Proiect SGBD

Platforma de E-Learning

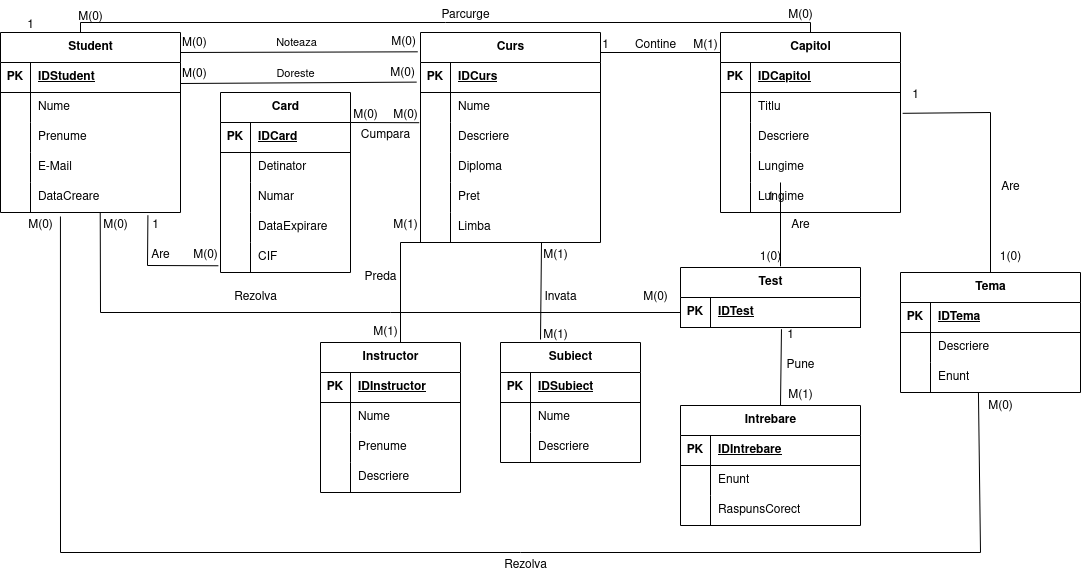
Miclea Alexandru – Grupa 231

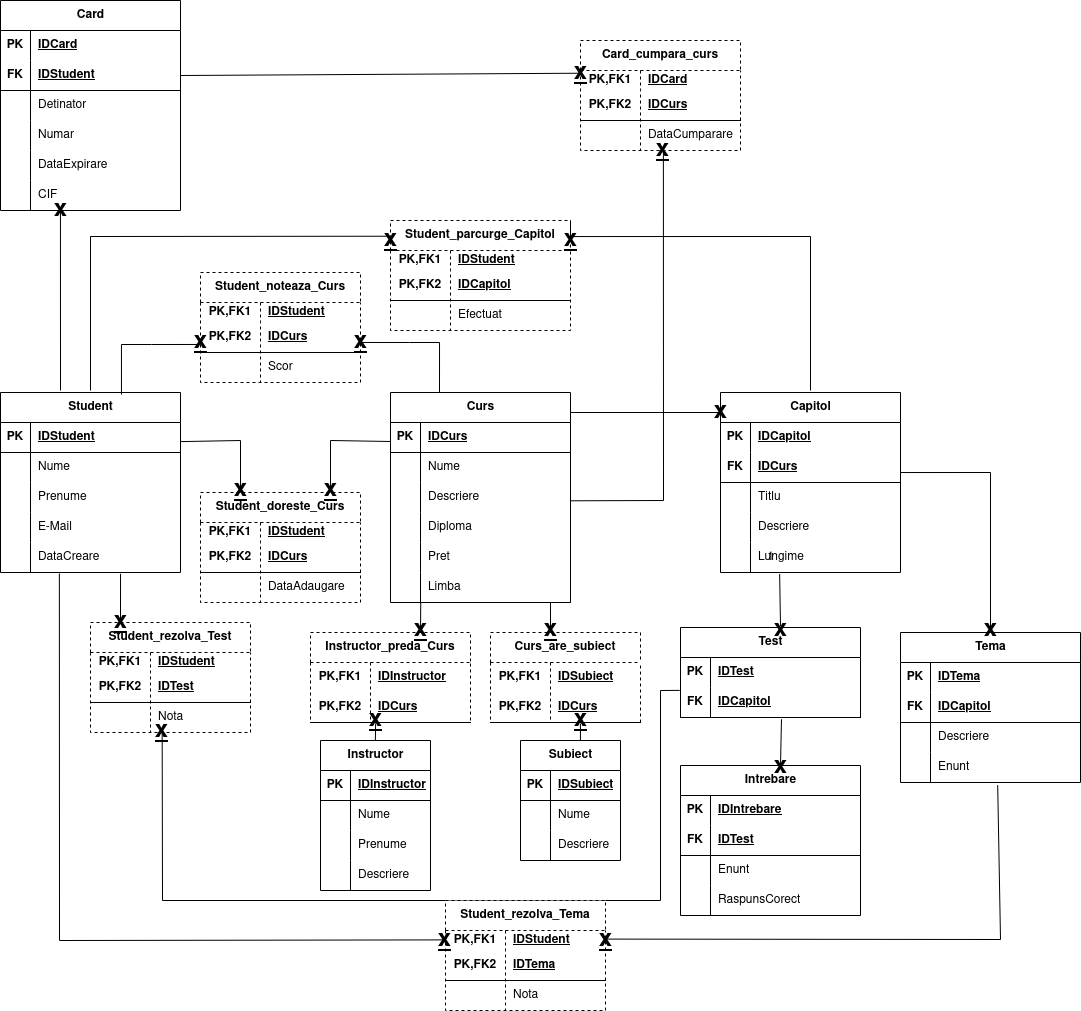
**1. Prezentați pe scurt baza de date (utilitatea ei)**

Modelul real al bazei de date este inspirat din platforme de E-Learning (Udemy, Coursera, etc.) disponibile pe internet. In acestea, studentii isi pot face un cont cu care sa cumpere acces la cursuri. Cursurile pot fi despre mai multe topicuri, precum programare, gatit, filozofie etc. Fiecare curs are mai multe capitole, fiecare capitol avand la randul lui teme si teste. Studentul poate sa adauge cursuri in wishlist, sa isi verifice progresul, sau sa adauge carduri de plata cu care va cumpara cursuri.

Utilitatea bazei de date este data de nevoia de a avea date referitoare la progresul unui student in cadrul cursurilor, de a le gestiona intr-un mod eficient si de a oferi posibilitatea acesarii intr-un mod facil a metricelor de interes pentru fiecare student. Totodata, pe baza acestor date, instructorii pot avea acces la date in timp real despre progresul studentilor lor.

**2. Realizați diagrama entitate-relație (ERD)**



**3. Pornind de la diagrama entitate-relație, realizați diagrama conceptuală** **4.** I**mplementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, definind toate constrângerile de integritate necesare (chei primare, cheile externe etc).**

create table Student (

idstudent int,

nume varchar2(40) constraint nn\_nume\_student not null,

prenume varchar2(40) constraint nn\_prenume\_student not null,

email varchar2(100) constraint nn\_email\_student not null,

datacreare date default to\_date(sysdate, 'DD-MM-YYYY'),

constraint pk\_student primary key(idstudent),

constraint uq\_email\_student unique(email)

);

create TABLE card (

idcard int,

idstudent int,

detinator varchar2(100) constraint nn\_detinator\_card not null,

numar number(16) constraint nn\_numar\_card not null,

dataexpirare date constraint nn\_dataexpirare\_card not null,

cif number(3) constraint nn\_cif\_card not null,

constraint pk\_card primary key(idcard),

constraint fk\_card\_student foreign key(idstudent) references student(idstudent) on delete cascade,

constraint uq\_numar\_card unique(numar)

);

create table Curs (

idcurs int,

nume varchar2(100) constraint nn\_nume\_curs not null,

descriere varchar2(400),

diploma number default 0 constraint nn\_diploma\_curs not null,

pret number(5,2) constraint nn\_pret\_curs not null,

limba varchar2(20),

constraint pk\_curs primary key(idcurs)

