PROIECT SGBD

Nume: Lucan Cristian Stefan

Seria: 25

Grupa 252

Anul Universitar: 2024-2025

Cuprins:

Cerinta 1: 2

Cerinta 2: 2

Cerinta 3: 3

Cerinta 4: 4

Cerinta 5: 9

Cerinta 6: 13

Cerinta 7: 16

Cerinta 8:19

Cerinta 9:22

Cerinta 10:26

Cerinta 11: 28

Cerinta 12: 29

Introducere:

Tema aleasa: Lant de spitale din Romania

Infrastructura aleasa:

Versiune: Oracle Database 21c Express Edition Release 21.0.0.0.0 - Production

RAM alocat- 16 GB

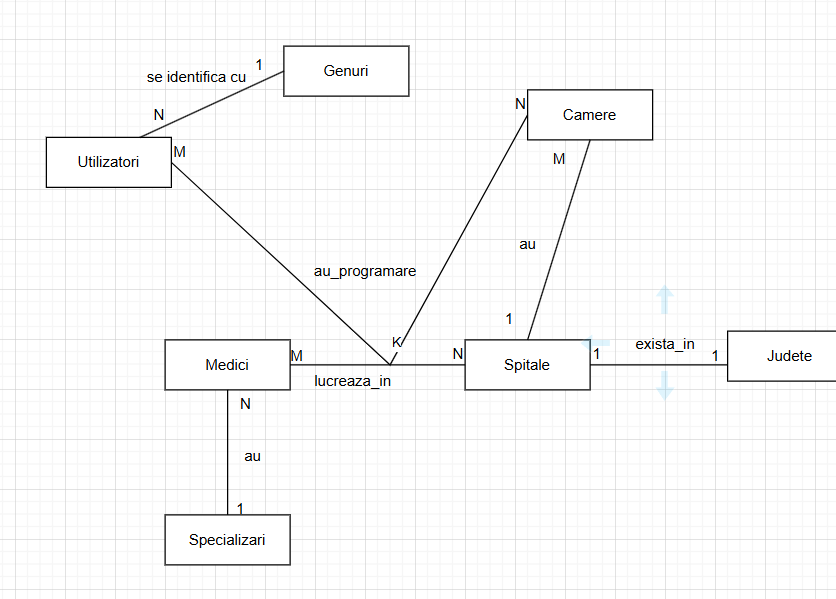
Sistem operare: Windows

Masina virtuala: Nu

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

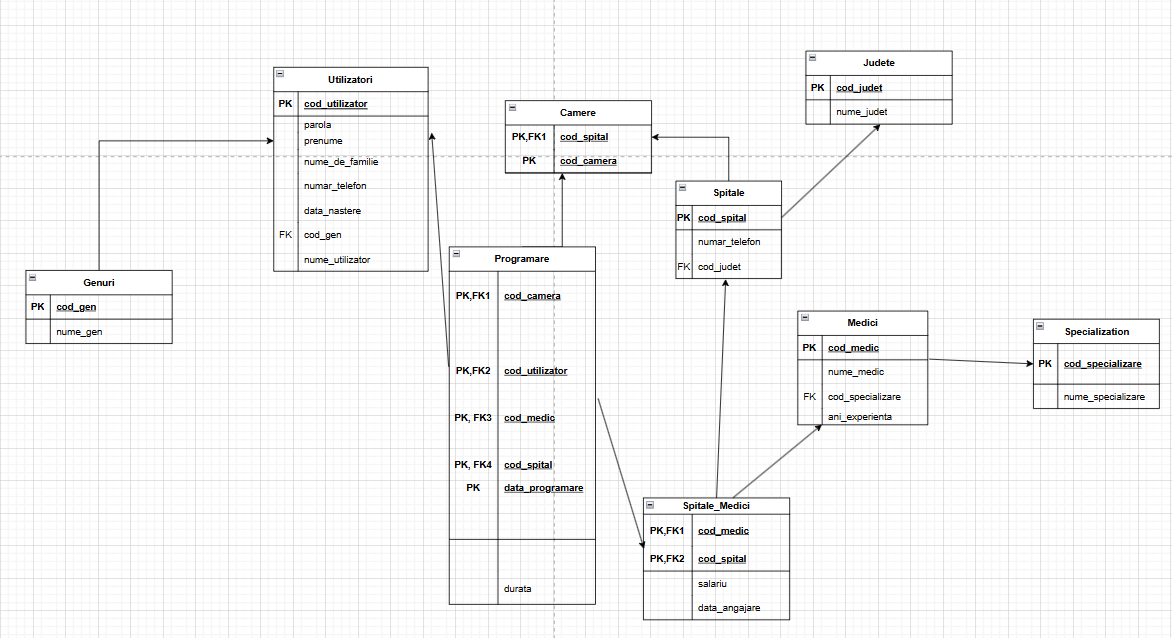
Baza de date reprezinta un lant de spitale ce exista cate unul in fiecare judet. Fiecare spital are angajati mai multi medici(si acesti medici pot lucra la mai multe spitale) avand o anumita specializare. Aceste spitale ofera utilizatorilor(pacientilor) posibiliatea de a-si face programari la oricare dintre medicii angajati, intr-una din camerele de care dispun. Fiecare utilizator se identifica cu un gen.

/\*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, adăugând toate

constrângerile de integritate necesare (chei primare, cheile externe etc).

CREATE TABLE genuri(

cod\_gen INTEGER,

nume\_gen VARCHAR(20) NOT NULL UNIQUE,

CONSTRAINT genuri\_pk PRIMARY KEY(cod\_gen)

);

CREATE TABLE utilizatori(

cod\_utilizator INTEGER,

data\_nastere DATE NOT NULL,

prenume VARCHAR(20) NOT NULL,

nume\_de\_famile VARCHAR(20) NOT NULL,

numar\_telefon CHAR(10) NOT NULL UNIQUE,

nume\_utilizator VARCHAR(20) NOT NULL UNIQUE,

parola varchar(100) NOT NULL,

cod\_gen INTEGER NOT NULL,

cod\_rol INTEGER NOT NULL,

CONSTRAINT utilizatori\_pk PRIMARY KEY(cod\_utilizator),

CONSTRAINT utilizatori\_rol\_fk FOREIGN KEY(cod\_rol) REFERENCES roluri(cod\_rol),

CONSTRAINT utilizatori\_gen\_fk FOREIGN KEY(cod\_gen) REFERENCES genuri(cod\_gen)

);

CREATE TABLE judete(

cod\_judet INTEGER ,

nume\_judet VARCHAR(20) NOT NULL UNIQUE,

CONSTRAINT judete\_pk PRIMARY KEY(cod\_judet)

);

CREATE TABLE spitale(

cod\_spital INTEGER ,

cod\_judet INTEGER NOT NULL UNIQUE,

numar\_telefon CHAR(10) NOT NULL UNIQUE,

CONSTRAINT spital\_pk PRIMARY KEY(cod\_spital),

CONSTRAINT spital\_judet\_fk FOREIGN KEY(cod\_judet) REFERENCES judete(cod\_judet)

);

CREATE TABLE camere(

cod\_camera INTEGER,

cod\_spital INTEGER,

CONSTRAINT spital\_camera\_pk PRIMARY KEY(cod\_camera, cod\_spital),

CONSTRAINT spital\_camera\_fk FOREIGN KEY(cod\_spital) REFERENCES spitale(cod\_spital)

);

CREATE TABLE specializari(

cod\_specializare INTEGER,

nume\_specializare VARCHAR(20) NOT NULL UNIQUE,

CONSTRAINT specializari\_pk PRIMARY KEY(cod\_specializare)

);

CREATE TABLE medici(

cod\_medic INTEGER ,

nume\_medic VARCHAR(20) NOT NULL UNIQUE,

cod\_specializare INTEGER NOT NULL,

ani\_experienta INTEGER NOT NULL,

CONSTRAINT medic\_pk PRIMARY KEY(cod\_medic),

CONSTRAINT medic\_spec\_fk FOREIGN KEY(cod\_specializare) REFERENCES specializari(cod\_specializare)

);

CREATE TABLE spitale\_medici(

cod\_spital INTEGER,

cod\_medic INTEGER,

data\_angajare DATE NOT NULL,

salariu NUMBER(6, 2) NOT NULL,

CONSTRAINT spitale\_medici\_pk PRIMARY KEY(cod\_spital, cod\_medic),

CONSTRAINT hosp\_med\_hosp\_fk FOREIGN KEY(cod\_spital) REFERENCES spitale(cod\_spital),

CONSTRAINT hosp\_med\_med\_fk FOREIGN KEY(cod\_medic) REFERENCES medici(cod\_medic)

);

