUTOR(cod\_autor), FOREIGN KEY (cod\_curier) REFERENCES CURIER(cod\_CURIER));

insert into CARTE values(100, 700, 300, NULL, 'Unf#ck yourself');

insert into CARTE values(101, 701, 301, NULL, 'Tata bogat tata sarac');

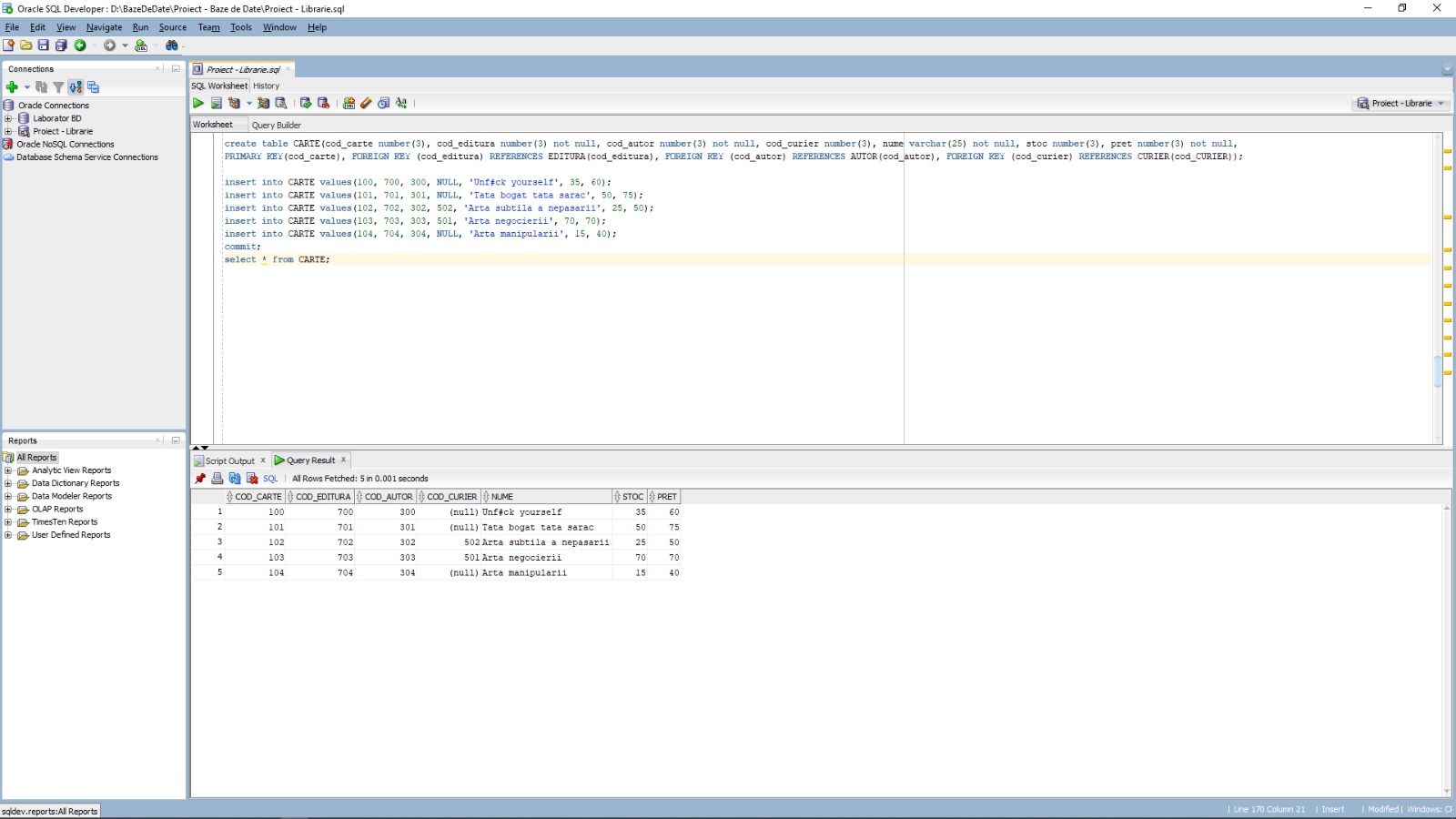
insert into CARTE values(102, 702, 302, 502, 'Arta subtila a nepasarii');

insert into CARTE values(103, 703, 303, 501, 'Arta negocierii');

insert into CARTE values(104, 704, 304, NULL, 'Arta manipularii');

commit;

select \* from CARTE;



--I\_SE\_FACE

create table I\_SE\_FACE(cod\_client number(3), cod\_oferta number(3), tip\_oferta varchar(15) not null, PRIMARY KEY(cod\_client, cod\_oferta),

FOREIGN KEY(cod\_client) REFERENCES CLIENT(cod\_client), FOREIGN KEY(cod\_oferta) REFERENCES OFERTA(cod\_oferta));

insert into I\_SE\_FACE values(100, 804, 'telefonica');

insert into I\_SE\_FACE values(102, 800, 'telefonica');

insert into I\_SE\_FACE values(103, 800, 'in magazin');

insert into I\_SE\_FACE values(104, 801, 'in magazin');

insert into I\_SE\_FACE values(106, 801, 'in magazin');

insert into I\_SE\_FACE values(107, 802, 'telefonica');

insert into I\_SE\_FACE values(110, 802, 'telefonica');

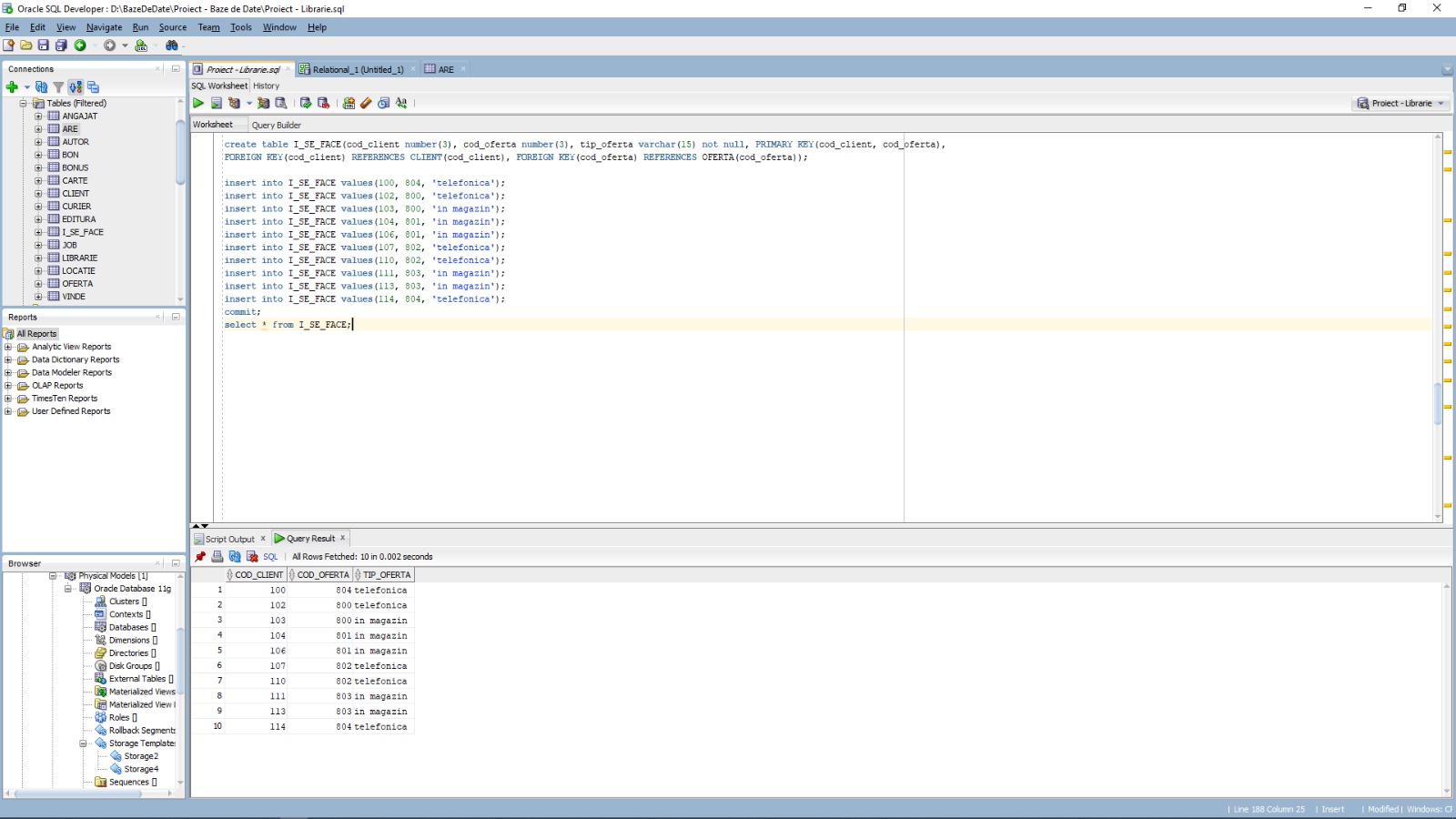
insert into I\_SE\_FACE values(111, 803, 'in magazin');

insert into I\_SE\_FACE values(113, 803, 'in magazin');

insert into I\_SE\_FACE values(114, 804, 'telefonica');

commit;

select \* from I\_SE\_FACE;



--ARE

create table ARE(cod\_librarie number(3), cod\_client number(3), numar\_produse number(2), PRIMARY KEY(cod\_librarie, cod\_client),

FOREIGN KEY(cod\_librarie) REFERENCES LIBRARIE(cod\_librarie), FOREIGN KEY(cod\_client) REFERENCES CLIENT(cod\_client));

insert into ARE values(1, 100, 20);

insert into ARE values(1, 101, 18);

insert into ARE values(2, 104, 5);

insert into ARE values(2, 105, 10);

insert into ARE values(3, 107, 13);

insert into ARE values(3, 108, 7);

insert into ARE values(4, 110, 15);