);

create table card\_cumpara\_curs (

idcard int,

idcurs int,

datacumparare date DEFAULT to\_date(sysdate, 'DD-MM-YYYY'),

constraint pk\_ccc primary key(idcard, idcurs),

constraint fk\_ccc\_card foreign key(idcard) references card(idcard) on delete cascade,

constraint fk\_ccc\_curs foreign key(idcurs) references curs(idcurs) on delete cascade

);

create table capitol (

idcapitol int,

idcurs int,

titlu varchar2(40) constraint nn\_titlu\_capitol not null,

descriere VARCHAR2(400),

lungime number(5,2) constraint nn\_lungime\_capitol not null,

constraint pk\_capitol primary key(idcapitol),

constraint fk\_capitol\_curs foreign key(idcurs) references curs(idcurs) on delete cascade

);

create table test (

idtest int,

idcapitol int,

constraint pk\_test primary key(idtest),

constraint fk\_test\_capitol foreign key(idcapitol) references capitol(idcapitol) on delete cascade

);

create table tema (

idtema int,

idcapitol int,

descriere VARCHAR2(400),

enunt VARCHAR2(400) constraint nn\_enunt\_tema not null,

constraint pk\_tema primary key(idtema),

constraint fk\_tema\_capitol foreign key(idcapitol) references capitol(idcapitol) on delete cascade

);

create table subiect (

idsubiect int,

nume VARCHAR2(40) constraint nn\_nume\_subiect not null,

descriere VARCHAR2(400),

constraint pk\_subiect primary key(idsubiect)

);

create table instructor (

idinstructor int,

nume varchar2(40) constraint nn\_nume\_instructor not null,

prenume varchar2(40) constraint nn\_prenume\_instructor not null,

descriere VARCHAR2(400),

constraint pk\_instructor primary key(idinstructor)

);

create table intrebare (

idintrebare int,

idtest int,

enunt VARCHAR2(400) constraint nn\_enunt\_intrebare not null,

raspunscorect VARCHAR2(400) constraint nn\_raspunscorect\_tema not null,

constraint pk\_intrebare primary key(idintrebare)

);

create table instructor\_preda\_curs (

idinstructor int,

idcurs int,

constraint pk\_ipc primary key(idinstructor, idcurs),

constraint fk\_ipc\_instructor foreign key(idinstructor) references instructor(idinstructor) on delete cascade,

constraint fk\_ipc\_curs foreign key(idcurs) references curs(idcurs) on delete cascade

);

create table curs\_are\_subiect (

idcurs int,

idsubiect int,

constraint pk\_cas primary key(idcurs, idsubiect),

constraint fk\_cas\_curs foreign key(idcurs) references curs(idcurs) on delete cascade,

constraint fk\_cas\_subiect foreign key(idsubiect) references subiect(idsubiect) on delete cascade

);

create table student\_rezolva\_tema (

idstudent int,

idtema int,

nota number(4,2),

constraint pk\_srtema primary key(idstudent, idtema),

constraint fk\_srtema\_student foreign key(idstudent) references student(idstudent) on delete cascade,

constraint fk\_srtema\_tema foreign key(idtema) references tema(idtema) on delete cascade

);

create table student\_rezolva\_test (

idstudent int,

idtest int,

nota number(4,2),

constraint pk\_srtest primary key(idstudent, idtest),

constraint fk\_srtest\_student foreign key(idstudent) references student(idstudent) on delete cascade,

constraint fk\_srtest\_test foreign key(idtest) references test(idtest) on delete cascade

);

create table student\_doreste\_curs (

idstudent int,

idcurs int,

dataadaugare date DEFAULT to\_date(sysdate, 'DD-MM-YYYY'),

constraint pk\_sdc primary key(idstudent, idcurs),

constraint fk\_sdc\_student foreign key(idstudent) references student(idstudent) on delete cascade,

constraint fk\_sdc\_curs foreign key(idcurs) references curs(idcurs) on delete cascade

);

create table student\_noteaza\_curs (

idstudent int,

idcurs int,

nota number(4,2),

constraint pk\_snc primary key(idstudent, idcurs),

constraint fk\_snc\_student foreign key(idstudent) references student(idstudent) on delete cascade,

constraint fk\_snc\_curs foreign key(idcurs) references curs(idcurs) on delete cascade

);