CREATE TABLE programari(

cod\_utilizator INTEGER,

cod\_spital INTEGER,

cod\_medic INTEGER,

cod\_camera INTEGER,

data\_programare DATE,

durata INTEGER DEFAULT 30 NOT NULL, -- IN MINUTES

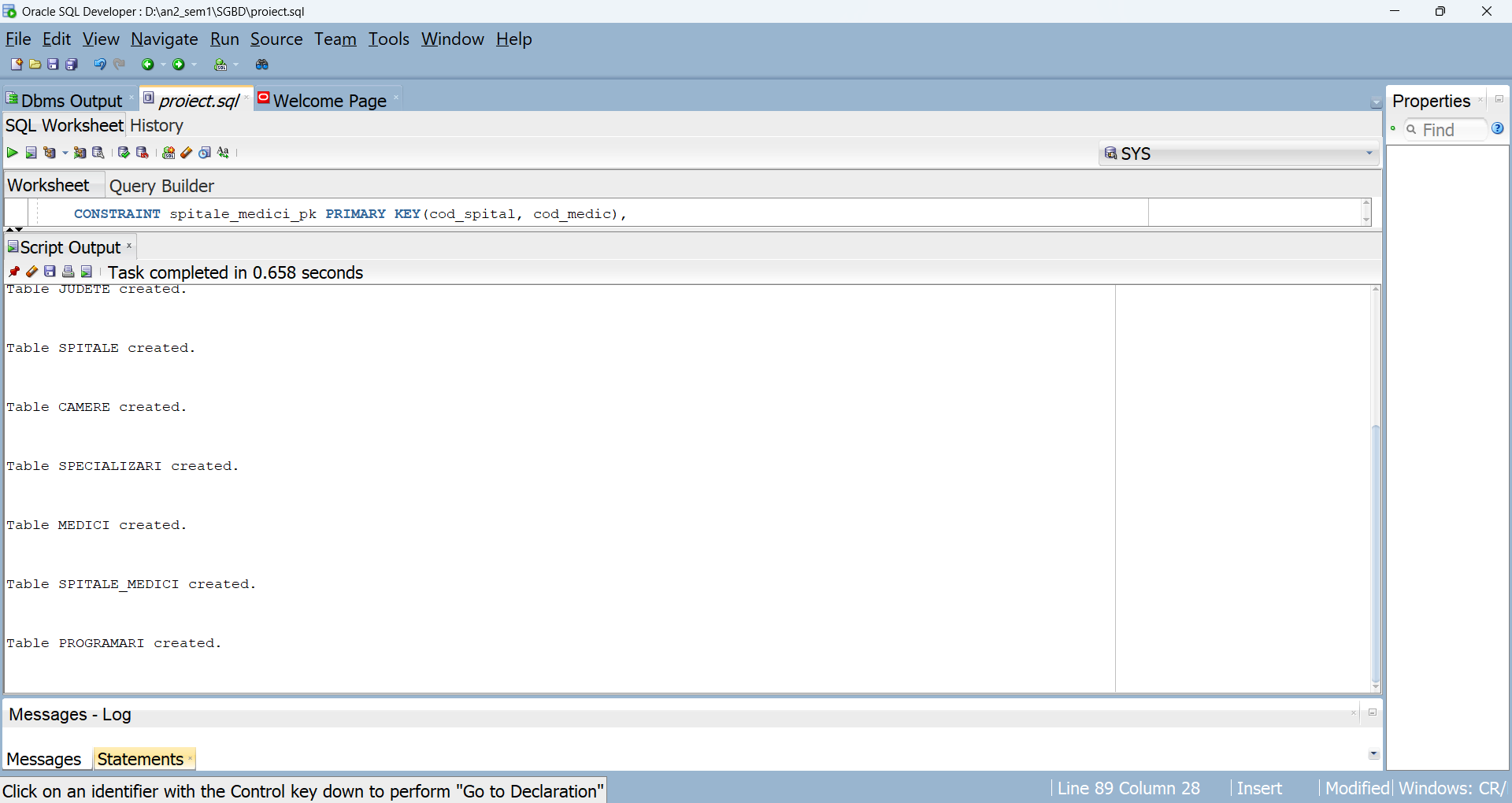
CONSTRAINT programari\_pk PRIMARY KEY(cod\_utilizator, cod\_spital, cod\_medic, cod\_camera, data\_programare),

CONSTRAINT programari\_camera\_fk FOREIGN KEY(cod\_camera, cod\_spital) REFERENCES camere(cod\_camera, cod\_spital),

CONSTRAINT programari\_user\_fk FOREIGN KEY(cod\_utilizator) REFERENCES utilizatori(cod\_utilizator),

CONSTRAINT programari\_hosp\_med\_fk FOREIGN KEY(cod\_spital, cod\_medic) REFERENCES spitale\_medici(cod\_spital, cod\_medic)

);



/\*5. Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate

independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).\*/

-- genuri data

INSERT INTO genuri VALUES(1, 'feminin');

INSERT INTO genuri VALUES(2, 'masculin');

INSERT INTO genuri VALUES(3, 'nebinar');

INSERT INTO genuri VALUES(4, 'omnigen');

INSERT INTO genuri VALUES(5, 'neurogen');

-- rol data

INSERT INTO roluri VALUES(1, 'admin');

INSERT INTO roluri VALUES(2, 'utilizator');

INSERT INTO roluri VALUES(3, 'medic');

INSERT INTO roluri VALUES(4, 'spital');

INSERT INTO roluri VALUES(5, 'asistent');

-- judet data

INSERT INTO judete VALUES(1, 'Bucuresti');

INSERT INTO judete VALUES(2, 'Cluj');

INSERT INTO judete VALUES(3, 'Iasi');

INSERT INTO judete VALUES(4, 'Constanta');

INSERT INTO judete VALUES(5, 'Timis');

-- spital data

INSERT INTO spitale VALUES(1, 1, '0218713860');

INSERT INTO spitale VALUES(2, 2, '0362952363');

INSERT INTO spitale VALUES(3, 3, '0786537062');

INSERT INTO spitale VALUES(4, 4, '0255249892');

INSERT INTO spitale VALUES(5, 5, '0351066109');

-- camere data

INSERT INTO camere VALUES(1, 1);

INSERT INTO camere VALUES(2, 1);

INSERT INTO camere VALUES(1, 2);

INSERT INTO camere VALUES(2, 2);

INSERT INTO camere VALUES(1, 3);

INSERT INTO camere VALUES(2, 3);

INSERT INTO camere VALUES(1, 4);

INSERT INTO camere VALUES(2, 4);

INSERT INTO camere VALUES(1, 5);

INSERT INTO camere VALUES(2, 5);

-- Specialization data

INSERT INTO specializari VALUES(1, 'Cardiologie');

INSERT INTO specializari VALUES(2, 'Dermatologie');

INSERT INTO specializari VALUES(3, 'Endocrinologie');

INSERT INTO specializari VALUES(4, 'Neurologie');

INSERT INTO specializari VALUES(5, 'Oftalmologie');

-- medici data

INSERT INTO medici VALUES(1, 'Teodor Paraschiv', 1, 1);

INSERT INTO medici VALUES(2, 'Antoaneta Oprea', 2, 5);

INSERT INTO medici VALUES(3, 'Doru Fodor', 3, 14);

INSERT INTO medici VALUES(4, 'Ina Mihalcea', 4, 15);

INSERT INTO medici VALUES(5, 'Sidonia Banica', 5, 15);

-- spitale\_medici data

-- spital medic data

INSERT INTO spitale\_medici VALUES(1, 1, SYSDATE, 4000);

INSERT INTO spitale\_medici VALUES(2, 2, SYSDATE, 5200);

INSERT INTO spitale\_medici VALUES(3, 3, SYSDATE, 3000);

INSERT INTO spitale\_medici VALUES(4, 4, SYSDATE, 1300);

INSERT INTO spitale\_medici VALUES(5, 5, SYSDATE, 1400);

INSERT INTO spitale\_medici VALUES(1, 2, SYSDATE, 2000);

INSERT INTO spitale\_medici VALUES(2, 3, SYSDATE, 4000);

INSERT INTO spitale\_medici VALUES(3, 4, SYSDATE, 2500);

INSERT INTO spitale\_medici VALUES(4, 5, SYSDATE, 3230);

INSERT INTO spitale\_medici VALUES(5, 1, SYSDATE, 6000);

INSERT INTO utilizatori VALUES(1, '12-Oct-1990', 'Alex', 'Marian', '1111111111', 'al\_m1', 'pass', 2, 2);

INSERT INTO utilizatori VALUES(2, '22-Aug-2000', 'Mihai', 'Joe', '2222222222', 'mh\_j1', 'pass1', 2, 3);

INSERT INTO utilizatori VALUES(3, '17-Nov-2000', 'Gabriela', 'Balan', '3333333333', 'gb\_b2', 'pass2', 1, 4);

INSERT INTO utilizatori VALUES(4, '5-Dec-2002', 'Kyle', 'Brock', '4444444444', 'kl\_b0', 'pass3', 4, 1);

INSERT INTO utilizatori VALUES(5, '23-May-2004', 'Alex', 'Nistor', '5555555555', 'al\_n1', 'pass4', 3, 5);

INSERT INTO programari VALUES(1, 1, 1, 1, SYSDATE, 30);

INSERT INTO programari VALUES(2, 2, 2, 1, SYSDATE, 30);

INSERT INTO programari VALUES(3, 3, 3, 1, SYSDATE, 30);

INSERT INTO programari VALUES(4, 4, 4, 1, SYSDATE, 30);

INSERT INTO programari VALUES(5, 5, 5, 1, SYSDATE, 30);