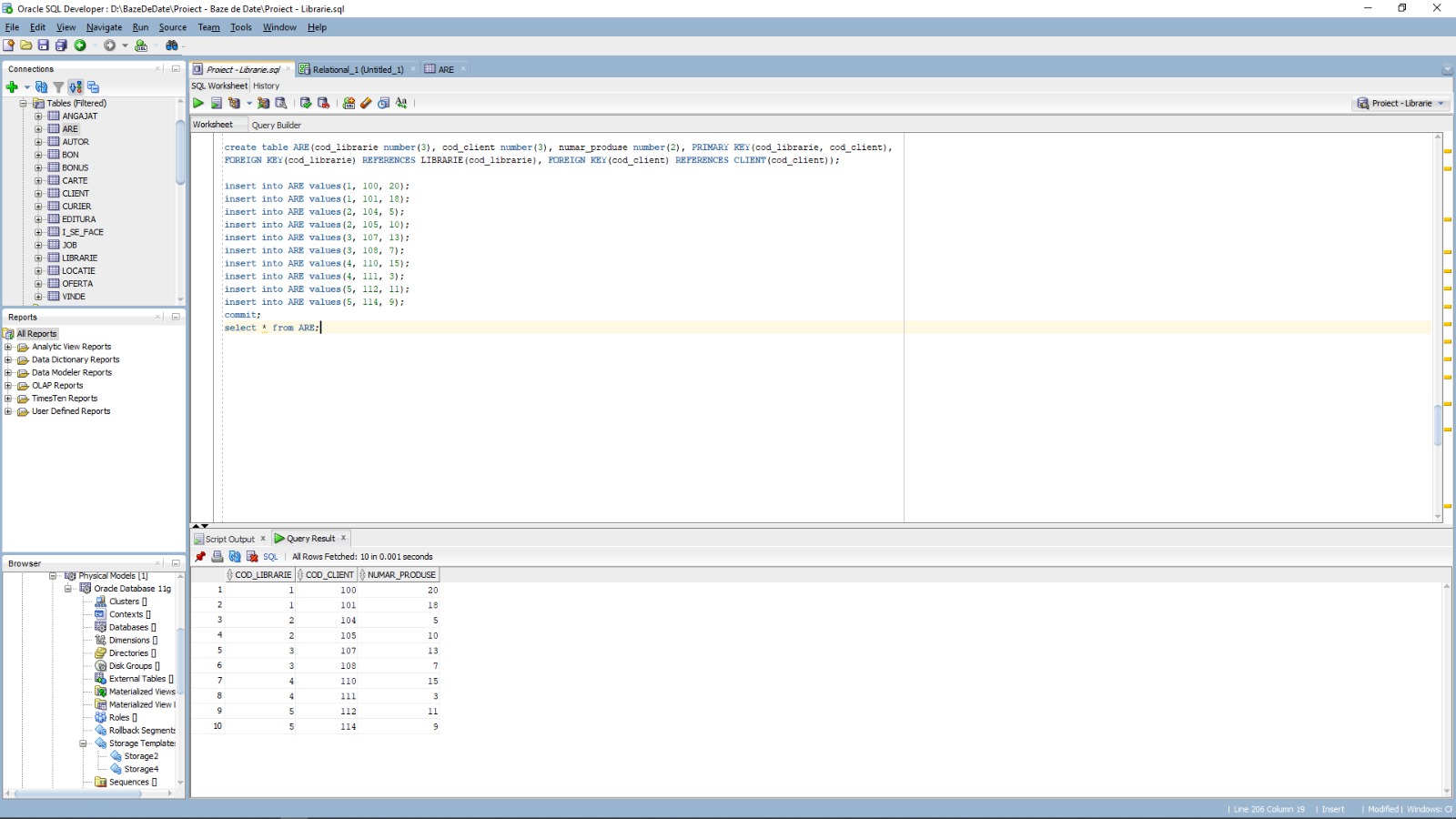
insert into ARE values(4, 111, 3);

insert into ARE values(5, 112, 11);

insert into ARE values(5, 114, 9);

commit;

select \* from ARE;



--VINDE

create table VINDE(cod\_librarie number(3), cod\_carte number(3), stoc\_librarie number(3), pret\_librarie number(3), PRIMARY KEY(cod\_librarie, cod\_carte),

FOREIGN KEY(cod\_librarie) REFERENCES LIBRARIE(cod\_librarie), FOREIGN KEY(cod\_carte) REFERENCES CARTE(cod\_carte));

insert into VINDE values(1, 100, 50, 65);

insert into VINDE values(1, 101, 100, 100);

insert into VINDE values(1, 102, 40, 50);

insert into VINDE values(1, 103, 60, 75);

insert into VINDE values(1, 104, 25, 40);

insert into VINDE values(2, 100, 30, 60);

insert into VINDE values(2, 101, 75, 90);

insert into VINDE values(2, 102, NULL, NULL);

insert into VINDE values(2, 103, 50, 70);

insert into VINDE values(2, 104, NULL, NULL);

insert into VINDE values(3, 100, NULL, NULL);

insert into VINDE values(3, 101, 42, 70);

insert into VINDE values(3, 102, NULL, NULL);

insert into VINDE values(3, 103, NULL, NULL);

insert into VINDE values(3, 104, 27, 50);

insert into VINDE values(4, 100, 23, 70);

insert into VINDE values(4, 101, 50, 99);

insert into VINDE values(4, 102, 38, 48);

insert into VINDE values(4, 103, 63, 72);

insert into VINDE values(4, 104, NULL, NULL);

insert into VINDE values(5, 100, 62, 67);

insert into VINDE values(5, 101, NULL, NULL);

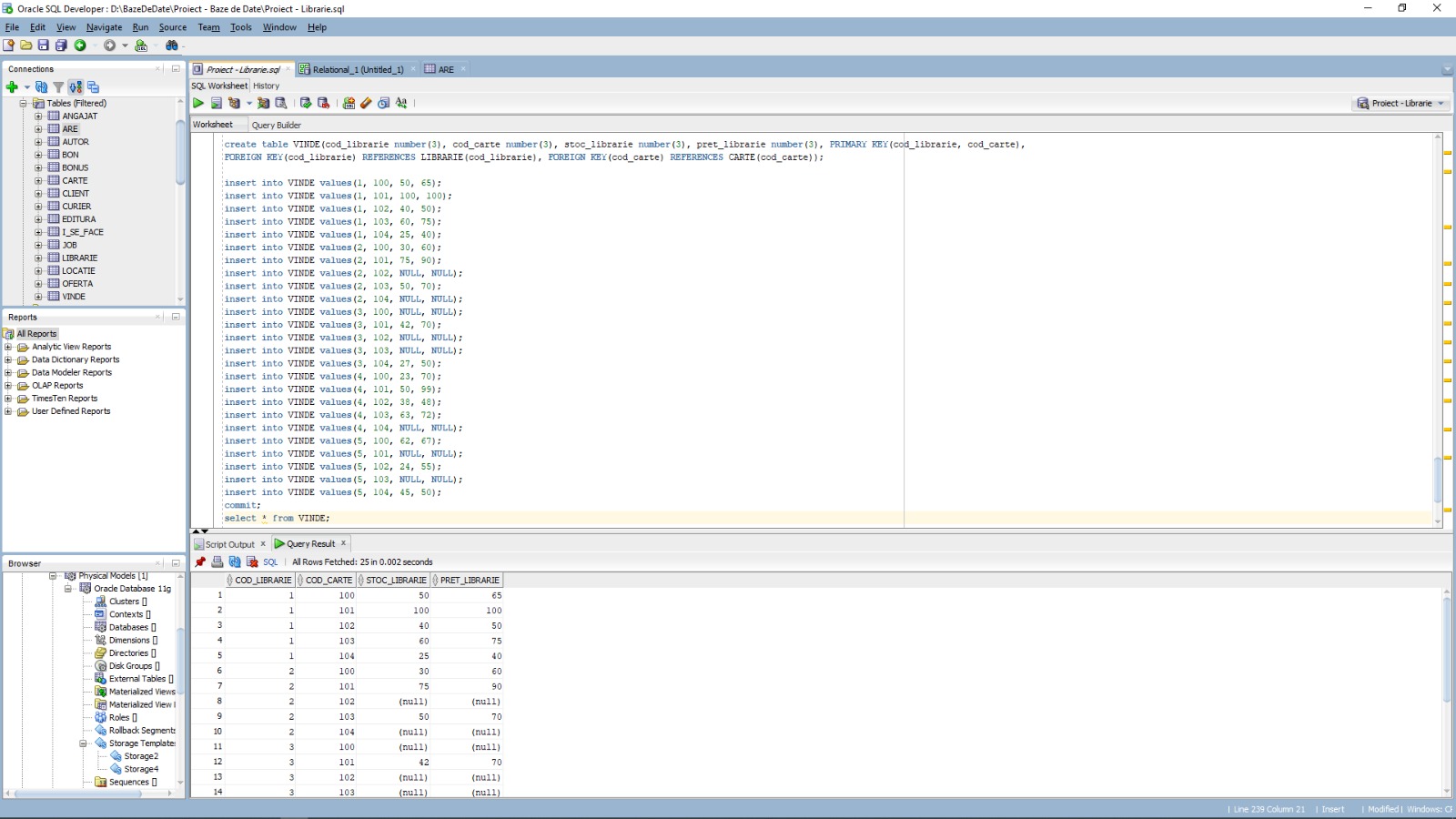
insert into VINDE values(5, 102, 24, 55);

insert into VINDE values(5, 103, NULL, NULL);

insert into VINDE values(5, 104, 45, 50);

commit;

select \* from VINDE;



1. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze două tipuri de colecție studiate. Apelați subprogramul.

/\*

Afisati salariile si datele de angajare unui anumit job din toate librariile cu un cod primit ca input de la user.

\*/

CREATE OR REPLACE

PROCEDURE subprogram\_1 (cod\_1 angajat.cod\_job%TYPE)

IS

TYPE tablou\_imbricat IS TABLE OF angajat.salariu%TYPE;

tablou tablou\_imbricat;

TYPE tablou\_indexat IS TABLE OF angajat.data\_angajare%TYPE INDEX BY PLS\_INTEGER;

tablou\_2 tablou\_indexat;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Job-ul cu codul ' || cod\_1 || ' are salariile : ');

SELECT salariu

BULK COLLECT INTO tablou

FROM angajat

WHERE cod\_job = cod\_1;

SELECT data\_angajare

BULK COLLECT INTO tablou\_2

FROM angajat

WHERE cod\_job = cod\_1;

FOR i in tablou.first .. tablou.last loop

DBMS\_OUTPUT.PUT\_LINE('Salariu ' || tablou(i) || ' si data angajarii ' || tablou\_2(i));

END LOOP;

END subprogram\_1;

/

DECLARE

cod\_aux angajat.cod\_job%TYPE := &cod\_aux;