create table student\_parcurge\_capitol (

idstudent int,

idcapitol int,

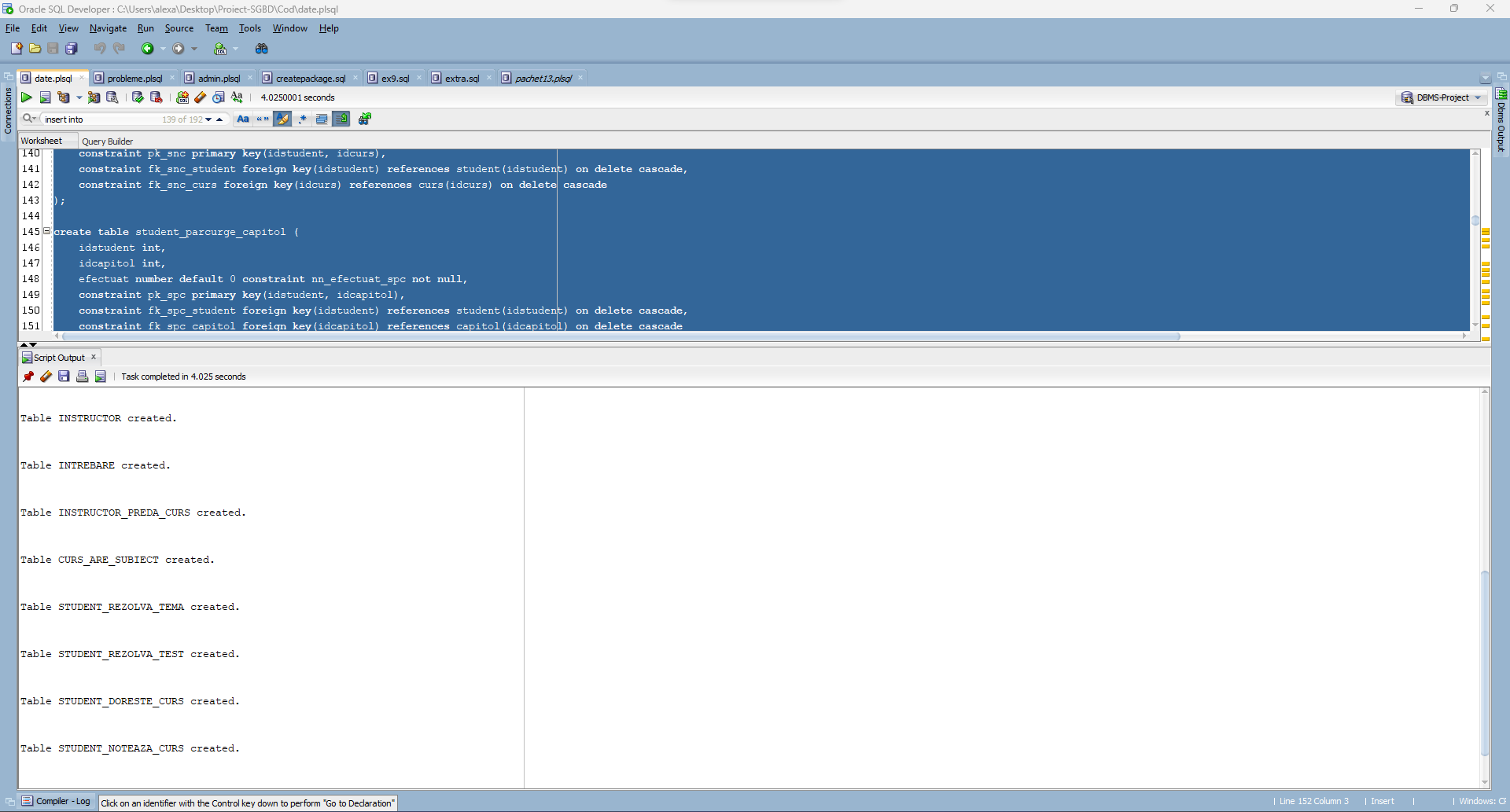
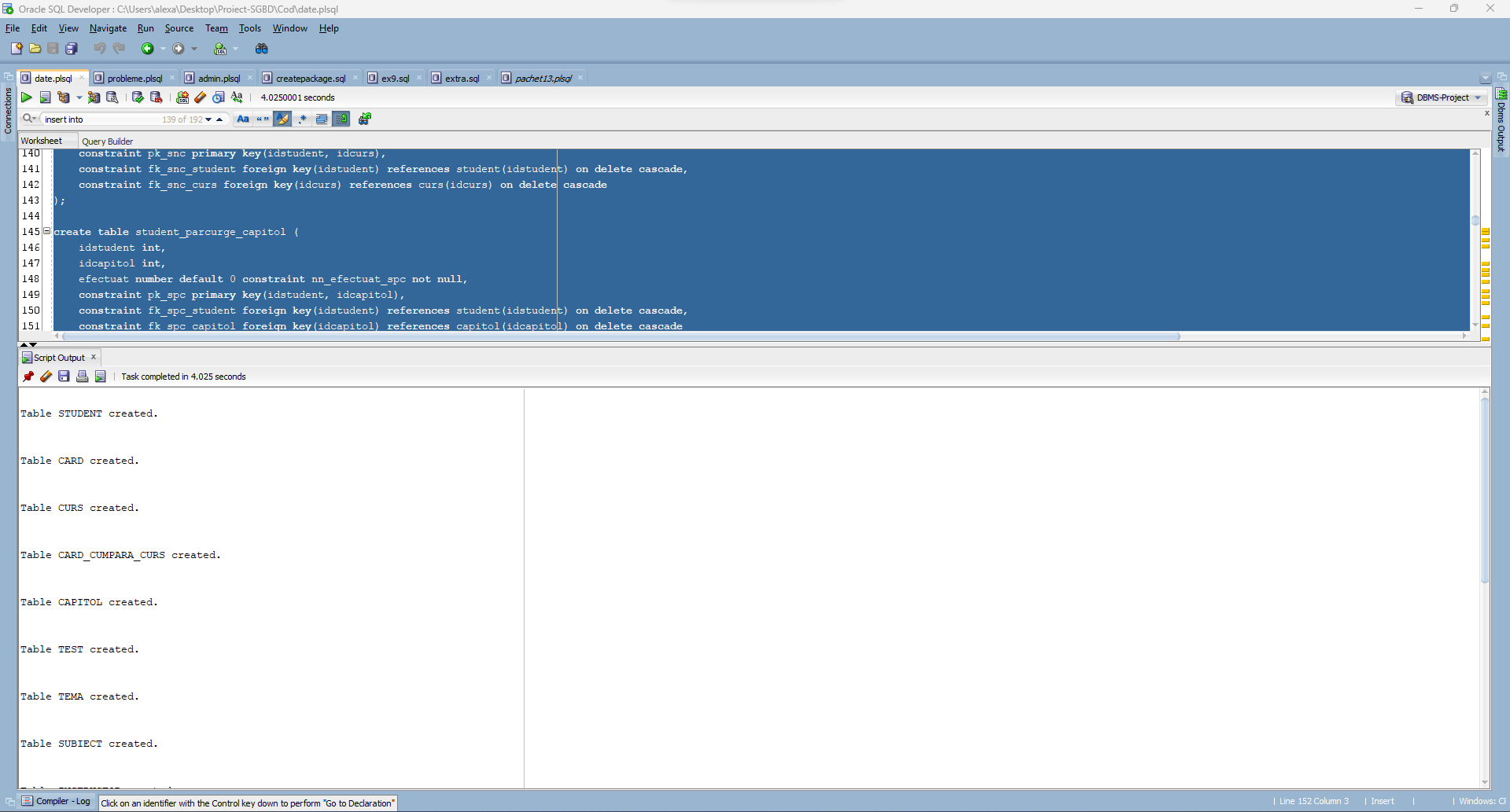
efectuat number default 0 constraint nn\_efectuat\_spc not null,

constraint pk\_spc primary key(idstudent, idcapitol),

constraint fk\_spc\_student foreign key(idstudent) references student(idstudent) on delete cascade,

constraint fk\_spc\_capitol foreign key(idcapitol) references capitol(idcapitol) on delete cascade

);



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

create sequence seq\_student start with 8 increment by 1;

insert into student (idstudent, nume, prenume, email, datacreare) values (1, 'Chirila', 'Alexandru Matei', 'chirilaalexandrumatei@outlook.com', sysdate - 30);

insert into student (idstudent, nume, prenume, email, datacreare) values (2, 'Costiniu', 'Gabriel', 'costiniugabriel@gmail.com', sysdate - 20);

insert into student (idstudent, nume, prenume, email, datacreare) values (3, 'Timandi', 'Karina', 'timandiliviaandreea@hotmail.com', sysdate - 150);

insert into student (idstudent, nume, prenume, email, datacreare) values (4, 'Stinga', 'Madalina', 'stingamadalina@gmail.com', sysdate - 10);

insert into student (idstudent, nume, prenume, email, datacreare) values (5, 'Balitiu', 'Teodora', 'balitiuteodora@outlook.com', sysdate - 360);

insert into student (idstudent, nume, prenume, email, datacreare) values (6, 'Stanasila', 'Ovidiu', 'stanasilaovidiu@protonmail.com', sysdate - 90);

insert into student (idstudent, nume, prenume, email, datacreare) values (7, 'Banica', 'Raul Cezar', 'banicaraulcezar@gmail.com', sysdate - 60);

select \* from student;

create sequence seq\_curs start with 6 increment by 1;

insert into curs (idcurs, nume, descriere, diploma, pret, limba) values (1, '100 days of code', 'Ia-o de la zero cu programarea in Python! Pe parcursul celor 100 de zile vei avea ceva de codat zilnic.', 1, 49.99, 'engleza');

insert into curs (idcurs, nume, descriere, diploma, pret, limba) values (2, 'Gateste cu Sylvester Stallone', 'Esentialul in gatit, astazi predat de catre nimeni altul decat Sylvester Stallone!', 0, 69.99, 'engleza');

insert into curs (idcurs, nume, descriere, diploma, pret, limba) values (3, 'Lectii de trompeta', 'Un instrument versatil, trompeta poate indulci orice coloana sonora.', 1, 39.99, 'romana');

insert into curs (idcurs, nume, descriere, diploma, pret, limba) values (4, 'Filozofie si etica academica', null, 0, 29.99, 'romana');

insert into curs (idcurs, nume, descriere, diploma, pret, limba) values (5, 'Chitara electrica 101', 'Totul despre chitara electrica. Invata sa canti rock, metal, whatever...', 1, 59.99, 'engleza');

select \* from curs;

create sequence seq\_card start with 11 increment by 1;

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (1, 1, 'CHIRILA ALEXANDRU MATEI', 374245455400126, to\_date('01/05/2026', 'DD/MM/YYYY'), 123);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (2, 2, 'COSTINIU GABRIEL', 378282246310005, to\_date('01/05/2026', 'DD/MM/YYYY'), 423);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (3, 3, 'TIMANDI KARINA', 6250941006528599, to\_date('01/06/2026', 'DD/MM/YYYY'), 434);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (4, 3, 'TIMANDI CRISTIAN', 6011000180331112, to\_date('01/02/2026', 'DD/MM/YYYY'), 543);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (5, 4, 'STINGA MADALINA', 6011000991300009, to\_date('01/12/2026', 'DD/MM/YYYY'), 564);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (6, 5, 'BALITIU TEODORA', 3566000020000410, to\_date('01/02/2026', 'DD/MM/YYYY'), 954);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (7, 6, 'STANASILA ANDREEA', 3530111333300000, to\_date('01/09/2026', 'DD/MM/YYYY'), 309);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (8, 6, 'STANASILA MARIAN', 5425233430109903, to\_date('01/02/2026', 'DD/MM/YYYY'), 534);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (9, 7, 'BANICA RAUL CEZAR', 2222420000001113, to\_date('01/03/2026', 'DD/MM/YYYY'), 890);

insert into card (idcard, idstudent, detinator, numar, dataexpirare, cif) values (10, 7, 'BANICA ADELINA', 5789432795823472, to\_date('01/04/2026', 'DD/MM/YYYY'), 654);

select \* from card;