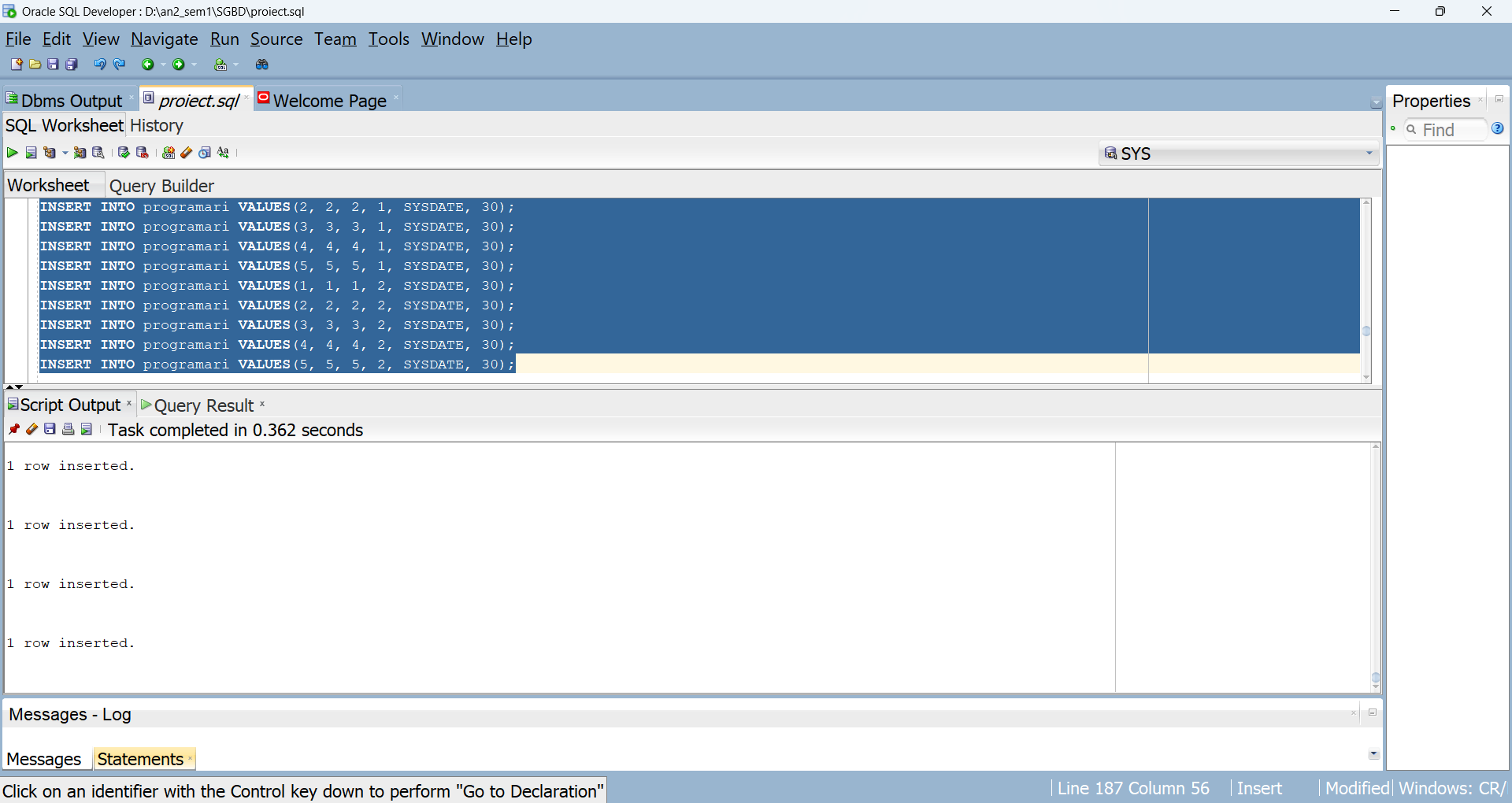
INSERT INTO programari VALUES(1, 1, 1, 2, SYSDATE, 30);

INSERT INTO programari VALUES(2, 2, 2, 2, SYSDATE, 30);

INSERT INTO programari VALUES(3, 3, 3, 2, SYSDATE, 30);

INSERT INTO programari VALUES(4, 4, 4, 2, SYSDATE, 30);

INSERT INTO programari VALUES(5, 5, 5, 2, SYSDATE, 30);



--/\*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.\*/

/\* Afisarea medicilor dintr-un spital(a salarilor si specializarilor lor\*/

CREATE OR REPLACE PROCEDURE raport\_medici\_din\_spital(p\_cod\_spital IN spitale.cod\_spital%TYPE) IS

TYPE medic\_array\_t IS VARRAY(10) OF medici.cod\_medic%TYPE;

lista\_medici medic\_array\_t := medic\_array\_t();

TYPE tabela\_salarii\_t IS TABLE OF spitale\_medici.salariu%TYPE;

lista\_salarii tabela\_salarii\_t := tabela\_salarii\_t();

TYPE spec\_array\_t IS TABLE OF specializari.nume\_specializare%TYPE INDEX BY PLS\_INTEGER;

lista\_spec spec\_array\_t;

CURSOR m\_cursor(s\_cod spitale.cod\_spital%TYPE) IS

SELECT m.cod\_medic, sm.salariu, s.nume\_specializare

FROM medici m

JOIN spitale\_medici sm ON m.cod\_medic = sm.cod\_medic

JOIN specializari s ON m.cod\_specializare = s.cod\_specializare

WHERE sm.cod\_spital = s\_cod;

m\_info m\_cursor%ROWTYPE;

BEGIN

OPEN m\_cursor(p\_cod\_spital);

LOOP

FETCH m\_cursor INTO m\_info;

EXIT WHEN m\_cursor%NOTFOUND;

lista\_medici.EXTEND;

lista\_medici(lista\_medici.COUNT) := m\_info.cod\_medic;

lista\_salarii.EXTEND;

lista\_salarii(lista\_salarii.COUNT) := m\_info.salariu;

lista\_spec(lista\_medici.COUNT) := m\_info.nume\_specializare;

END LOOP;

DBMS\_OUTPUT.PUT\_LINE('Medicii din spitalul cu ID ' || p\_cod\_spital || ':');

FOR i IN 1 .. lista\_medici.COUNT LOOP

DBMS\_OUTPUT.PUT\_LINE('Cod Medic: ' || lista\_medici(i) || ', Salariu: ' || lista\_salarii(i) || ', Specializare: ' || lista\_spec(i));

END LOOP;

CLOSE m\_cursor;

END raport\_medici\_din\_spital;

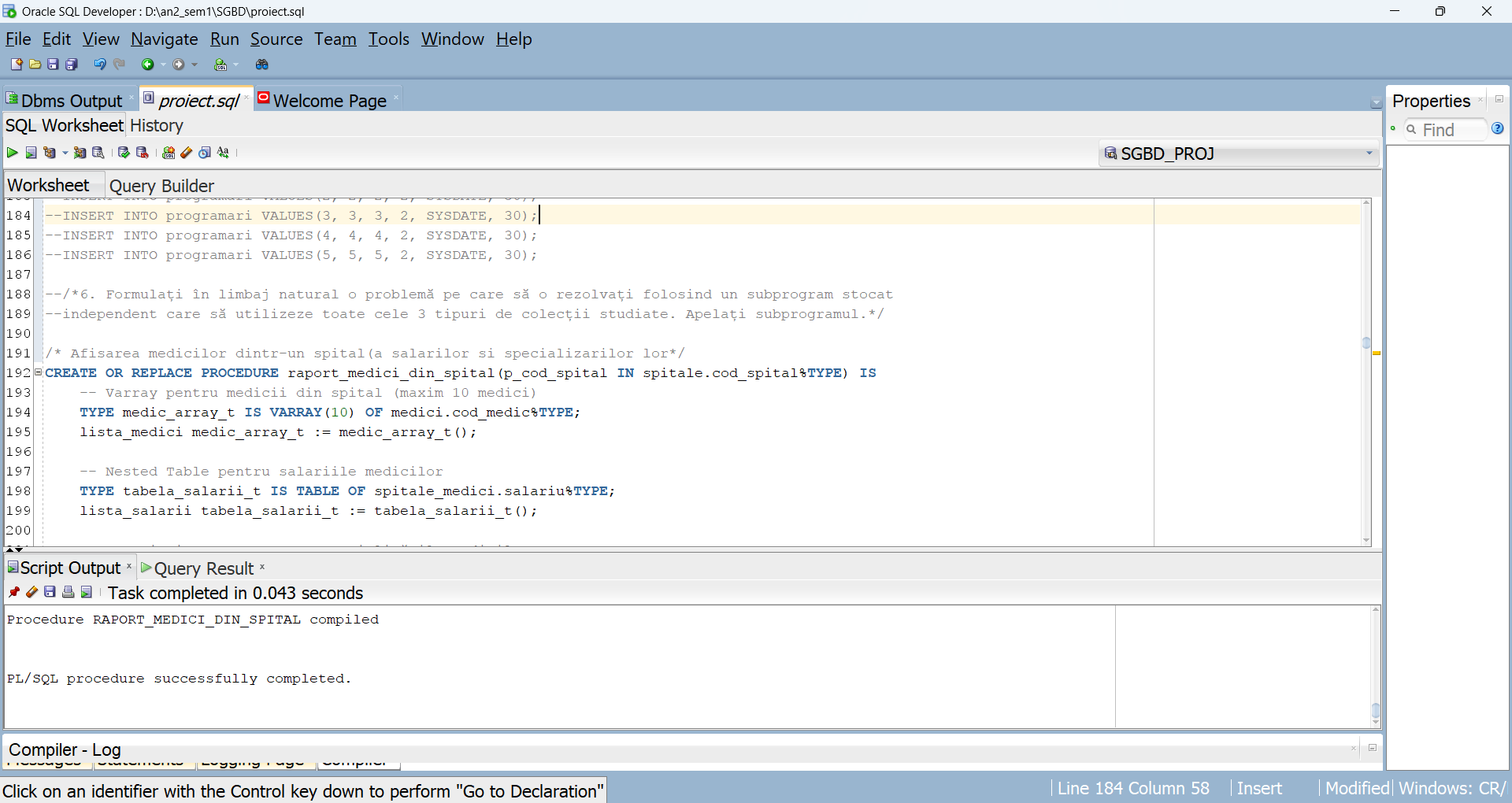
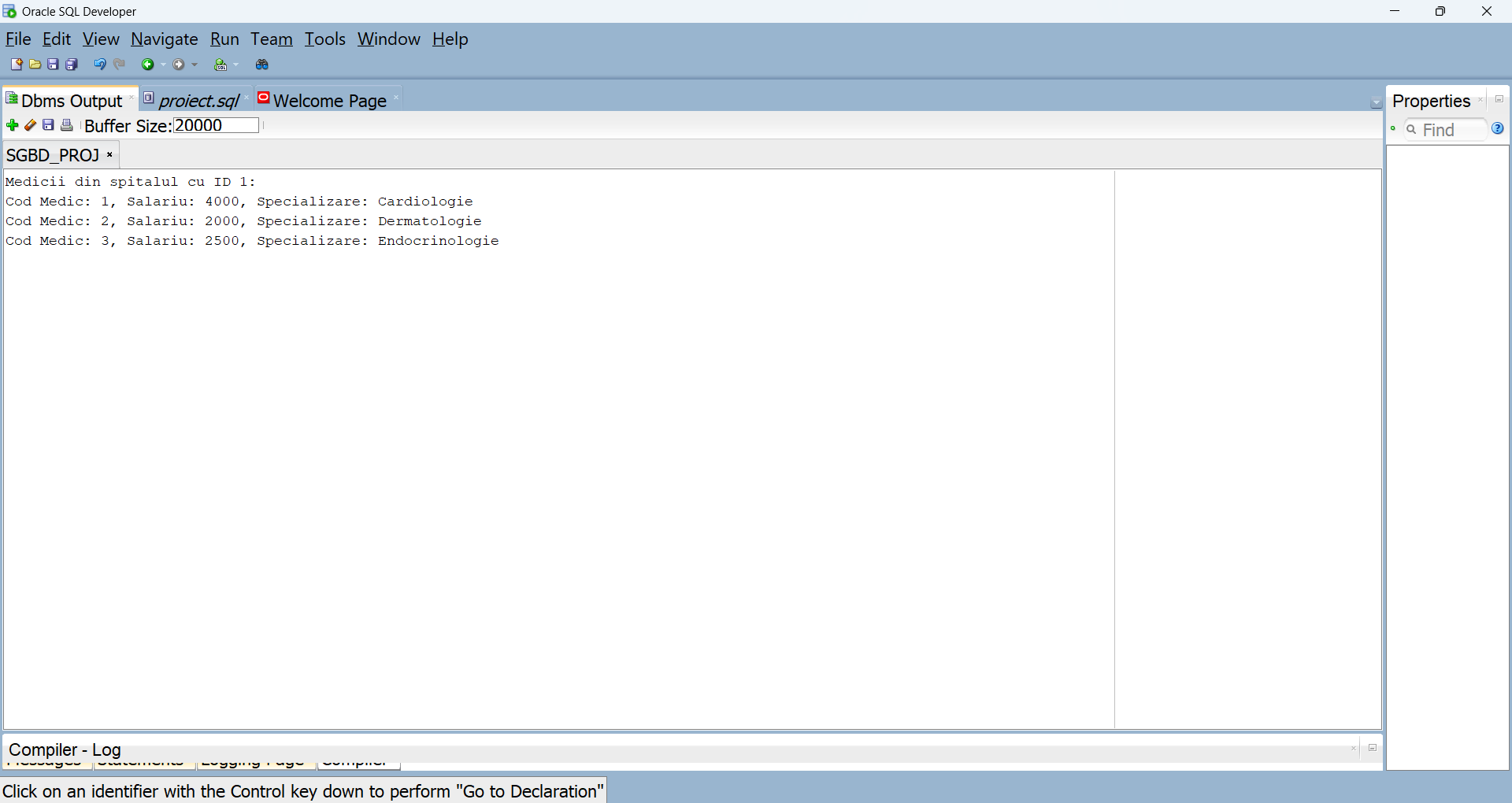
/