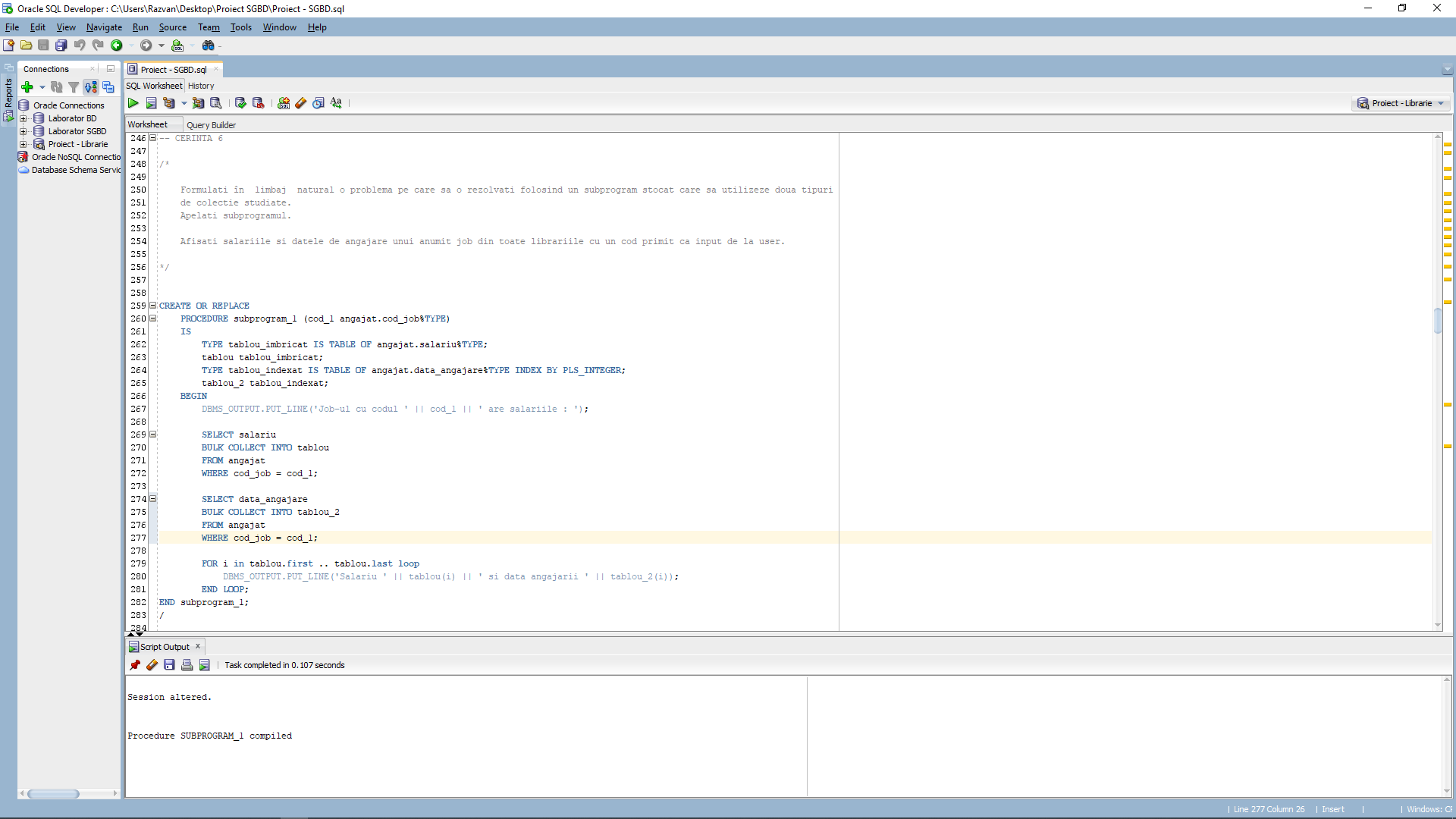
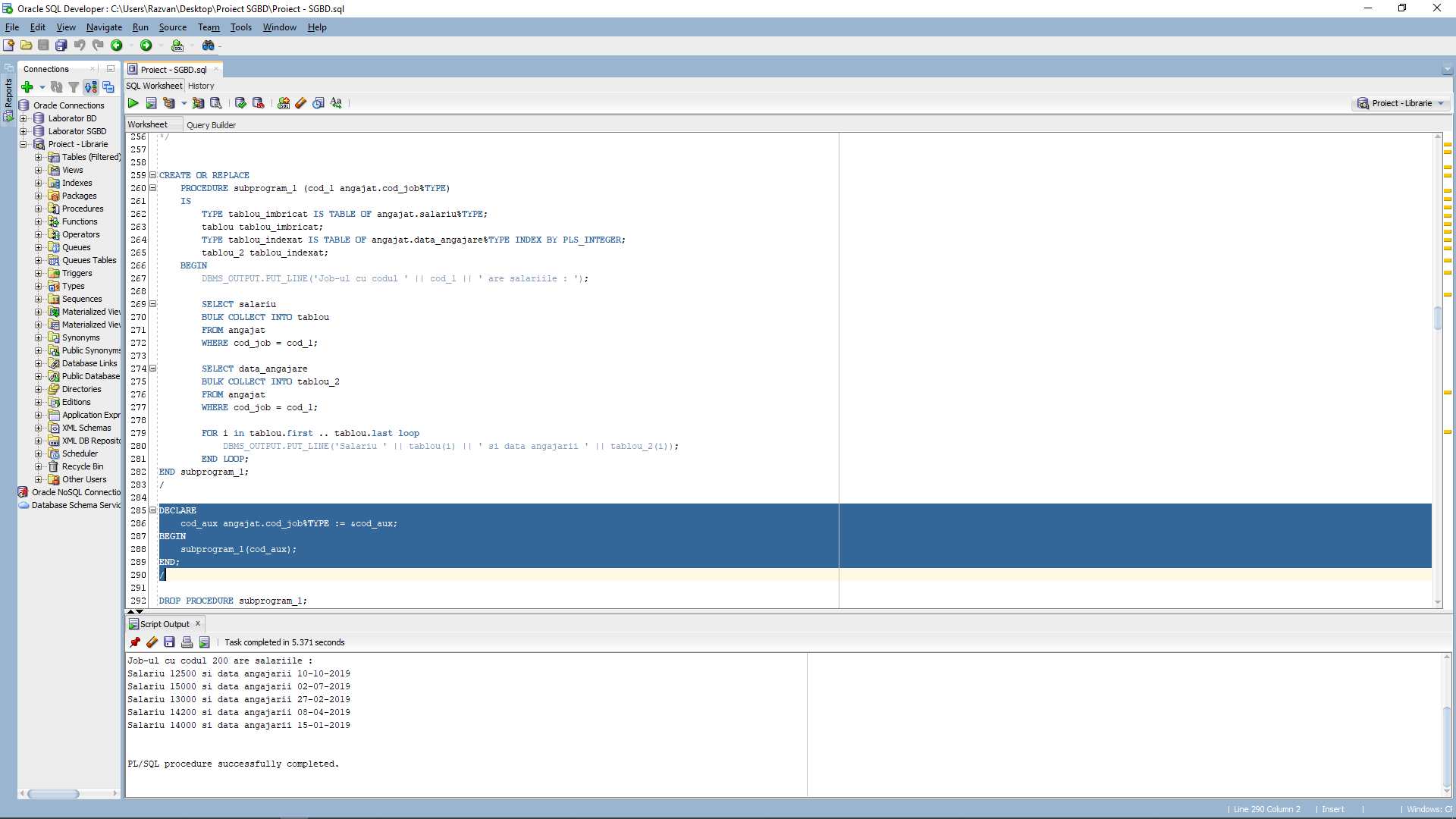
BEGIN

subprogram\_1(cod\_aux);

END;

/

DROP PROCEDURE subprogram\_1;



1. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

/\*

Afisati cate bonuri a eliberat fiecare librarie, valoarea tuturor bonurilor si valoarea medie a unui bon.

\*/

CREATE OR REPLACE

PROCEDURE subprogram\_2

IS

cod angajat.cod\_librarie%TYPE;

nr NUMBER(2);

val NUMBER(5);

CURSOR c IS

SELECT angajat.cod\_librarie, COUNT(bon.cod\_bon), SUM(bon.valoare)

FROM angajat, bon, librarie

WHERE angajat.cod\_angajat = bon.cod\_angajat AND angajat.cod\_librarie = librarie.cod\_librarie

GROUP BY angajat.cod\_librarie;

BEGIN

OPEN C;

LOOP

FETCH c INTO cod, nr, val;

EXIT WHEN c%NOTFOUND;

IF nr = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Libraria cu codul ' || cod || ' nu a eliberat bonuri.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Libraria cu codul ' || cod || ' a eliberat ' || nr || ' bonuri cu valoarea ' || val ||

' si valoarea medie a unui bon este ' || val / nr);

END IF;

END LOOP;

CLOSE c;

END subprogram\_2;