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (1, 1, sysdate - 27);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (1, 4, sysdate - 23);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (2, 2, sysdate - 10);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (3, 3, sysdate - 70);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (4, 5, sysdate - 40);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (6, 1, sysdate - 150);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (6, 2, sysdate - 150);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (7, 3, sysdate - 85);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (8, 4, sysdate - 60);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (9, 4, sysdate - 50);

insert into card\_cumpara\_curs (idcard, idcurs, datacumparare) values (10, 5, sysdate - 30);

select \* from card\_cumpara\_curs;

create sequence seq\_subiect start with 10 increment by 1;

insert into subiect (idsubiect, nume, descriere) values (1, 'Programare', null);

insert into subiect (idsubiect, nume, descriere) values (2, 'Python', null);

insert into subiect (idsubiect, nume, descriere) values (3, 'Gatit', null);

insert into subiect (idsubiect, nume, descriere) values (4, 'Instrumente', null);

insert into subiect (idsubiect, nume, descriere) values (5, 'Filozofie', null);

insert into subiect (idsubiect, nume, descriere) values (6, 'Academic', null);

insert into subiect (idsubiect, nume, descriere) values (7, 'Chitara', null);

insert into subiect (idsubiect, nume, descriere) values (8, 'Arama', null);

insert into subiect (idsubiect, nume, descriere) values (9, 'Actorie', null);

select \* from subiect;

insert into curs\_are\_subiect (idcurs, idsubiect) values (1, 1);

insert into curs\_are\_subiect (idcurs, idsubiect) values (1, 2);

insert into curs\_are\_subiect (idcurs, idsubiect) values (2, 3);

insert into curs\_are\_subiect (idcurs, idsubiect) values (2, 9);

insert into curs\_are\_subiect (idcurs, idsubiect) values (3, 3);

insert into curs\_are\_subiect (idcurs, idsubiect) values (3, 8);

insert into curs\_are\_subiect (idcurs, idsubiect) values (4, 5);

insert into curs\_are\_subiect (idcurs, idsubiect) values (4, 6);

insert into curs\_are\_subiect (idcurs, idsubiect) values (5, 4);

insert into curs\_are\_subiect (idcurs, idsubiect) values (5, 7);

select \* from curs\_are\_subiect;

create sequence seq\_instructor start with 11 increment by 1;

insert into instructor (idinstructor, nume, prenume, descriere) values (1, 'Blidariu', 'Mihnea', 'Mihnea Blidariu (n. 6 septembrie 1979, Bacau, Romania) este un muzician roman, membru al formatiei rock Luna Amara, din Cluj-Napoca. Interpreteaza vocal, la trompeta si la chitara.');

insert into instructor (idinstructor, nume, prenume, descriere) values (2, 'Stallone', 'Sylvester', 'Sylvester Enzio Stallone (n. 6 iulie 1946, New York City, New York, SUA) este un actor, regizor si scenarist american.');

insert into instructor (idinstructor, nume, prenume, descriere) values (3, 'Yu', 'Angela', 'I am Angela, I am a developer with a passion for teaching. I am the lead instructor at the London App Brewery, a leading Programming Bootcamp.');

insert into instructor (idinstructor, nume, prenume, descriere) values (4, 'Stoenescu', 'Constantin', 'Prof. Dr.');

insert into instructor (idinstructor, nume, prenume, descriere) values (5, 'Brancoveanu', 'Romulus', 'Prof. Dr.');

insert into instructor (idinstructor, nume, prenume, descriere) values (6, 'Patrunsu', 'Dorina Mihaela', 'Prof. Dr.');

insert into instructor (idinstructor, nume, prenume, descriere) values (7, 'Cioaba', 'Catalin', 'Prof. Dr.');

insert into instructor (idinstructor, nume, prenume, descriere) values (8, 'Malan', 'David', null);

insert into instructor (idinstructor, nume, prenume, descriere) values (9, 'Botan', 'Andrei', null);

insert into instructor (idinstructor, nume, prenume, descriere) values (10, 'Fagadar', 'Nick', 'Nick Fagadar este un muzician roman, membru al formatiei rock Luna Amara, din Cluj-Napoca. Interpreteaza vocal si la chitara.');

select \* from instructor;

insert into instructor\_preda\_curs (idinstructor, idcurs) values (3, 1);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (8, 1);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (2, 2);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (1, 3);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (4, 4);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (5, 4);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (6, 4);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (7, 4);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (9, 5);

insert into instructor\_preda\_curs (idinstructor, idcurs) values (10, 5);

select \* from instructor\_preda\_curs;

create sequence seq\_capitol start with 8 increment by 1;

insert into capitol (idcapitol, idcurs, titlu, descriere, lungime) values (1, 1, 'Python introduction', null, 109.35);

insert into capitol (idcapitol, idcurs, titlu, descriere, lungime) values (2, 1, 'Daily assignments', null, 65.21);

insert into capitol (idcapitol, idcurs, titlu, descriere, lungime) values (3, 2, 'Salads with Adrian Balboa', null, 36.56);

insert into capitol (idcapitol, idcurs, titlu, descriere, lungime) values (4, 2, 'Steak. Meats.', null, 36.56);

insert into capitol (idcapitol, idcurs, titlu, descriere, lungime) values (5, 3, 'Ritmuri de hora in Sol Major', null, 20.15);

insert into capitol (idcapitol, idcurs, titlu, descriere, lungime) values (6, 4, 'Filozofia in Inteligenta Artificiala', null, 300.15);

insert into capitol (idcapitol, idcurs, titlu, descriere, lungime) values (7, 5, 'Stilul Grunge', null, 60.25);

select \* from capitol;

create sequence seq\_test start with 12 increment by 1;

insert into test (idtest, idcapitol) values (1, 1);

insert into test (idtest, idcapitol) values (2, 1);

insert into test (idtest, idcapitol) values (3, 1);

insert into test (idtest, idcapitol) values (4, 3);

insert into test (idtest, idcapitol) values (5, 4);

insert into test (idtest, idcapitol) values (6, 5);

insert into test (idtest, idcapitol) values (7, 6);

insert into test (idtest, idcapitol) values (8, 6);

insert into test (idtest, idcapitol) values (9, 6);

insert into test (idtest, idcapitol) values (10, 6);

insert into test (idtest, idcapitol) values (11, 7);

select \* from test;

create sequence seq\_intrebare start with 23 increment by 1;

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (1, 1, 'What is a list?', 'Data collection');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (2, 1, 'What does print() do?', 'Prints contents to the STDOUT');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (3, 2, 'What is the difference between a list and a tuple?', 'Tuples are immutable, lists are not');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (4, 2, 'How to reverse a list using slices?', 'list = list[::-1]');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (5, 3, 'Is backtracking efficient?', 'No.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (6, 3, 'What is the most optimal sorting complexity?' , 'O(n)');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (7, 4, 'Ceaser salad. Dressing?', 'Light dressing.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (8, 4, 'Should you add red fruits to salad?', 'Yes (but only a couple slices).');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (9, 5, 'Should you cook steak beyond medium-rare?', 'Yes, but only certain cuts.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (10, 5, 'Does lamb go well with wine?', 'The greeks have been doing it for centuries so of course yes.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (11, 6, 'Ce hora este cea mai populara in Romania?', 'Hora Unirii.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (13, 7, 'Are viata sfarsit?', 'In conceptia religiei, nu.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (15, 8, 'Ce este moralitatea?', 'Depinde pe cine intrebi.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (17, 9, 'Este inteligenta artificiala un domeniu ce reprezinta interes academic?', 'Da.');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (19, 10, 'Reprezinta inteligenta arificiala nesupervizata un pericol?', 'Nu (se cer detalii).');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (21, 11, 'Care a fost prima trupa considerata Grunge?', 'Green River');

insert into intrebare (idintrebare, idtest, enunt, raspunscorect) values (22, 11, 'Care a fost cea mai populara trupa Grunge?', 'Nirvana');