BEGIN

raport\_medici\_din\_spital(1);

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\*/

CREATE PROCEDURE afis\_medici\_spitale IS

CURSOR c\_spitale IS SELECT cod\_spital FROM spitale;

CURSOR c\_medici(p\_cod\_spital spitale.cod\_spital%TYPE) IS SELECT m.cod\_medic as cod, m.nume\_medic as nume, m.cod\_specializare as spec,

m.ani\_experienta as ani\_exp

FROM medici m JOIN spitale\_medici sm ON sm.cod\_medic = m.cod\_medic

WHERE sm.cod\_spital = p\_cod\_spital;

spital\_info c\_spitale%ROWTYPE;

medic\_info c\_medici%ROWTYPE;

BEGIN

-- Loop through the department cursor

OPEN c\_spitale;

LOOP

FETCH c\_spitale INTO spital\_info;

EXIT WHEN c\_spitale%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Cod spital: ' || spital\_info.cod\_spital);

OPEN c\_medici(spital\_info.cod\_spital);

LOOP

FETCH c\_medici INTO medic\_info;

EXIT WHEN c\_medici%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE('Medic: ' || medic\_info.cod || ' - ' || medic\_info.nume || ' ' || medic\_info.spec || ' ' || medic\_info.ani\_exp);

END LOOP;

CLOSE c\_medici;

END LOOP;

CLOSE c\_spitale;

END afis\_medici\_spitale;

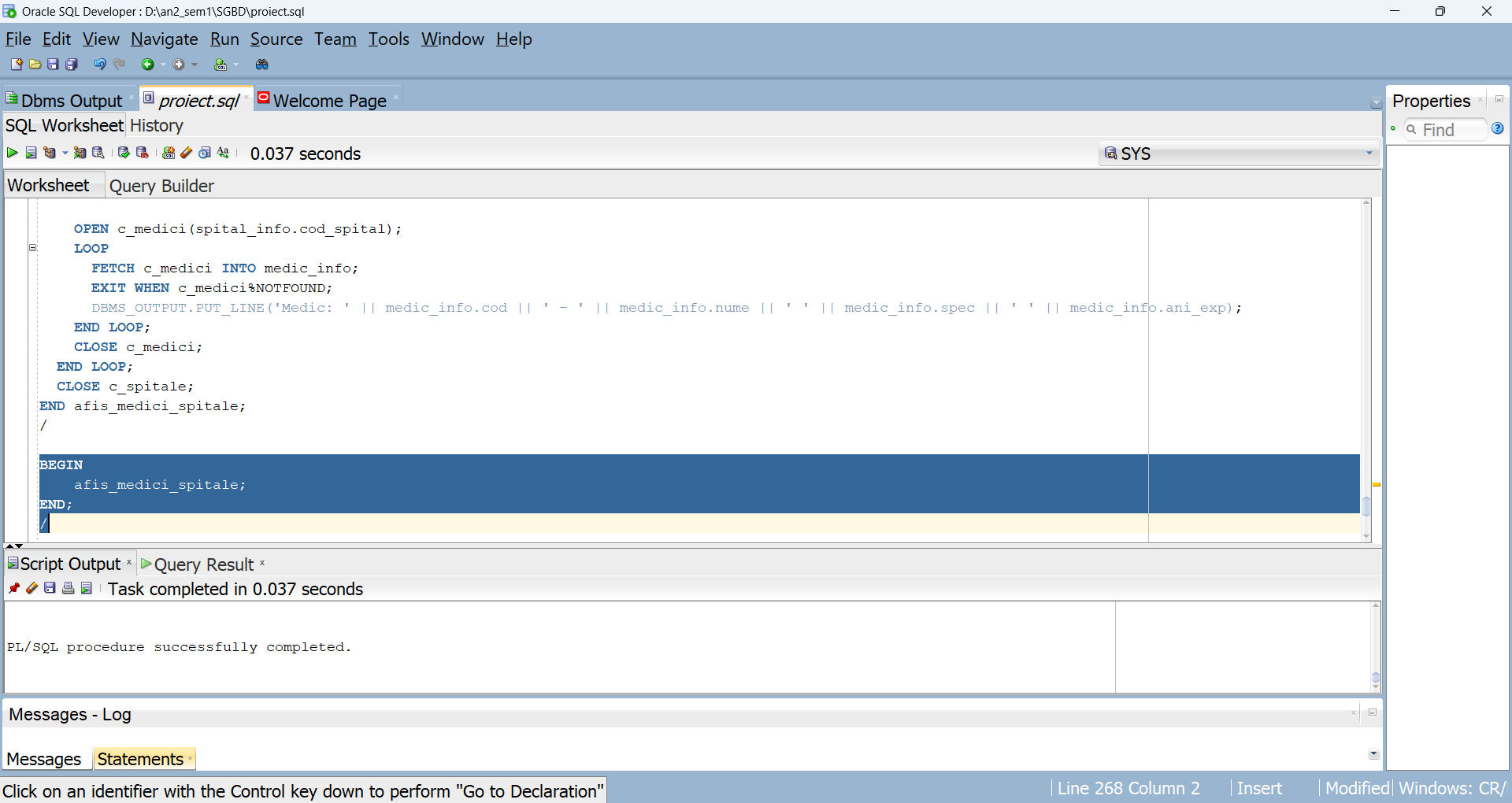
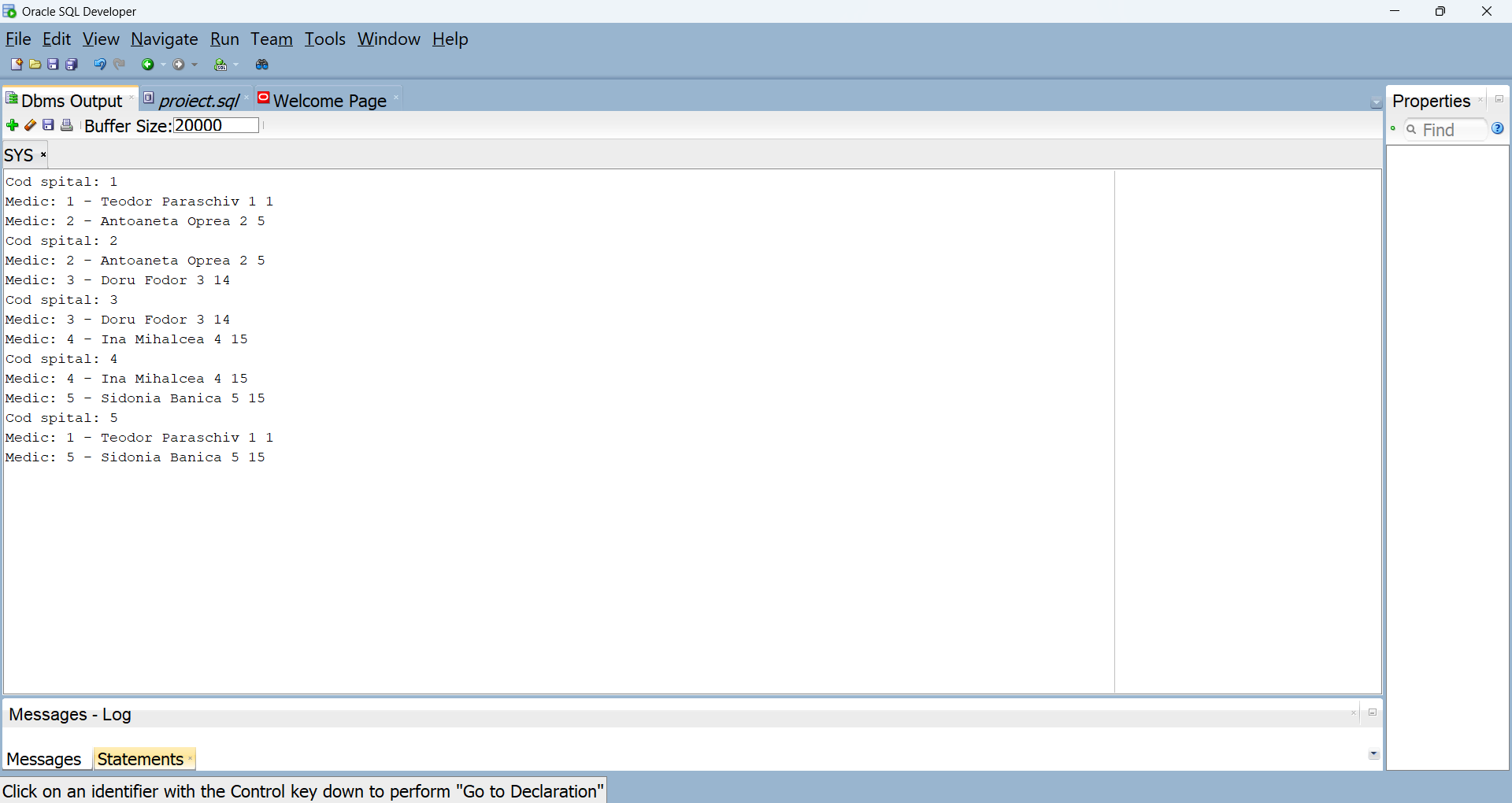
/