/

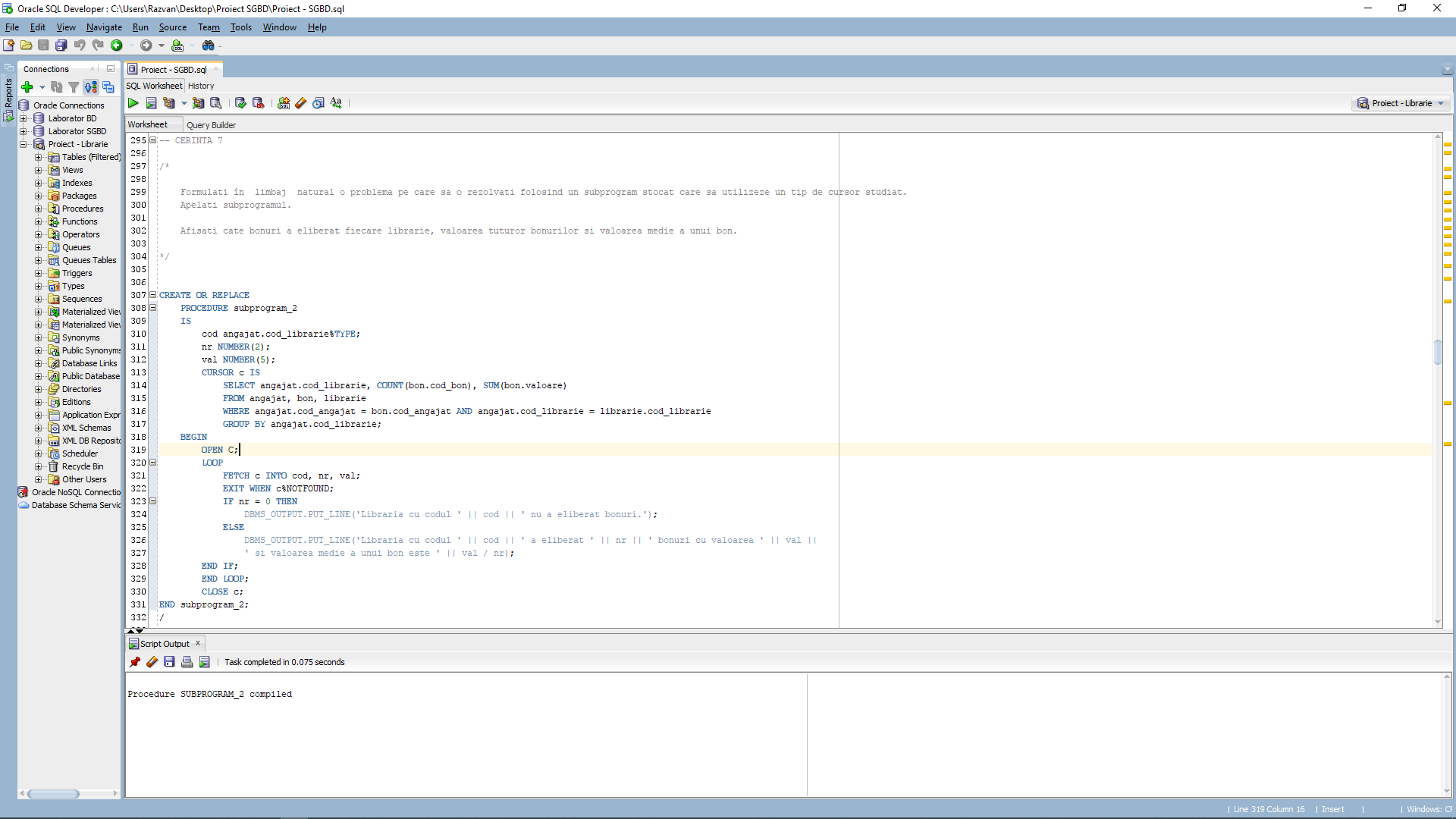
BEGIN

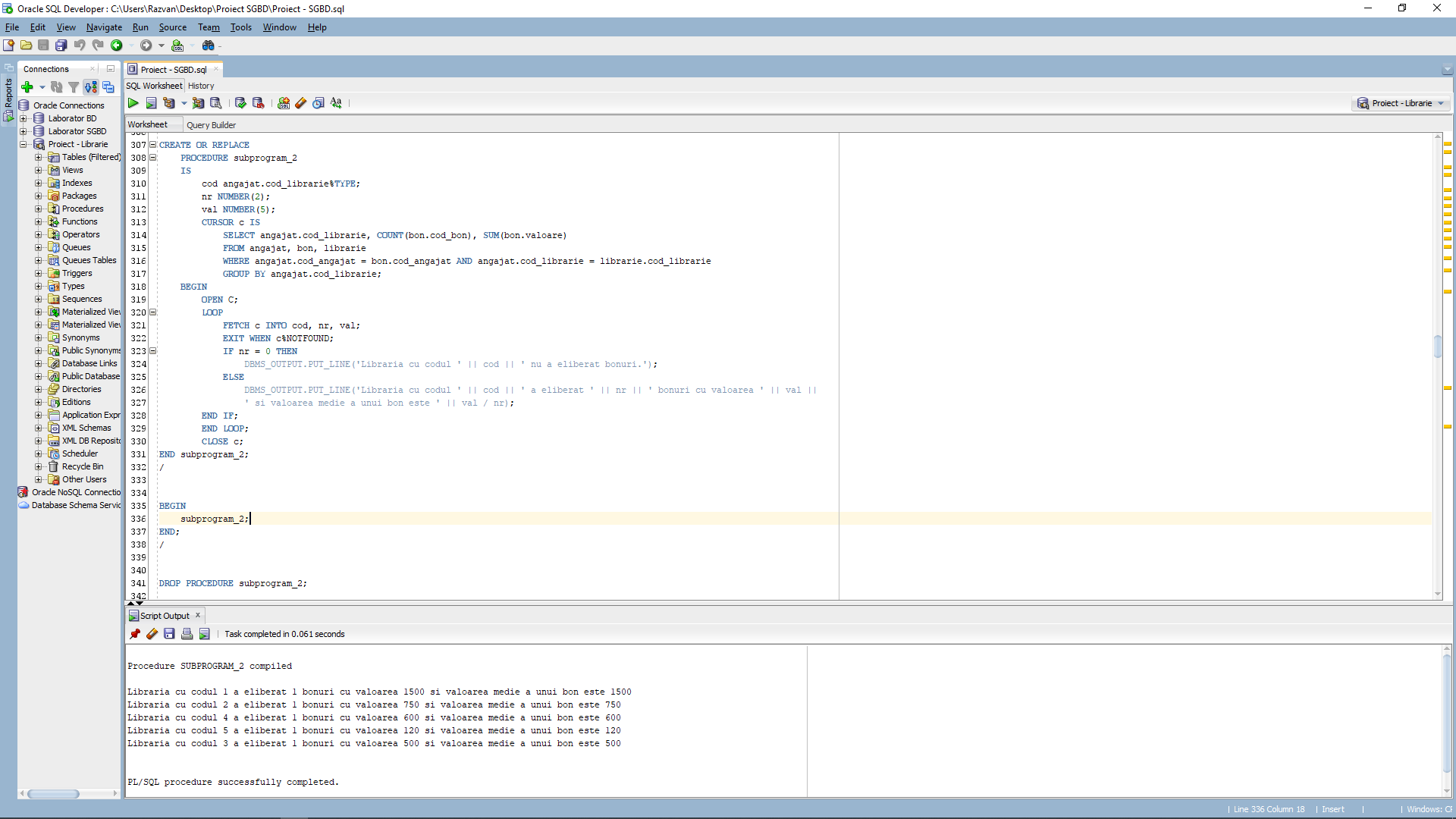
subprogram\_2;

END;

/

DROP PROCEDURE subprogram\_2;





1. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

/\*

In cazul in care clientul a primit o oferta, afisati valoarea bonului corespunzator aplicant reducerea.

In cazul in care clientul nu a primit o oferta afisati un mesaj corespunzator.

\*/

CREATE OR REPLACE

FUNCTION functie\_1(cod\_c client.cod\_client%TYPE)

RETURN NUMBER

IS

valoare\_1 bon.valoare%TYPE;

reducere oferta.dicount%TYPE;

BEGIN

SELECT bon.valoare

INTO valoare\_1

FROM bon, i\_se\_face, oferta

WHERE bon.cod\_client = cod\_c AND cod\_c = i\_se\_face.cod\_client AND i\_se\_face.cod\_oferta = oferta.cod\_oferta;

SELECT oferta.dicount

INTO reducere

FROM bon, i\_se\_face, oferta

WHERE bon.cod\_client = cod\_c AND cod\_c = i\_se\_face.cod\_client AND i\_se\_face.cod\_oferta = oferta.cod\_oferta;

RETURN valoare\_1 - ( valoare\_1 \* (reducere \* 0.01) );

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu codul ' || cod\_c || ' nu beneficiaza de reducere.');

END functie\_1;

/

DECLARE

cod client.cod\_client%TYPE;

CURSOR c\_2 IS

SELECT cod\_client

FROM client;

BEGIN

OPEN c\_2;

LOOP

FETCH c\_2 INTO cod;

EXIT WHEN c\_2%NOTFOUND;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu codul ' || cod || ' beneficiaza de reducere si plateste: ' || functie\_1(cod));

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT('');

END;

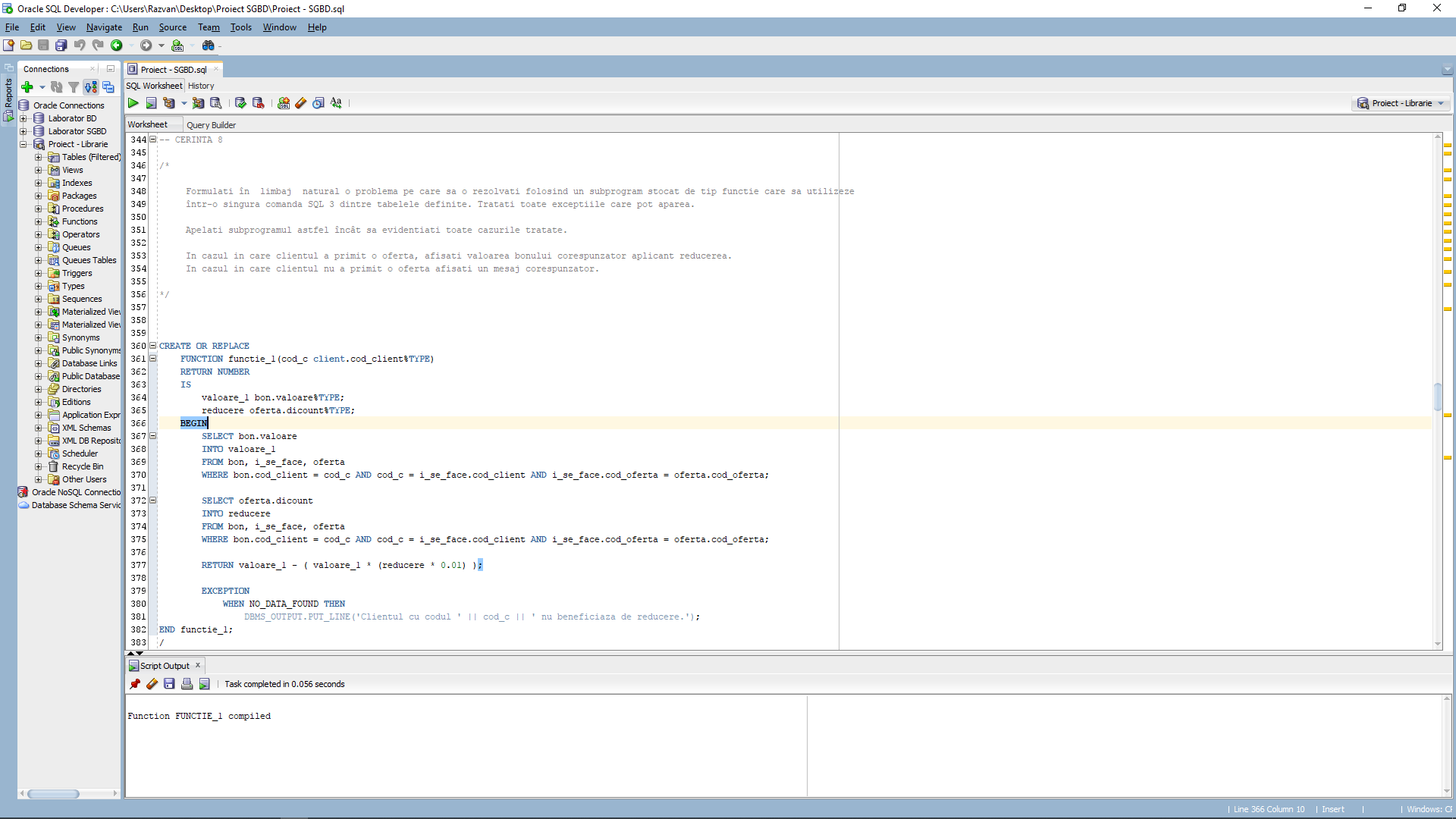
END LOOP;

CLOSE c\_2;

END;

/

DROP FUNCTION functie\_1;





1. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat 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 locatia librariei care are carti ce sunt livrate de catre un curier dat.