select \* from intrebare;

create sequence seq\_tema start with 9 increment by 1;

insert into tema (idtema, idcapitol, descriere, enunt) values (1, 1, null, 'Create a Python script that implements as many concepts as possible.');

insert into tema (idtema, idcapitol, descriere, enunt) values (2, 2, null, 'Day 20: Nth Fibonacci number');

insert into tema (idtema, idcapitol, descriere, enunt) values (3, 2, null, 'Day 40: First 100 prime numbers');

insert into tema (idtema, idcapitol, descriere, enunt) values (4, 2, null, 'Day 60: Cash register');

insert into tema (idtema, idcapitol, descriere, enunt) values (5, 2, null, 'Day 80: Bank administration');

insert into tema (idtema, idcapitol, descriere, enunt) values (6, 2, null, 'Day 100: print("I did it!");');

insert into tema (idtema, idcapitol, descriere, enunt) values (7, 5, null, 'Filmeaza ritmul tau');

insert into tema (idtema, idcapitol, descriere, enunt) values (8, 7, null, 'Canta o melodie preferata din stilul grunge');

select \* from tema;

insert into student\_noteaza\_curs (idstudent, idcurs) values (1, 1);

insert into student\_noteaza\_curs (idstudent, idcurs) values (2, 2);

insert into student\_noteaza\_curs (idstudent, idcurs) values (3, 3);

insert into student\_noteaza\_curs (idstudent, idcurs) values (5, 1);

insert into student\_noteaza\_curs (idstudent, idcurs) values (6, 3);

insert into student\_noteaza\_curs (idstudent, idcurs) values (7, 4);

insert into student\_noteaza\_curs (idstudent, idcurs, nota) values (1, 4, 8.00);

insert into student\_noteaza\_curs (idstudent, idcurs, nota) values (3, 5, 9.00);

insert into student\_noteaza\_curs (idstudent, idcurs, nota) values (5, 2, 10.00);

insert into student\_noteaza\_curs (idstudent, idcurs, nota) values (6, 4, 7.50);

insert into student\_noteaza\_curs (idstudent, idcurs, nota) values (7, 5, 3.00);

select \* from student\_noteaza\_curs;

insert into student\_doreste\_curs (idstudent, idcurs) values (4, 1);

insert into student\_doreste\_curs (idstudent, idcurs) values (4, 2);

insert into student\_doreste\_curs (idstudent, idcurs) values (4, 3);

insert into student\_doreste\_curs (idstudent, idcurs) values (1, 3);

insert into student\_doreste\_curs (idstudent, idcurs) values (3, 4);

insert into student\_doreste\_curs (idstudent, idcurs) values (5, 5);

insert into student\_doreste\_curs (idstudent, idcurs) values (6, 1);

insert into student\_doreste\_curs (idstudent, idcurs) values (6, 2);

insert into student\_doreste\_curs (idstudent, idcurs) values (7, 1);

insert into student\_doreste\_curs (idstudent, idcurs) values (7, 2);

select \* from student\_doreste\_curs;

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (1, 1, 1);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (1, 2, 0);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (1, 6, 0);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (2, 3, 1);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (2, 4, 0);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (3, 5, 0);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (3, 7, 0);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (5, 1, 1);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (5, 2, 1);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (5, 3, 1);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (5, 4, 0);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (6, 5, 1);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (6, 6, 0);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (7, 6, 1);

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (7, 7, 1);

select \* from student\_parcurge\_capitol;

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (1, 1, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (1, 2, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (1, 3, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema) values (1, 4);

insert into student\_rezolva\_tema (idstudent, idtema) values (1, 5);

insert into student\_rezolva\_tema (idstudent, idtema) values (1, 6);

insert into student\_rezolva\_tema (idstudent, idtema) values (3, 7);

insert into student\_rezolva\_tema (idstudent, idtema) values (3, 8);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (5, 1, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (5, 2, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (5, 3, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (5, 4, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (5, 5, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (5, 6, 10.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (6, 5, 9.00);

insert into student\_rezolva\_tema (idstudent, idtema, nota) values (7, 8, 8.00);

select \* from student\_rezolva\_tema;

insert into student\_rezolva\_test (idstudent, idtest, nota) values (1, 1, 10.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (1, 2, 9.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (1, 3, 10.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (1, 7, 7.00);

insert into student\_rezolva\_test (idstudent, idtest) values (1, 8);

insert into student\_rezolva\_test (idstudent, idtest) values (1, 9);

insert into student\_rezolva\_test (idstudent, idtest) values (1, 10);

insert into student\_rezolva\_test (idstudent, idtest) values (2, 4);

insert into student\_rezolva\_test (idstudent, idtest) values (3, 6);

insert into student\_rezolva\_test (idstudent, idtest) values (3, 11);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (5, 1, 10.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (5, 2, 10.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (5, 3, 10.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (5, 4, 10.00);

insert into student\_rezolva\_test (idstudent, idtest) values (5, 5);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (6, 6, 10.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (6, 7, 6.00);

insert into student\_rezolva\_test (idstudent, idtest) values (6, 8);

insert into student\_rezolva\_test (idstudent, idtest) values (6, 9);

insert into student\_rezolva\_test (idstudent, idtest) values (6, 10);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (7, 7, 9.00);