BEGIN

afis\_medici\_spitale;

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 create.

Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și

TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.\*/

/\*Afisarea numarului de telefon al spitalului, numelui medicului si a salariului pe care il castiga, cunoscand codul de specializare

si codul spitalului\*/

--Pentru a putea testa si too\_many\_rows

INSERT INTO medici VALUES (6, 'Marius Andrei', 1, 4);

INSERT INTO spitale\_medici VALUES(1, 6, SYSDATE, 2700);

CREATE OR REPLACE FUNCTION medic\_interesant(spec\_id specializari.cod\_specializare%TYPE, s\_id spitale.cod\_spital%TYPE) RETURN VARCHAR2 IS

TYPE medic\_info IS RECORD(

nr\_tel\_spital spitale.numar\_telefon%TYPE,

nume medici.nume\_medic%TYPE,

salariu spitale\_medici.salariu%TYPE);

m\_info medic\_info;

BEGIN

SELECT s.numar\_telefon, m.nume\_medic, sm.salariu

INTO m\_info.nr\_tel\_spital, m\_info.nume, m\_info.salariu

FROM medici m

JOIN spitale\_medici sm ON sm.cod\_medic = m.cod\_medic

JOIN spitale s ON s.cod\_spital = sm.cod\_spital

WHERE m.cod\_specializare = spec\_id AND s.cod\_spital = s\_id;

RETURN 'Spitalul ' || s\_id || ' are numarul de telefon ' || m\_info.nr\_tel\_spital || ' si pentru specializare ' || spec\_id ||

' pune la dispozitie medicul ' || m\_info.nume || ' cu salariul ' || m\_info.salariu;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RETURN 'Medicul interesant nu a fost gasit...';

WHEN TOO\_MANY\_ROWS THEN

RETURN 'Prea multi medici interesanti!!!';

END medic\_interesant;

/

BEGIN

--Output normal

DBMS\_OUTPUT.PUT\_LINE('Output Normal');

DBMS\_OUTPUT.PUT\_LINE(medic\_interesant(2, 2));

--NO\_DATA\_FOUND

DBMS\_OUTPUT.PUT\_LINE('Output NO\_DATA\_FOUND');

DBMS\_OUTPUT.PUT\_LINE(medic\_interesant(9, 7));

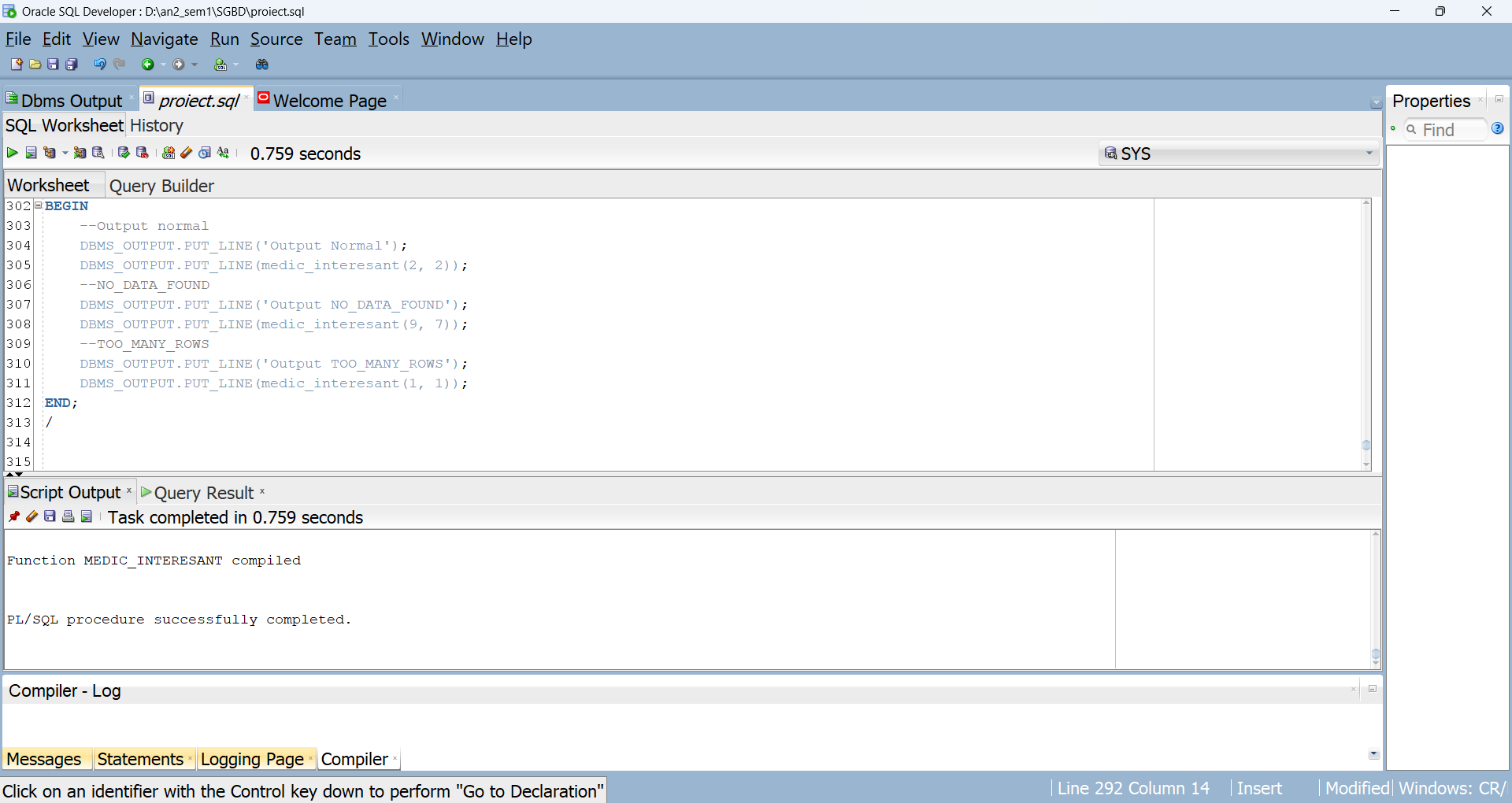
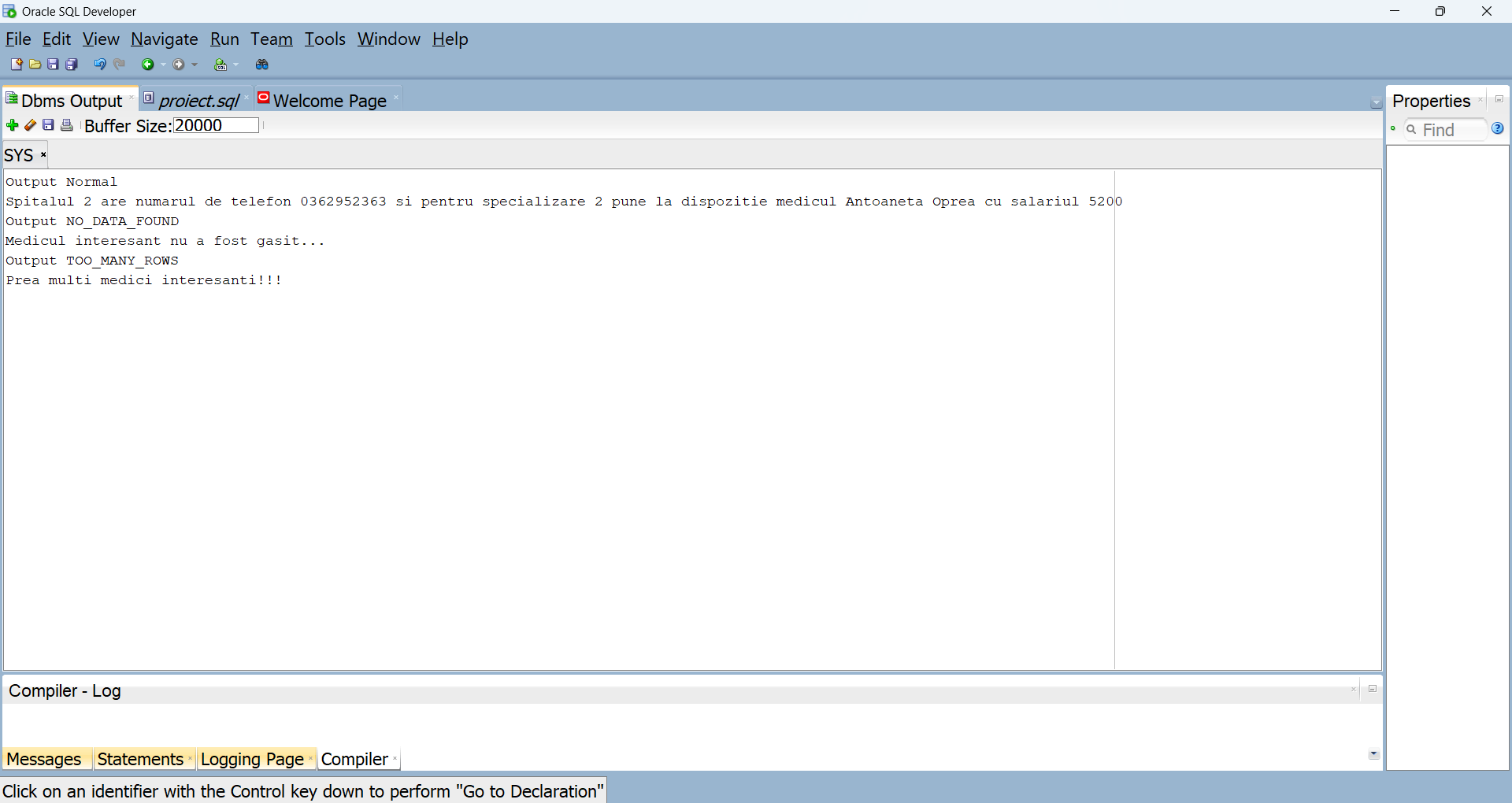
--TOO\_MANY\_ROWS