\*/

CREATE OR REPLACE

PROCEDURE subprogram\_3(cod\_c curier.cod\_curier%TYPE)

IS

adr locatie.adresa%TYPE;

BEGIN

SELECT locatie.adresa

INTO adr

FROM locatie, librarie, vinde, carte, curier

WHERE curier.cod\_curier = cod\_c AND curier.cod\_curier = carte.cod\_curier AND

carte.cod\_carte = vinde.cod\_carte AND vinde.cod\_librarie = librarie.cod\_librarie AND

librarie.cod\_librarie = locatie.cod\_librarie;

DBMS\_OUTPUT.PUT\_LINE('Adresa librariei la care livreaza curierul ' || cod\_c || ' este : ' || adr);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Curierul ' || cod\_c || ' nu livreaza la nicio librarie');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Curierul ' || cod\_c || ' livreaza la mai multe librarii');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Alta eroare!');

END subprogram\_3;

/

DECLARE

cod\_c curier.cod\_curier%TYPE;

CURSOR c\_3 IS

SELECT cod\_curier

FROM curier;

BEGIN

OPEN c\_3;

LOOP

FETCH c\_3 INTO cod\_c;

EXIT WHEN c\_3%NOTFOUND;

subprogram\_3(cod\_c);

END LOOP;

CLOSE c\_3;

END;

/

-- AM ADAUGAT ACESTE 2 LINII TEMPORAR IN TABEL DEOARECE AVEAM DOAR CAZURI CARE IMI INTRAU PE EXCEPTII.

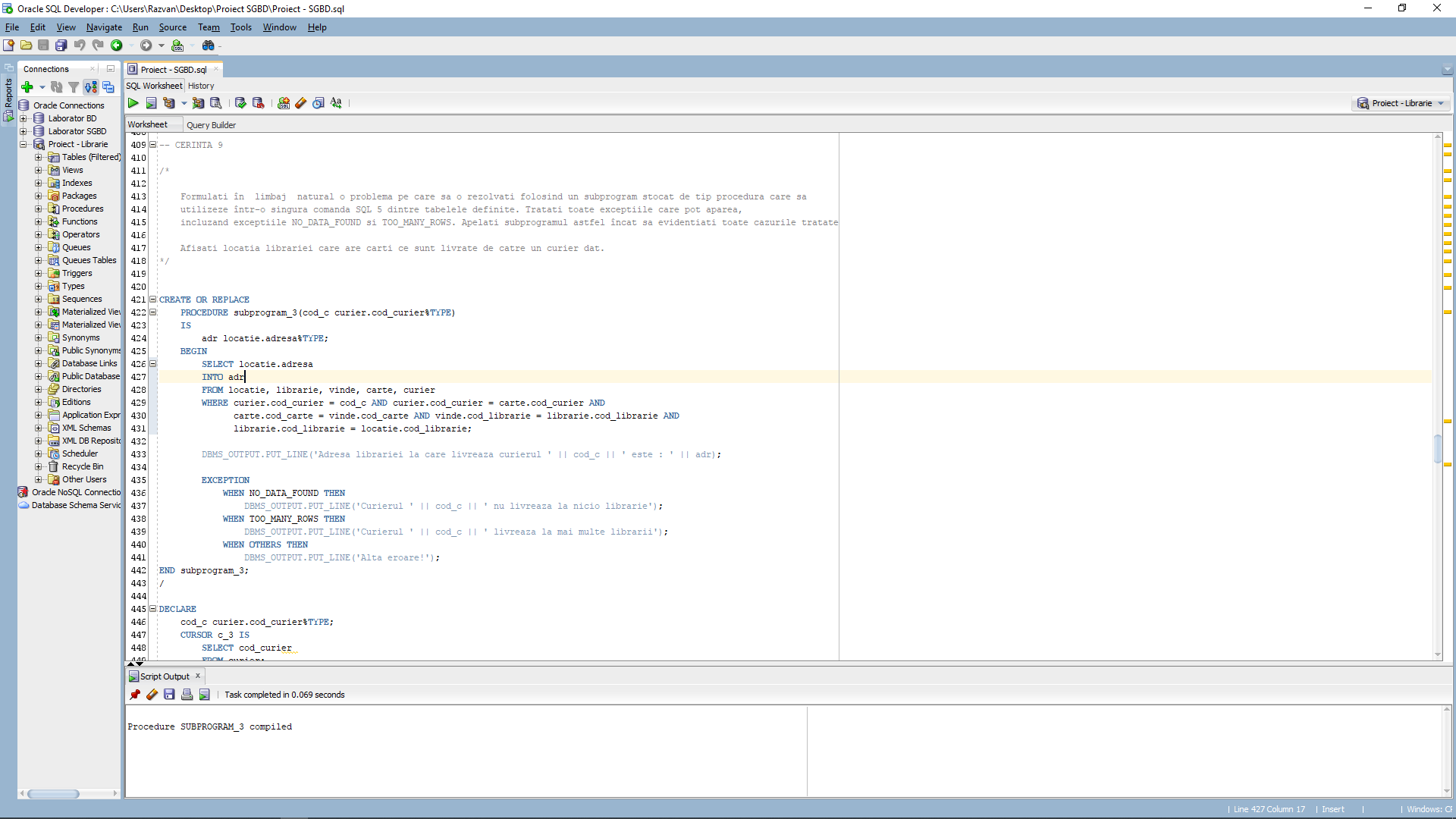
insert into CARTE values(105, 704, 304, 500, 'Sapiens');

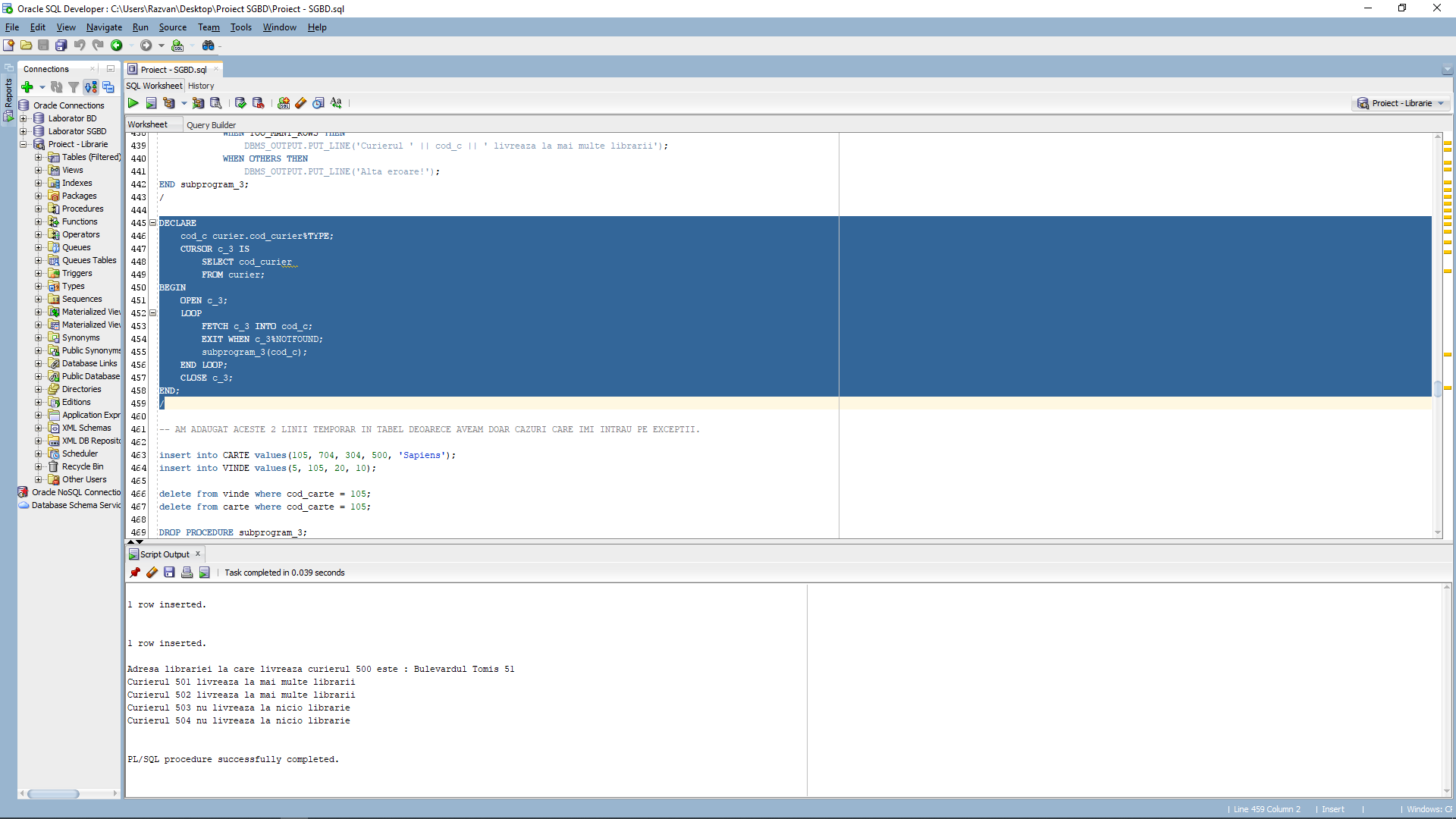
insert into VINDE values(5, 105, 20, 10);

delete from vinde where cod\_carte = 105;

delete from carte where cod\_carte = 105;

DROP PROCEDURE subprogram\_3;





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

/\*

Creati un trigger de tip LMD care permite modificarea inregistrarilor din tabela VINDE

doar de luni pana vineri intre orele 8 - 20.