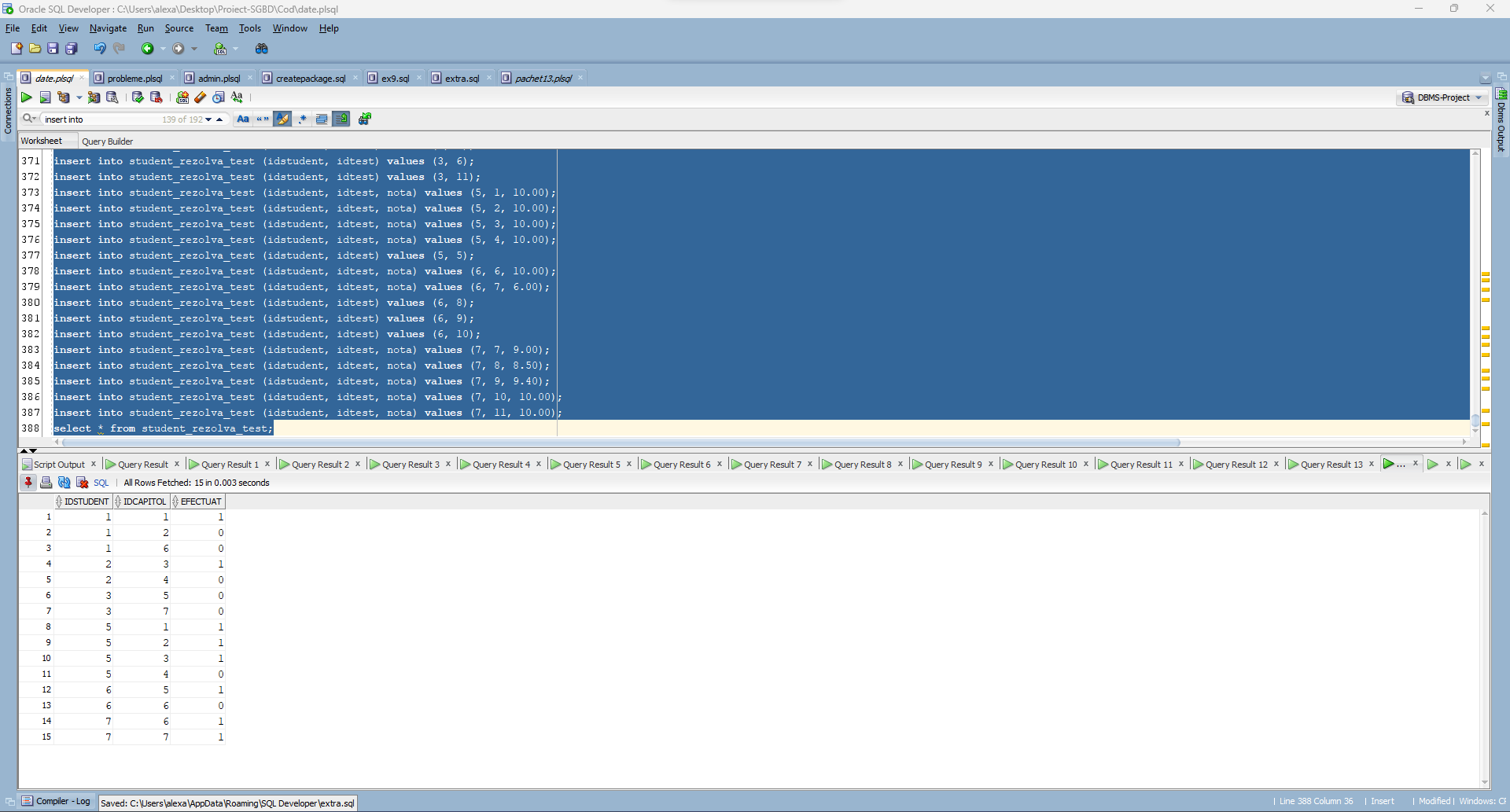
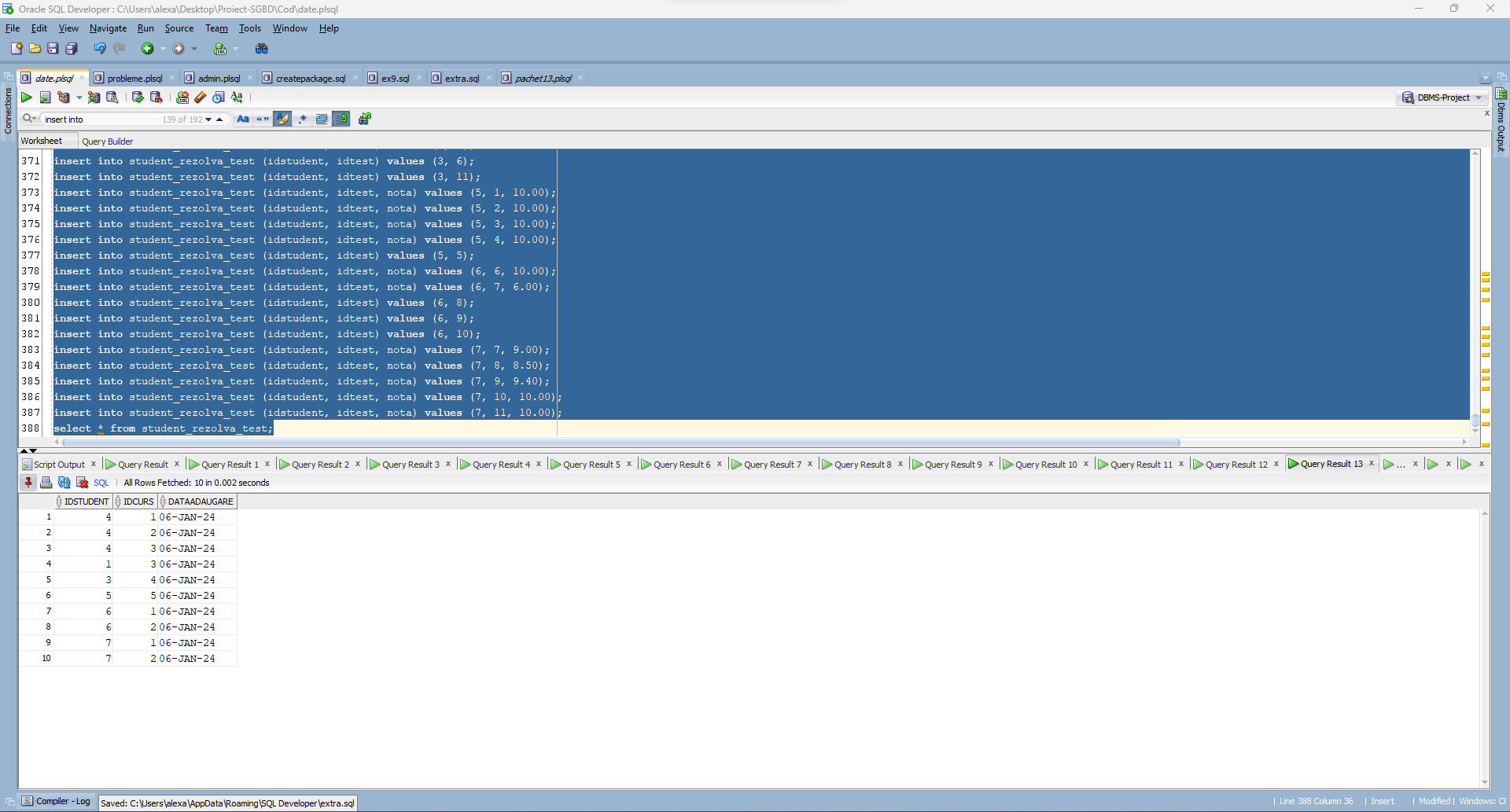