DBMS\_OUTPUT.PUT\_LINE('Output TOO\_MANY\_ROWS');

DBMS\_OUTPUT.PUT\_LINE(medic\_interesant(1, 1));

END;

/



/\*9. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat

independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură

comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele

predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite

și tratate.\*/

/\* Afisarea judetului din care face parte spitalul cu numarul de telefon pasat ca parametru impreuna cu numele medicului

ce are numarul de ani de experienta pasati ca parametru, salariul pe care il castiga la spitalul mentionat,

numele utilizatorului care este programat si durata programarii\*/

-- Pentru cazul TOO\_MANY\_ROWS

INSERT INTO programari VALUES (1, 4, 5, 1, SYSDATE, 30);

CREATE OR REPLACE PROCEDURE programari\_interesante(ani\_exp medici.ani\_experienta%TYPE, nr\_tel\_spital spitale.numar\_telefon%TYPE) IS

TYPE prog\_info IS RECORD(

nume\_judet judete.nume\_judet%TYPE,

nume\_utilizator utilizatori.nume\_utilizator%TYPE,

nume\_medic medici.nume\_medic%TYPE,

durata programari.durata%TYPE,

salariu\_medic spitale\_medici.salariu%TYPE);

p\_info prog\_info;

tel\_invalid Exception;

nr\_ani\_exp\_invalizi Exception;

BEGIN

IF ani\_exp < 0 OR ani\_exp > 50 THEN

RAISE nr\_ani\_exp\_invalizi;

END IF;

IF REGEXP\_LIKE(nr\_tel\_spital, '^\d+$') = False THEN

RAISE tel\_invalid;

END IF;

SELECT DISTINCT j.nume\_judet, u.nume\_utilizator, m.nume\_medic, p.durata, sm.salariu

INTO p\_info.nume\_judet, p\_info.nume\_utilizator, p\_info.nume\_medic, p\_info.durata, p\_info.salariu\_medic

FROM medici m

JOIN spitale\_medici sm ON m.cod\_medic = sm.cod\_medic

JOIN spitale s ON sm.cod\_spital = s.cod\_spital

JOIN judete j ON j.cod\_judet = s.cod\_judet

JOIN programari p ON (p.cod\_spital = s.cod\_spital AND p.cod\_medic = m.cod\_medic)

JOIN utilizatori u ON u.cod\_utilizator = p.cod\_utilizator

WHERE m.ani\_experienta = ani\_exp AND s.numar\_telefon = nr\_tel\_spital;

DBMS\_OUTPUT.PUT\_LINE('Programarea are o durata de ' || p\_info.durata || ' minute, pacientul ' || p\_info.nume\_utilizator ||

' va fi tratat de ' || p\_info.nume\_medic || ' , angajat al spitalului din judetul ' || p\_info.nume\_judet ||

' , ce are un salariu de ' || p\_info.salariu\_medic);

EXCEPTION

WHEN tel\_invalid THEN

DBMS\_OUTPUT.PUT\_LINE('Numar de telefon invalid!');

WHEN nr\_ani\_exp\_invalizi THEN

DBMS\_OUTPUT.PUT\_LINE('Numarul de ani primit nu este in limitele legale!');

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('Nu exista programari interesante...');

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('Prea multe programari interesante');

END;

/

BEGIN

DBMS\_OUTPUT.PUT\_LINE('Output normal');

programari\_interesante(1, '0218713860');

DBMS\_OUTPUT.PUT\_LINE('Output NO\_DATA\_FOUND');

programari\_interesante(25, '0211713863');

DBMS\_OUTPUT.PUT\_LINE('Output TOO\_MANY\_ROWS');

programari\_interesante(15, '0255249892');

DBMS\_OUTPUT.PUT\_LINE('Output telefon invalid');