\*/

CREATE OR REPLACE TRIGGER trigger\_1

BEFORE INSERT OR UPDATE OR DELETE ON vinde

BEGIN

IF (TO\_CHAR(SYSDATE,'D') = 1 OR TO\_CHAR(SYSDATE,'D') = 7) OR (TO\_CHAR(SYSDATE,'HH24') NOT BETWEEN 8 AND 20) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu se poate modifica stocul librariilor in afara programului de munca!');

END IF;

END;

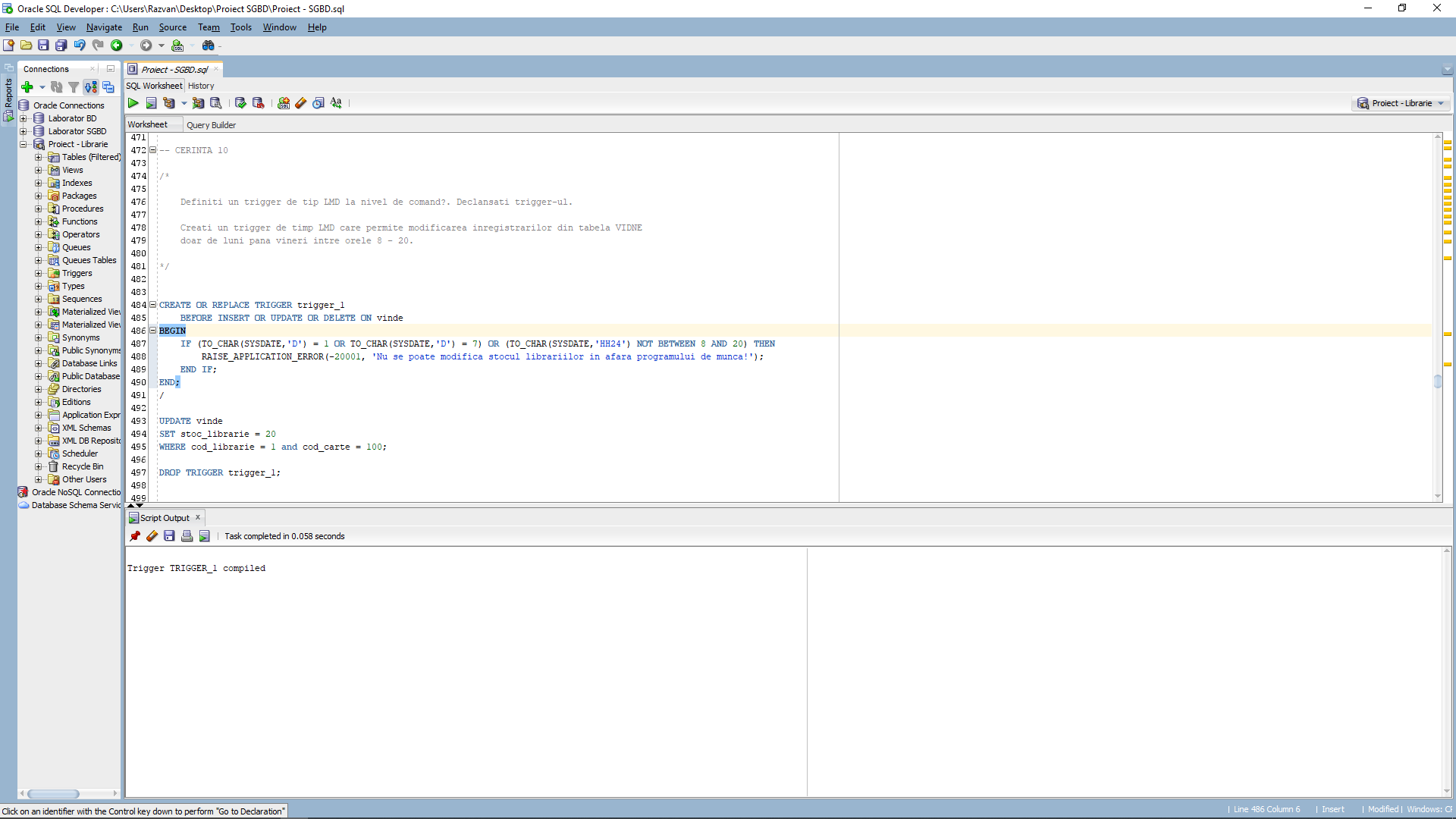
/

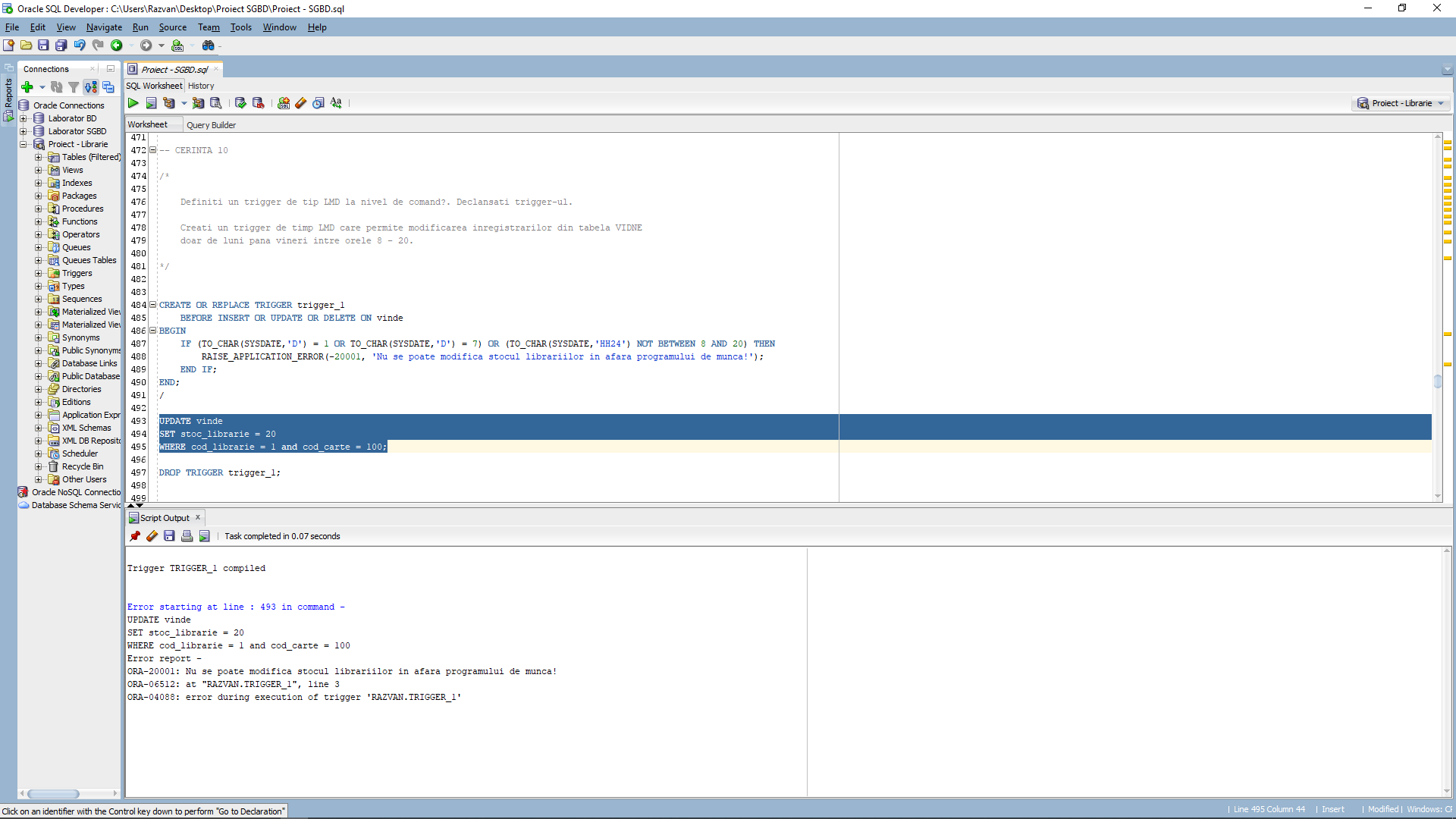
UPDATE vinde

SET stoc\_librarie = 20

WHERE cod\_librarie = 1 and cod\_carte = 100;

DROP TRIGGER trigger\_1;





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

/\*

Creati un trigger de tip LMD care nu permite modificarea vechimiii unui client la o data mai recenta decat cea precedenta.

\*/

CREATE OR REPLACE TRIGGER trigger\_2

BEFORE UPDATE OF vechime ON client

FOR EACH ROW

BEGIN

IF( SYSDATE - :NEW.vechime < SYSDATE - :OLD.vechime) THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Noua vechime a unui client nu poate fi mai mica decat vechimea precedenta!');

END IF;

END;