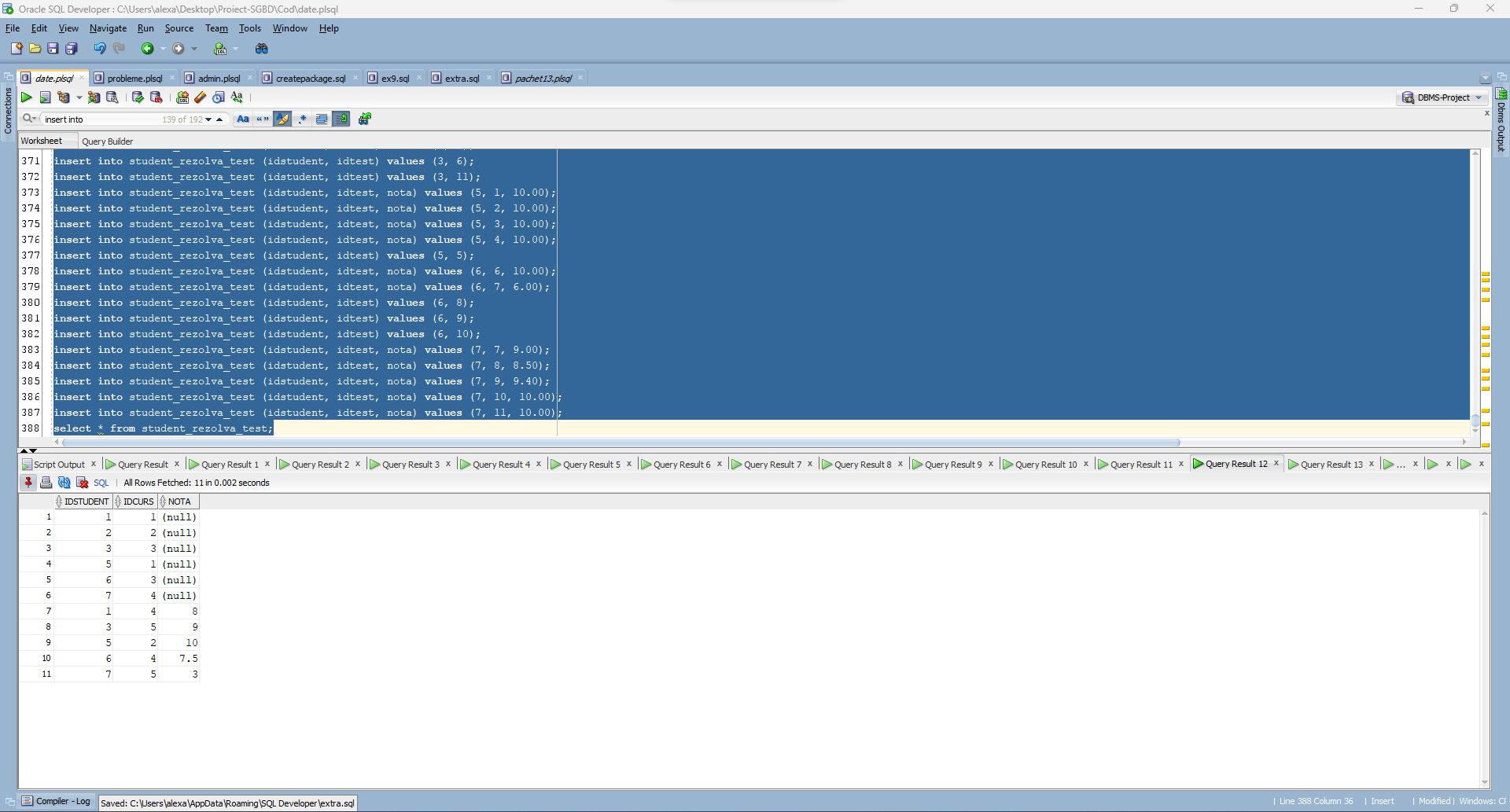
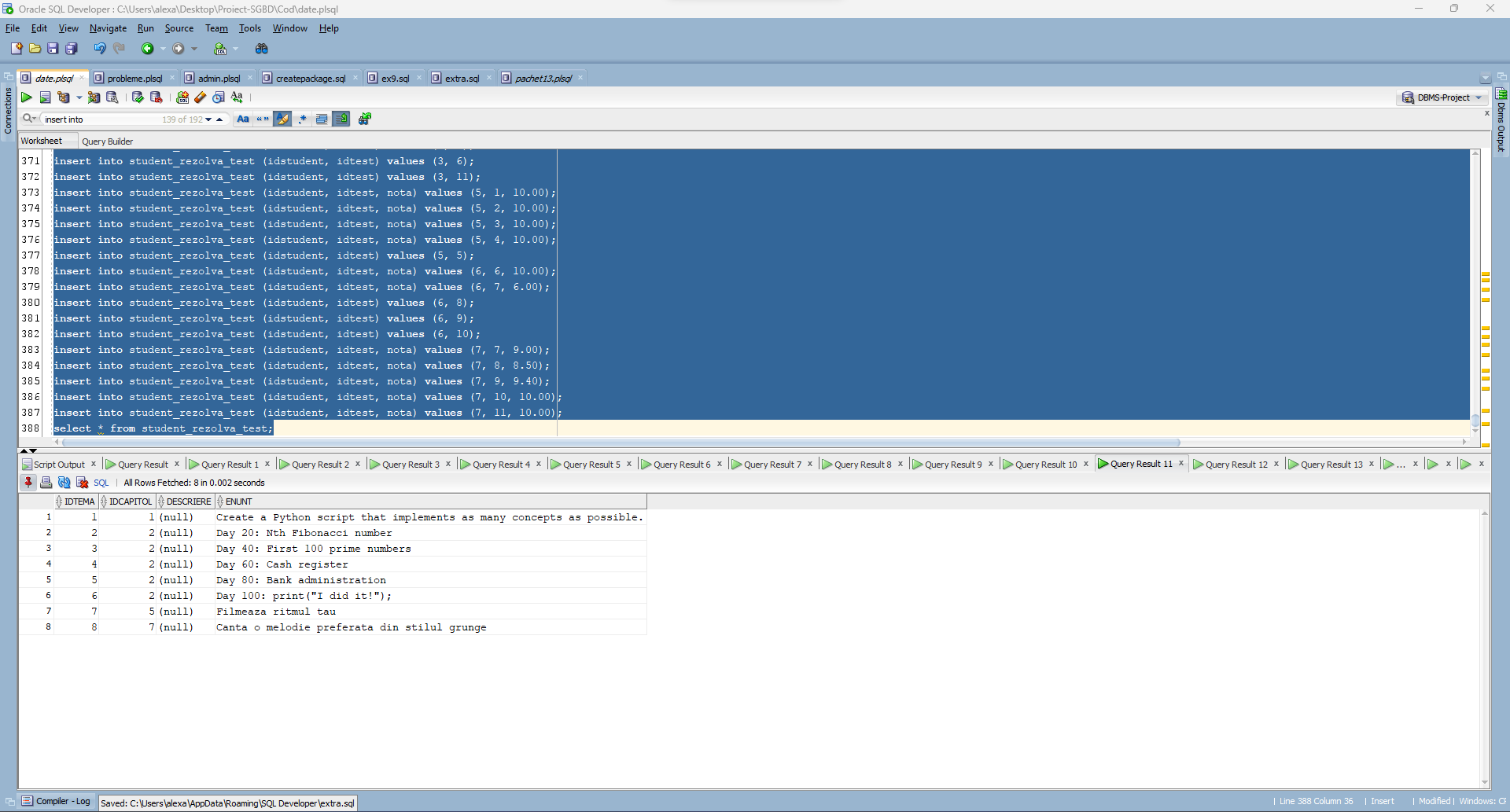
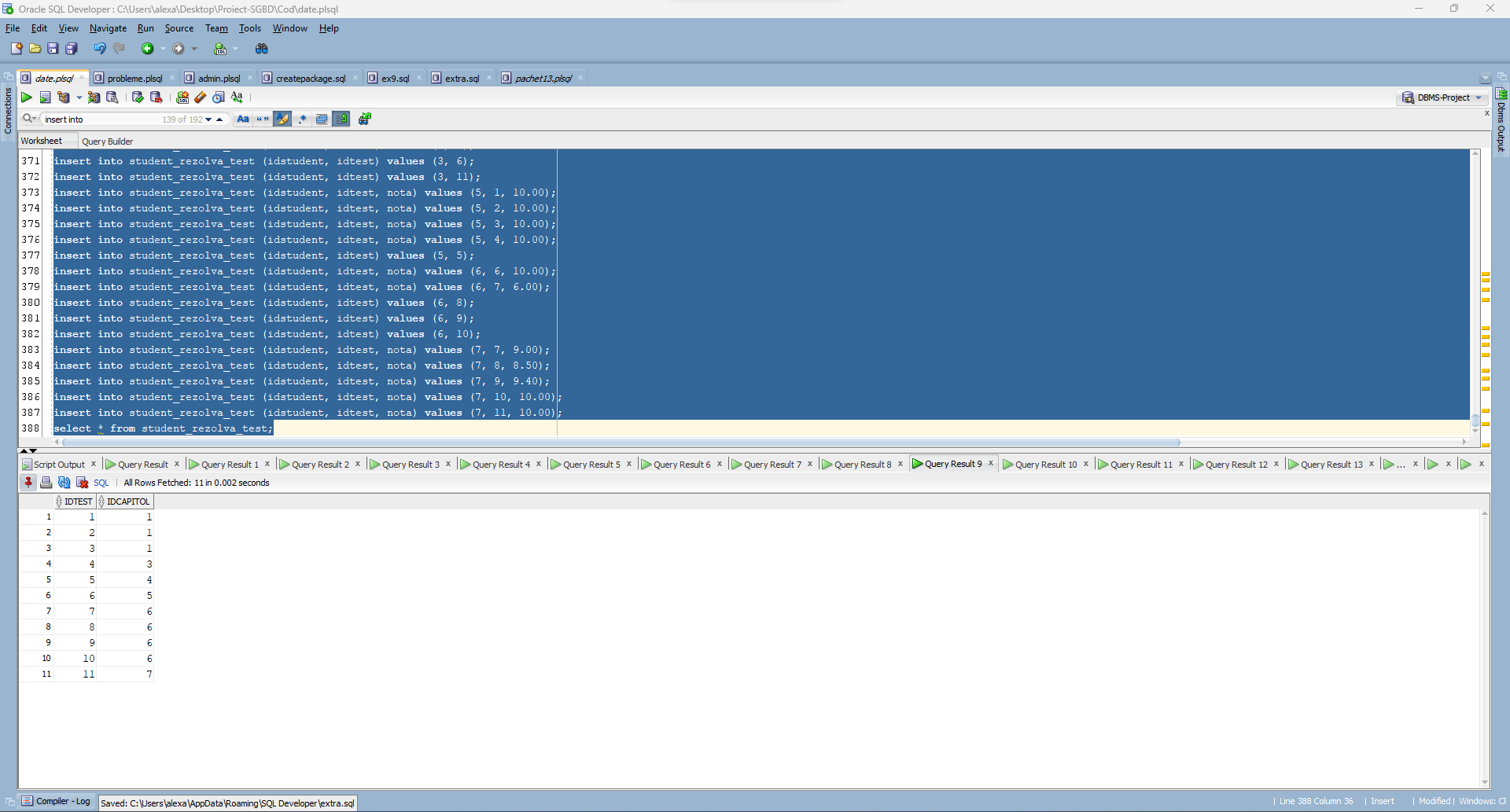