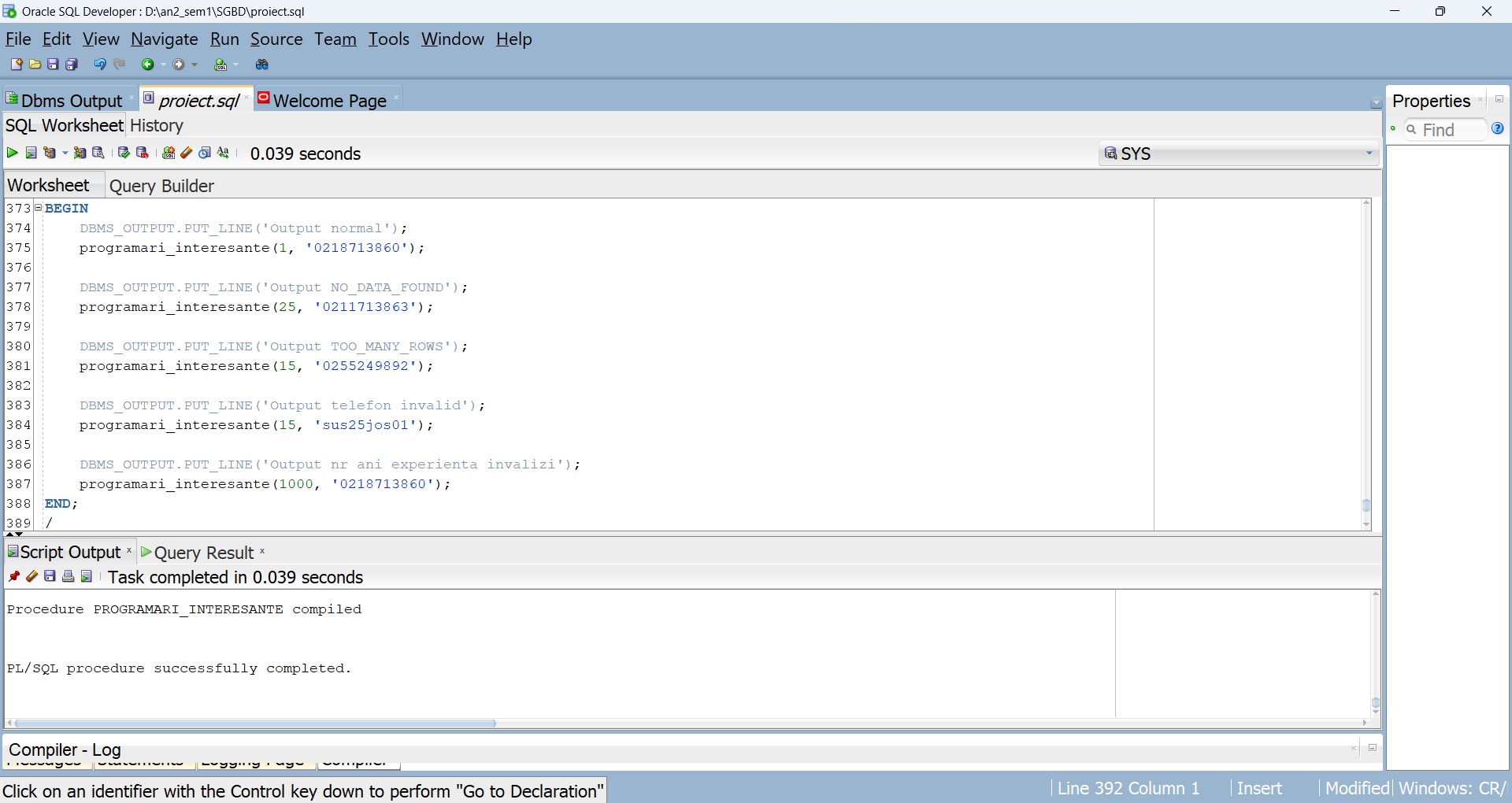
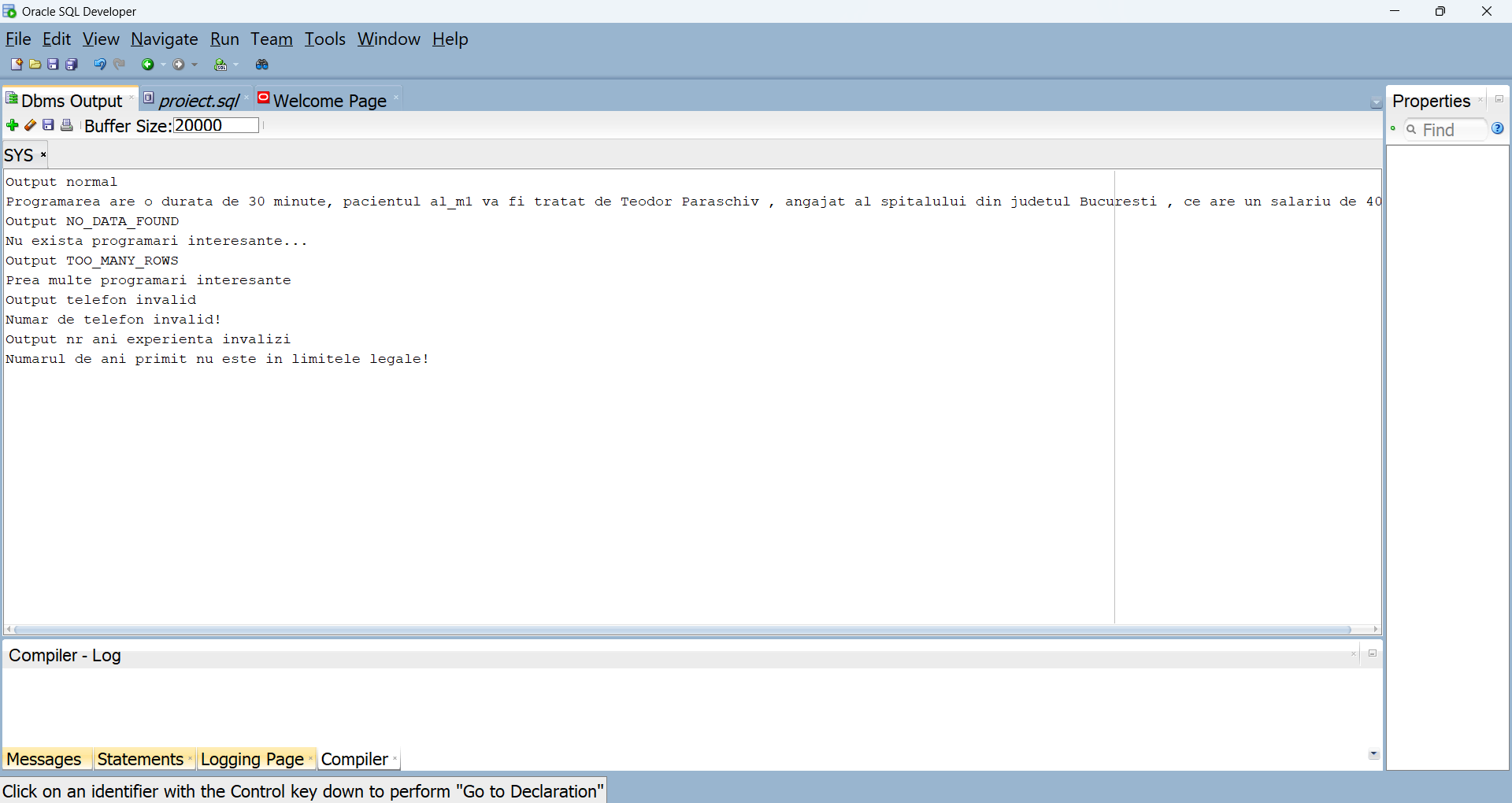
programari\_interesante(15, 'sus25jos01');

DBMS\_OUTPUT.PUT\_LINE('Output nr ani experienta invalizi');

programari\_interesante(1000, '0218713860');

END;

/



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

CREATE OR REPLACE TRIGGER maxim\_2\_m\_spec

AFTER INSERT ON spitale\_medici

DECLARE

CURSOR m\_cursor IS SELECT COUNT(cod\_medic) nr\_medici, cod\_spital FROM spitale\_medici

GROUP BY cod\_spital;

info m\_cursor%ROWTYPE;

BEGIN

OPEN m\_cursor;

LOOP

FETCH m\_cursor INTO info;

EXIT WHEN m\_cursor%NOTFOUND;

IF info.nr\_medici = 4 THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Prea multi medici la spitalul ' || info.cod\_spital);

END IF;

END LOOP;

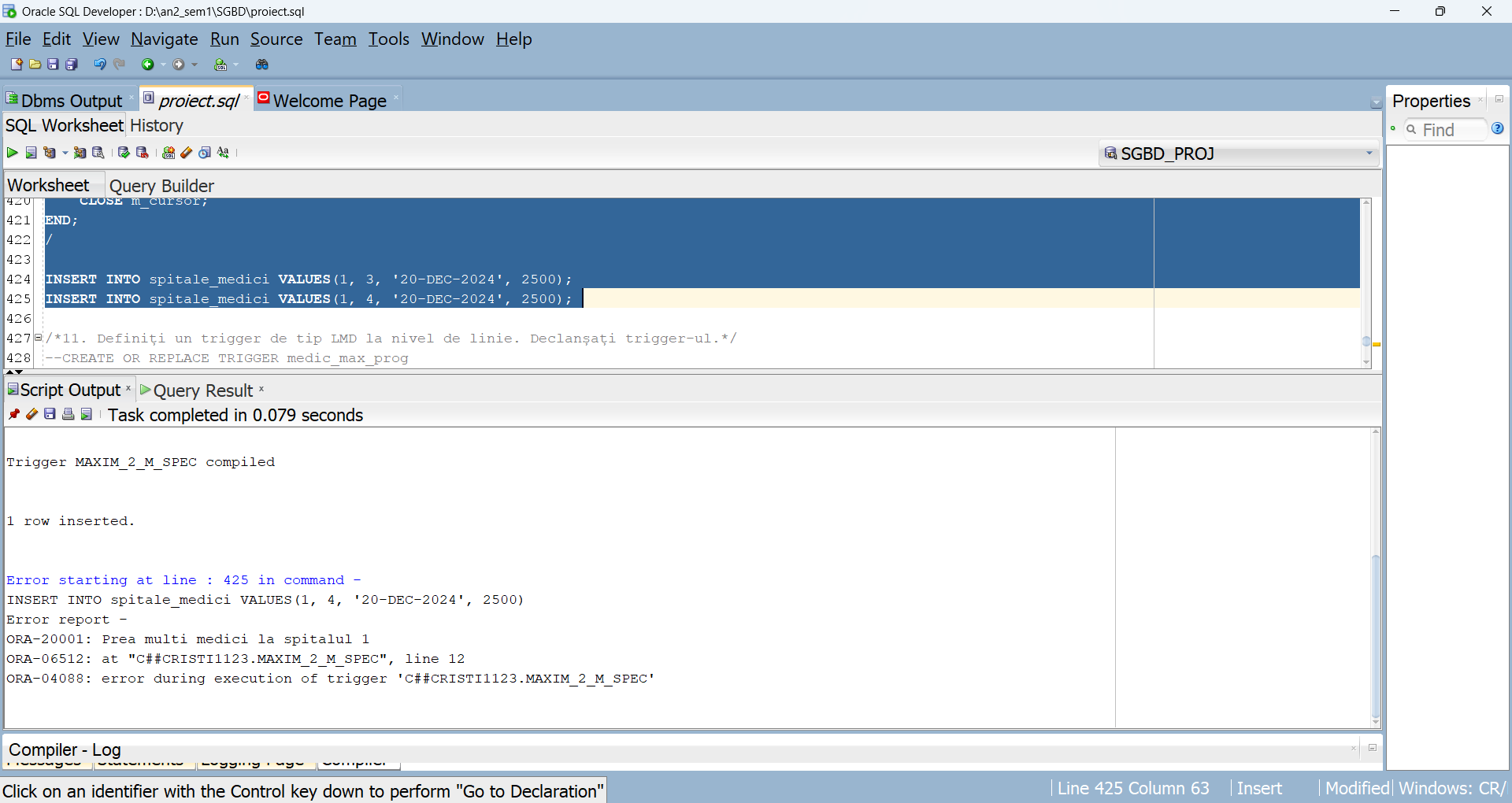
CLOSE m\_cursor;

END;

/

INSERT INTO spitale\_medici VALUES(1, 3, '20-DEC-2024', 2500);

INSERT INTO spitale\_medici VALUES(1, 4, '20-DEC-2024', 2500);



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

CREATE OR REPLACE TRIGGER medic\_max\_prog

BEFORE INSERT OR UPDATE ON programari

FOR EACH ROW

DECLARE

nr\_prog NUMBER;