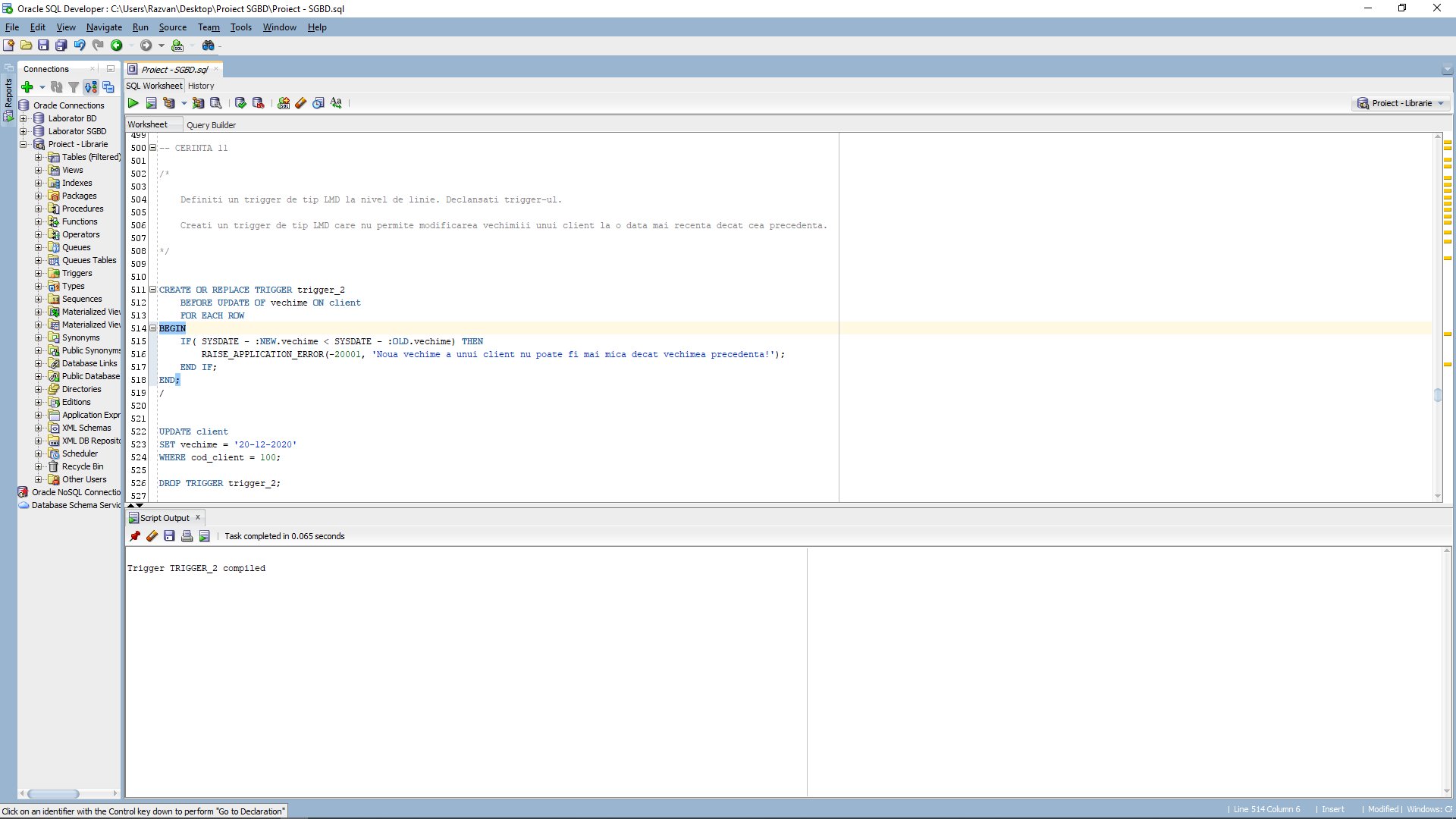
/

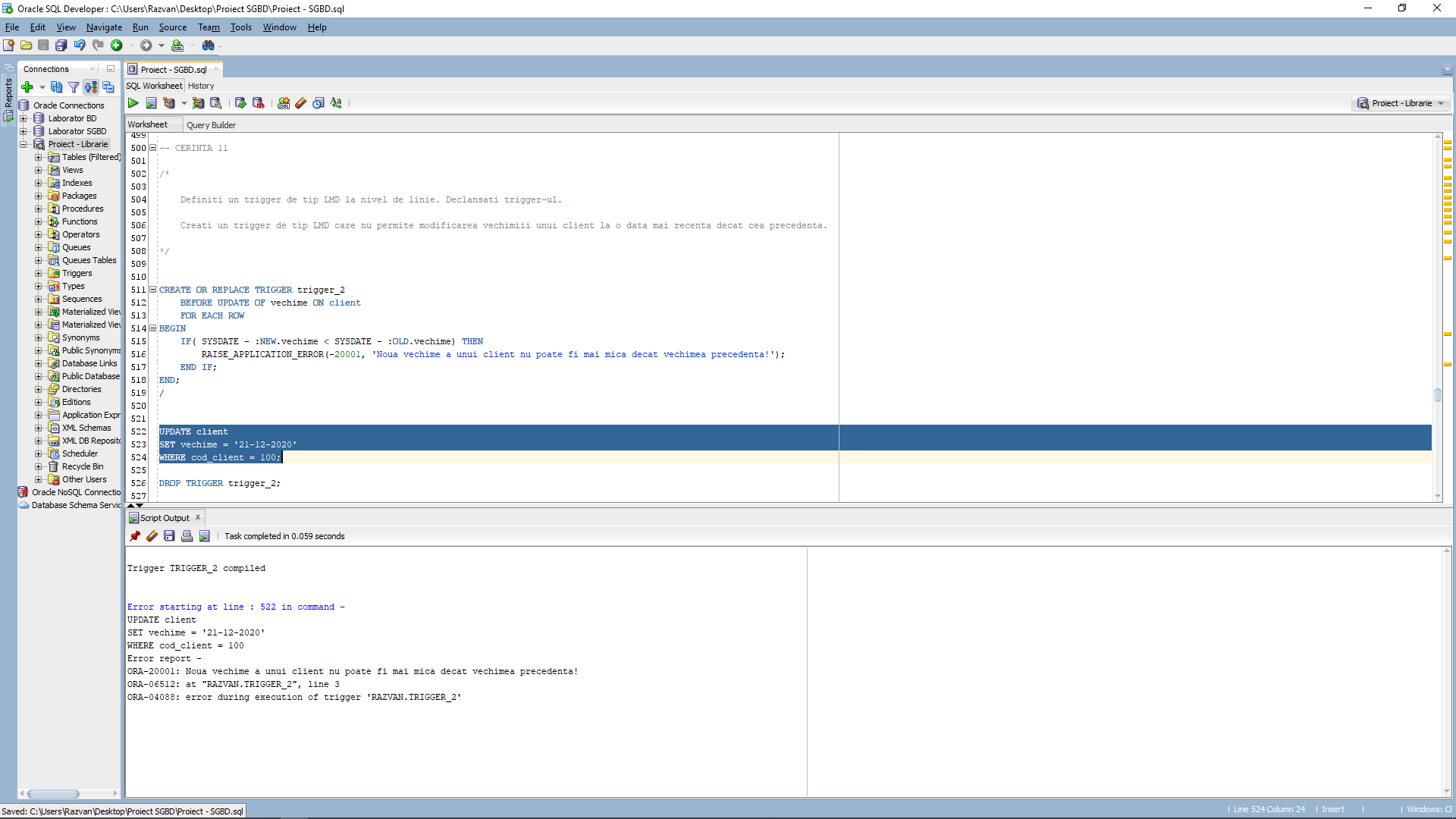
UPDATE client

SET vechime = '21-12-2020'

WHERE cod\_client = 100;

DROP TRIGGER trigger\_2;





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

/\*

Definiti un trigger de tip LDD. Declansati trigger-ul.

Creati un trigger care tine evidenta modificarilor de tip LDD facute asupra tabelei ANGAJAT ( User, Comanda, Data);

Stocati modificarile intr-un tabel auxiliar.

\*/

CREATE TABLE logs (nume\_user VARCHAR2(25),

modificare VARCHAR2(25),

tabela\_modificare VARCHAR2(25),

data\_modificarii DATE);

CREATE OR REPLACE TRIGGER trigger\_3

AFTER CREATE OR DROP OR ALTER ON DATABASE

BEGIN

INSERT INTO logs

VALUES(SYS.LOGIN\_USER, SYS.SYSEVENT, SYS.DICTIONARY\_OBJ\_NAME, SYSDATE);

END;

/

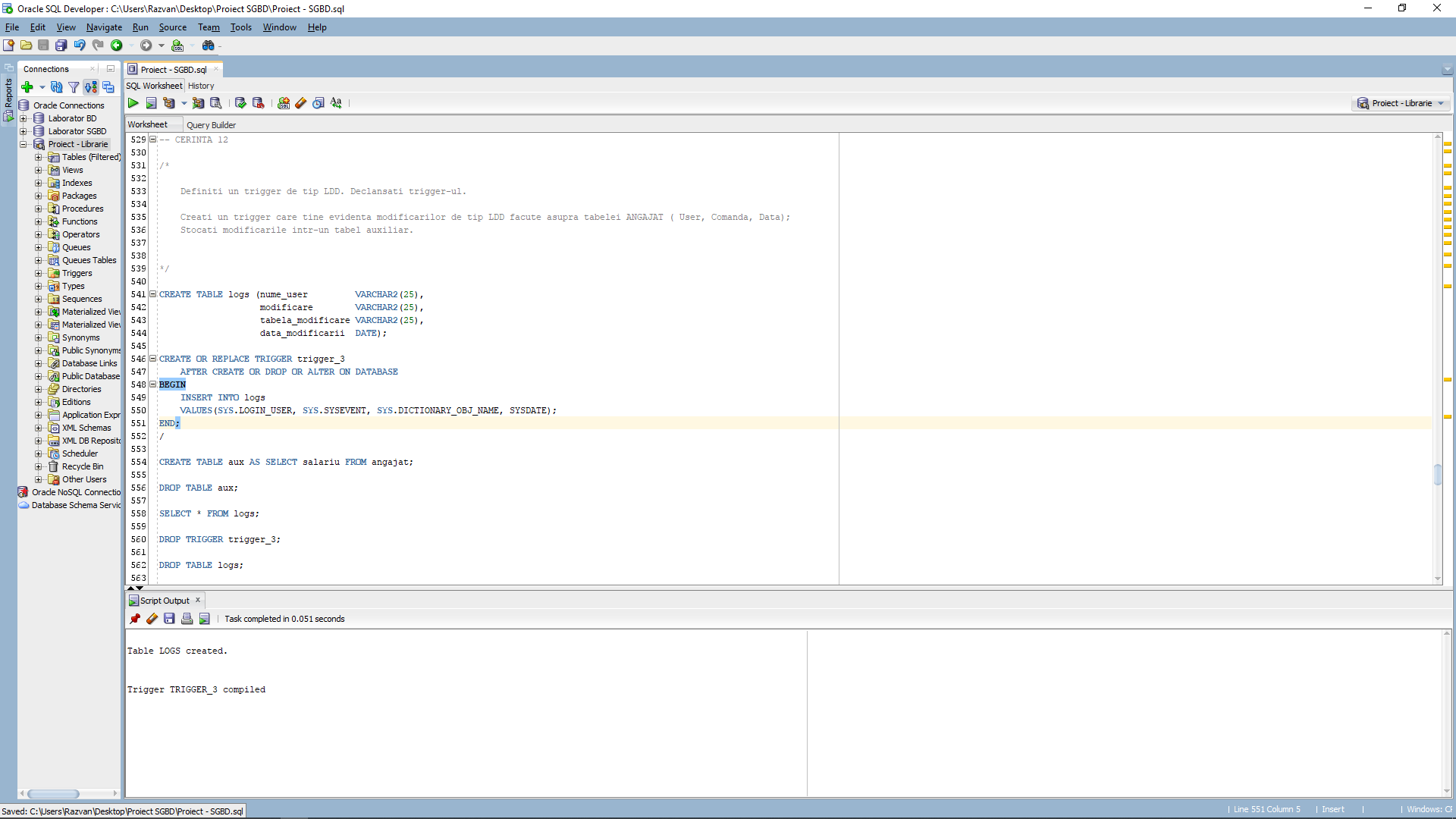
CREATE TABLE aux AS SELECT salariu FROM angajat;