BEGIN

SELECT COUNT(\*)

INTO nr\_prog

FROM programari

WHERE cod\_medic = :NEW.cod\_medic AND TRUNC(data\_programare) = TRUNC(:NEW.data\_programare);

IF nr\_prog >= 4 THEN

RAISE\_APPLICATION\_ERROR(-20002, 'Medicul acesta si-a atins numarul maxim de programari pentru o zi...');

END IF;

END medic\_max\_prog;

/

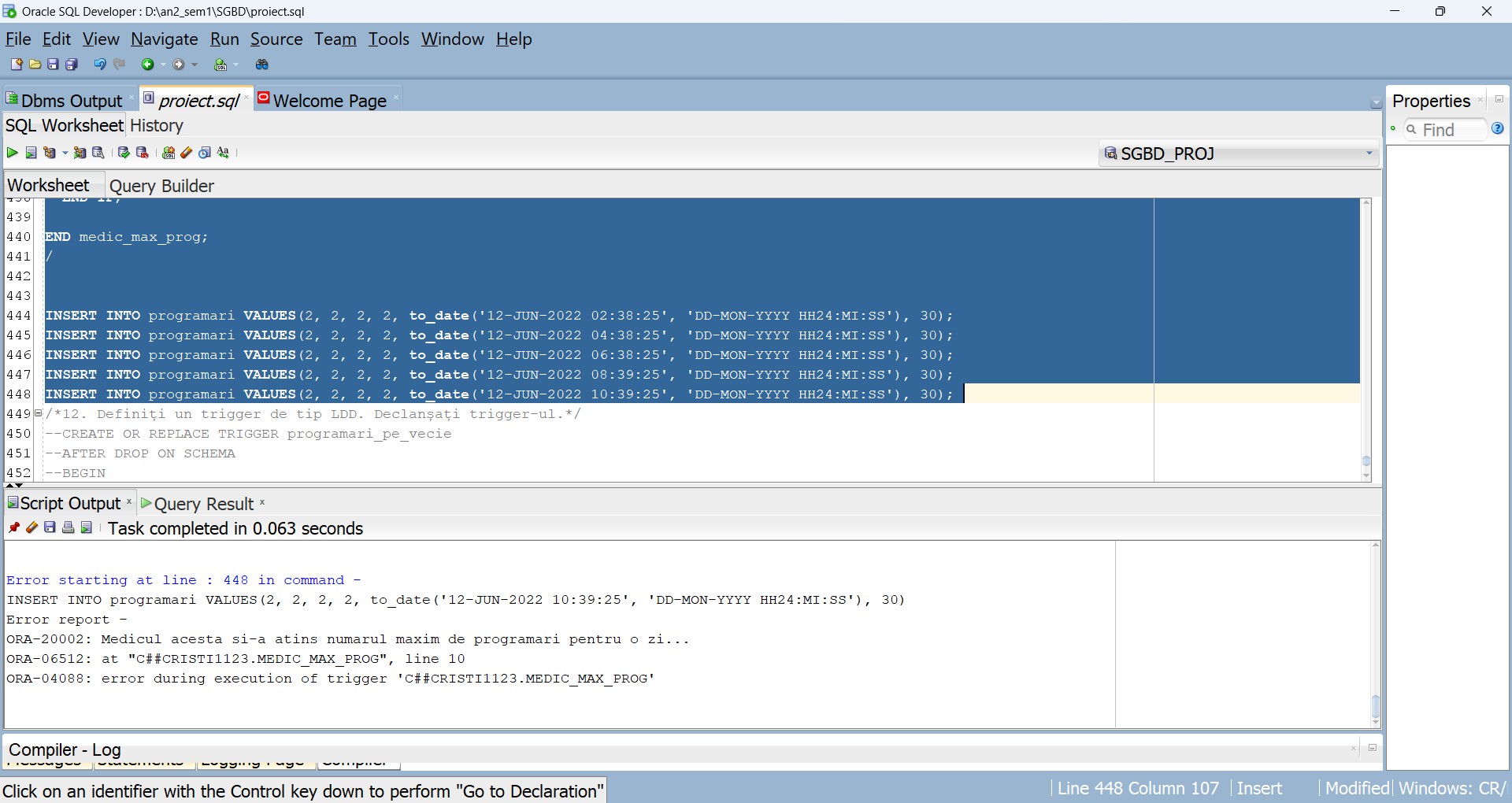
INSERT INTO programari VALUES(2, 2, 2, 2, to\_date('12-JUN-2022 02:38:25', 'DD-MON-YYYY HH24:MI:SS'), 30);

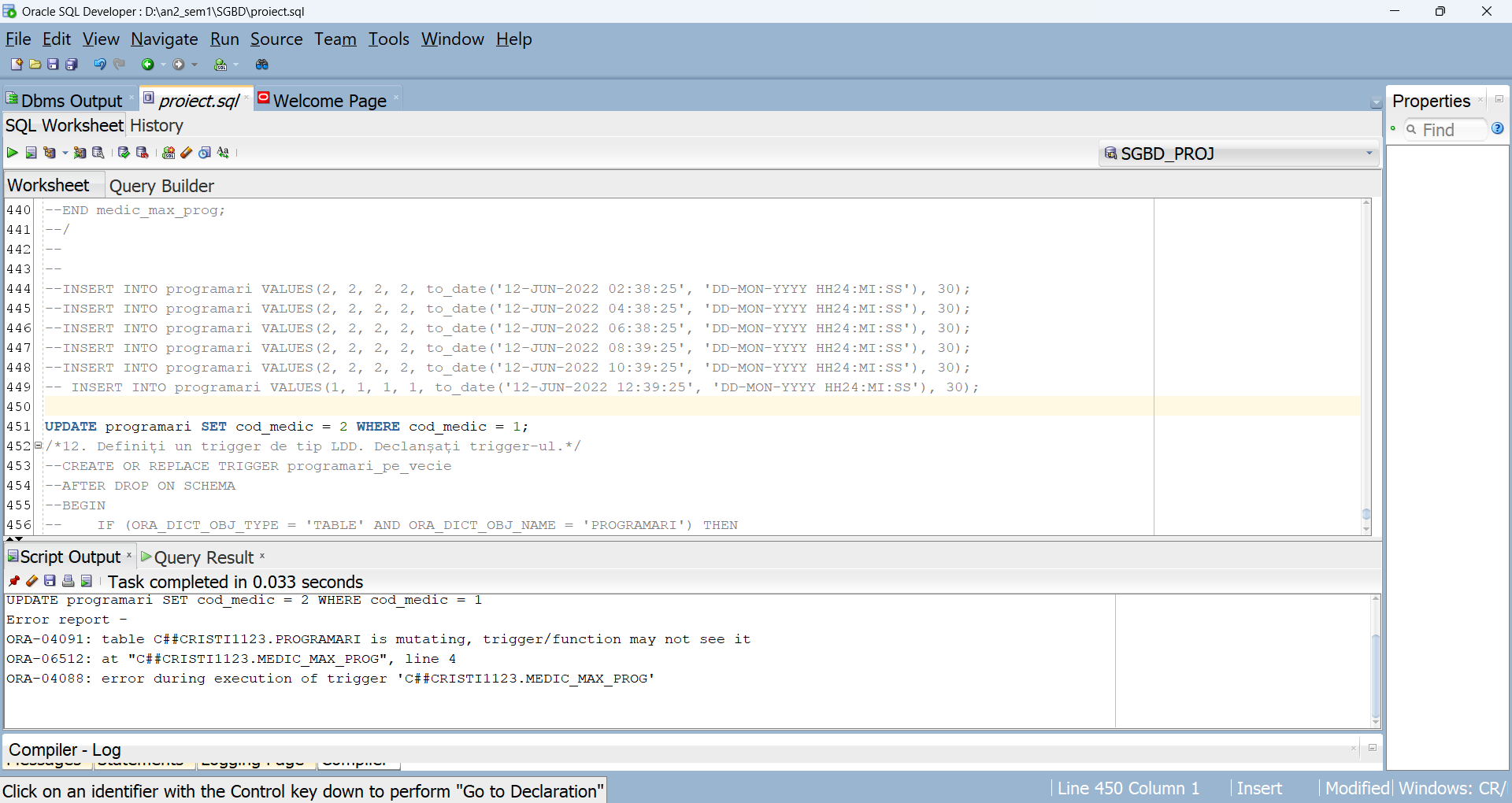
INSERT INTO programari VALUES(2, 2, 2, 2, to\_date('12-JUN-2022 04:38:25', 'DD-MON-YYYY HH24:MI:SS'), 30);

INSERT INTO programari VALUES(2, 2, 2, 2, to\_date('12-JUN-2022 06:38:25', 'DD-MON-YYYY HH24:MI:SS'), 30);

INSERT INTO programari VALUES(2, 2, 2, 2, to\_date('12-JUN-2022 08:39:25', 'DD-MON-YYYY HH24:MI:SS'), 30);

INSERT INTO programari VALUES(2, 2, 2, 2, to\_date('12-JUN-2022 10:39:25', 'DD-MON-YYYY HH24:MI:SS'), 30);

/\*Mutating table error la update\*/



/\*12. Definiți un trigger de tip LDD. Declanșați trigger-ul.\*/

CREATE OR REPLACE TRIGGER programari\_pe\_vecie

AFTER DROP ON SCHEMA

BEGIN

IF (ORA\_DICT\_OBJ\_TYPE = 'TABLE' AND ORA\_DICT\_OBJ\_NAME = 'PROGRAMARI') THEN

RAISE\_APPLICATION\_ERROR(-20001, 'Nu poti sterge tabelul programari...');

END IF;

END;

/

DROP TABLE programari;