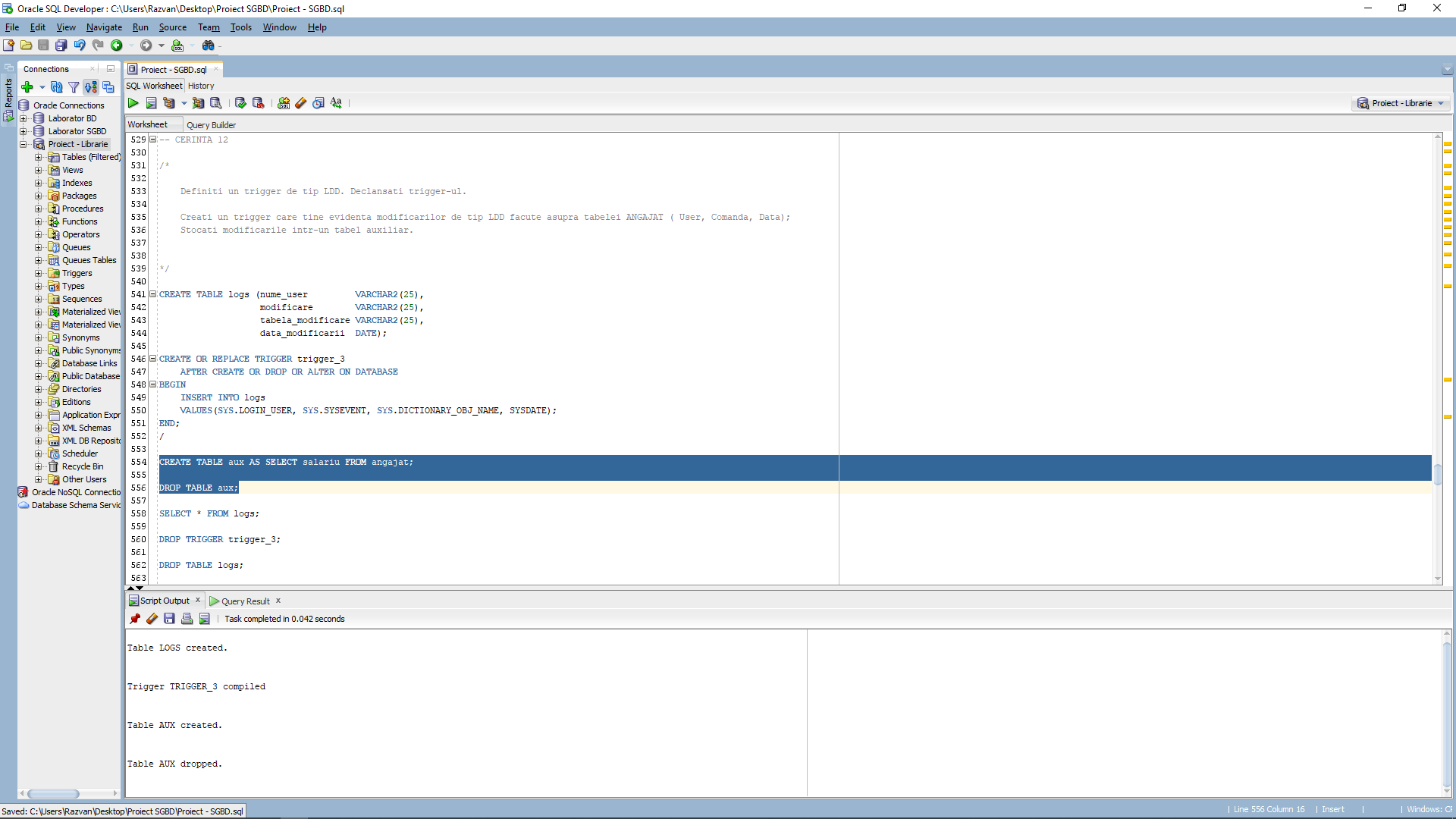
DROP TABLE aux;

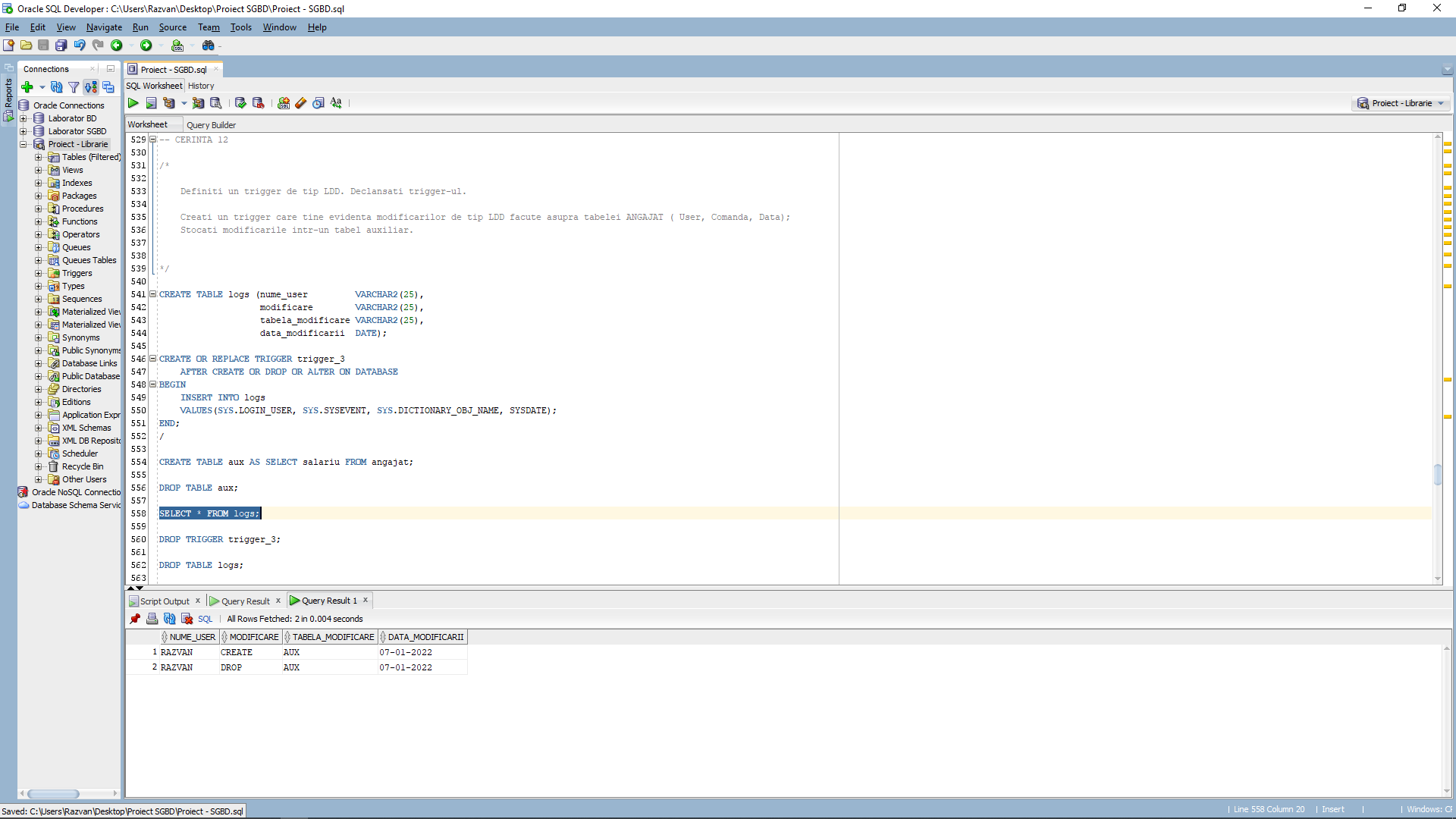
SELECT \* FROM logs;

DROP TRIGGER trigger\_3;

DROP TABLE logs;







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

CREATE OR REPLACE PACKAGE pachet AS

PROCEDURE subprogram\_1 (cod\_1 angajat.cod\_job%TYPE);

PROCEDURE subprogram\_2;

FUNCTION functie\_1(cod\_c client.cod\_client%TYPE)

RETURN NUMBER;

PROCEDURE subprogram\_3(cod\_c curier.cod\_curier%TYPE);

END pachet;

/

CREATE OR REPLACE PACKAGE BODY pachet AS

PROCEDURE subprogram\_1 (cod\_1 angajat.cod\_job%TYPE)

IS

TYPE tablou\_imbricat IS TABLE OF angajat.salariu%TYPE;

tablou tablou\_imbricat;

TYPE tablou\_indexat IS TABLE OF angajat.data\_angajare%TYPE INDEX BY PLS\_INTEGER;

tablou\_2 tablou\_indexat;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Job-ul cu codul ' || cod\_1 || ' are salariile : ');

SELECT salariu

BULK COLLECT INTO tablou

FROM angajat

WHERE cod\_job = cod\_1;

SELECT data\_angajare

BULK COLLECT INTO tablou\_2

FROM angajat

WHERE cod\_job = cod\_1;

FOR i in tablou.first .. tablou.last loop

DBMS\_OUTPUT.PUT\_LINE('Salariu ' || tablou(i) || ' si data angajarii ' || tablou\_2(i));

END LOOP;

END subprogram\_1;

PROCEDURE subprogram\_2

IS

cod angajat.cod\_librarie%TYPE;

nr NUMBER(2);

val NUMBER(5);

CURSOR c IS

SELECT angajat.cod\_librarie, COUNT(bon.cod\_bon), SUM(bon.valoare)

FROM angajat, bon, librarie

WHERE angajat.cod\_angajat = bon.cod\_angajat AND angajat.cod\_librarie = librarie.cod\_librarie

GROUP BY angajat.cod\_librarie;

BEGIN

OPEN C;

LOOP

FETCH c INTO cod, nr, val;

EXIT WHEN c%NOTFOUND;

IF nr = 0 THEN

DBMS\_OUTPUT.PUT\_LINE('Libraria cu codul ' || cod || ' nu a eliberat bonuri.');

ELSE

DBMS\_OUTPUT.PUT\_LINE('Libraria cu codul ' || cod || ' a eliberat ' || nr || ' bonuri cu valoarea ' || val ||

' si valoarea medie a unui bon este ' || val / nr);

END IF;

END LOOP;

CLOSE c;

END subprogram\_2;

FUNCTION functie\_1(cod\_c client.cod\_client%TYPE)

RETURN NUMBER

IS

valoare\_1 bon.valoare%TYPE;

reducere oferta.dicount%TYPE;

BEGIN

SELECT bon.valoare

INTO valoare\_1

FROM bon, i\_se\_face, oferta

WHERE bon.cod\_client = cod\_c AND cod\_c = i\_se\_face.cod\_client AND i\_se\_face.cod\_oferta = oferta.cod\_oferta;

SELECT oferta.dicount

INTO reducere

FROM bon, i\_se\_face, oferta

WHERE bon.cod\_client = cod\_c AND cod\_c = i\_se\_face.cod\_client AND i\_se\_face.cod\_oferta = oferta.cod\_oferta;

RETURN valoare\_1 - ( valoare\_1 \* (reducere \* 0.01) );

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Clientul cu codul ' || cod\_c || ' nu beneficiaza de reducere.');

END functie\_1;

PROCEDURE subprogram\_3(cod\_c curier.cod\_curier%TYPE)

IS

adr locatie.adresa%TYPE;

BEGIN

SELECT locatie.adresa

INTO adr

FROM locatie, librarie, vinde, carte, curier

WHERE curier.cod\_curier = cod\_c AND curier.cod\_curier = carte.cod\_curier AND

carte.cod\_carte = vinde.cod\_carte AND vinde.cod\_librarie = librarie.cod\_librarie AND

librarie.cod\_librarie = locatie.cod\_librarie;

DBMS\_OUTPUT.PUT\_LINE('Adresa librariei la care livreaza curierul ' || cod\_c || ' este : ' || adr);

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Curierul ' || cod\_c || ' nu livreaza la nicio librarie');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Curierul ' || cod\_c || ' livreaza la mai multe librarii');

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('Alta eroare!');

END;

END pachet;

/

EXECUTE pachet.subprogram\_1(200);

EXECUTE pachet.subprogram\_2;

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Valoare bon dupa discount: ' || pachet.functie\_1(100));

DBMS\_OUTPUT.PUT\_LINE('Valoare : ' || pachet.functie\_1(101));

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT('');

END;

/

EXECUTE pachet.subprogram\_3(501);

DROP PACKAGE pachet;