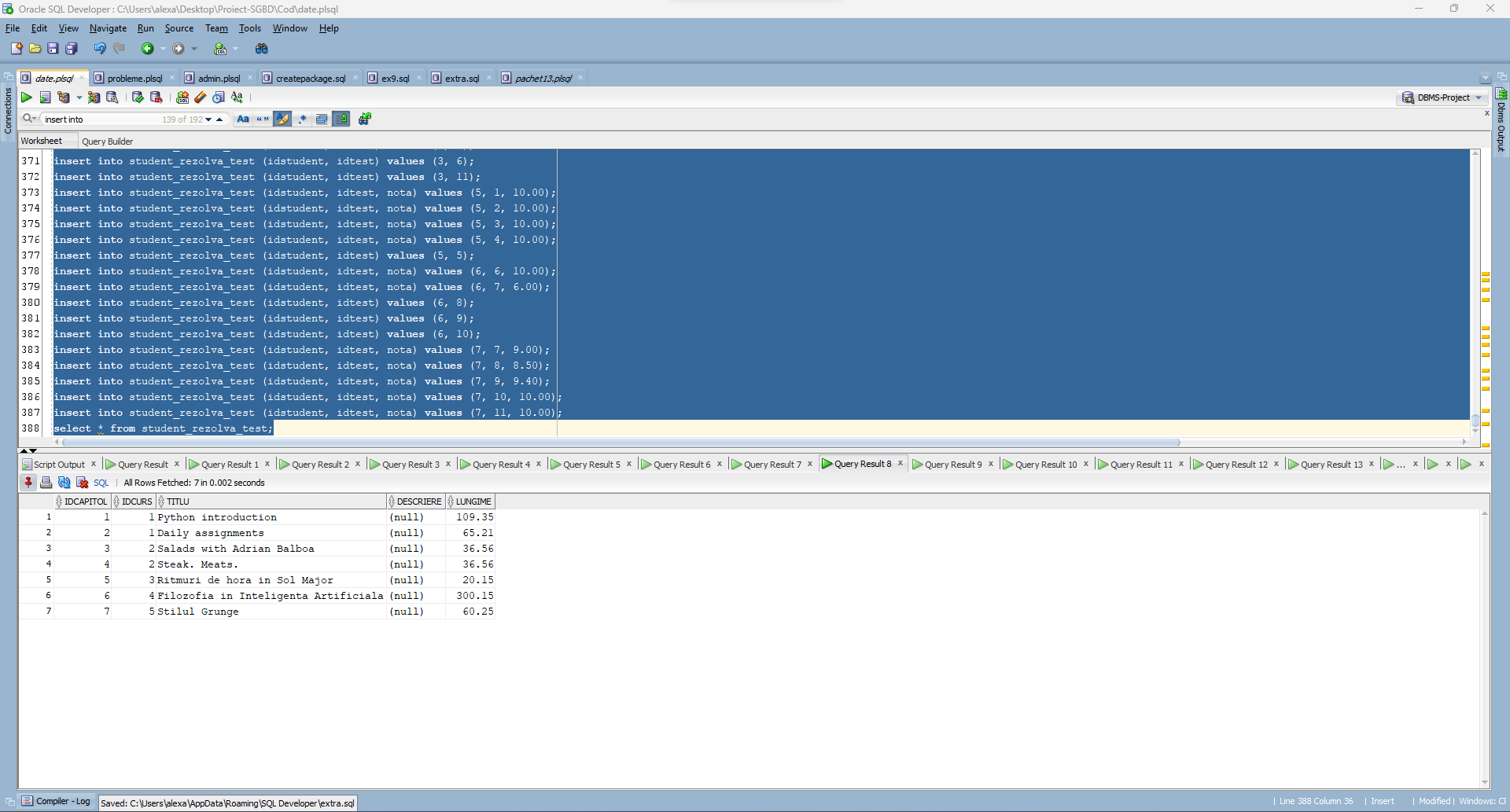
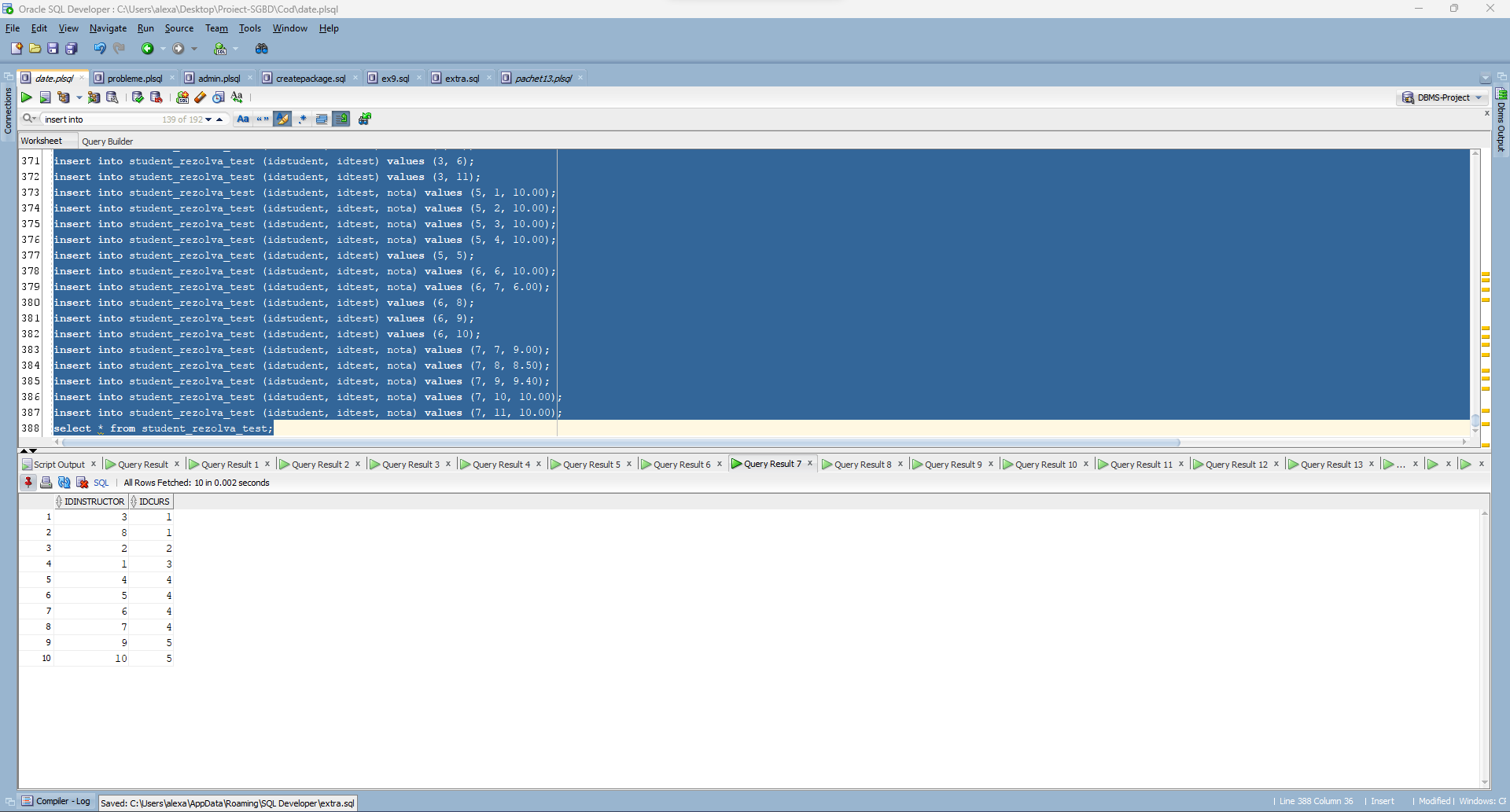
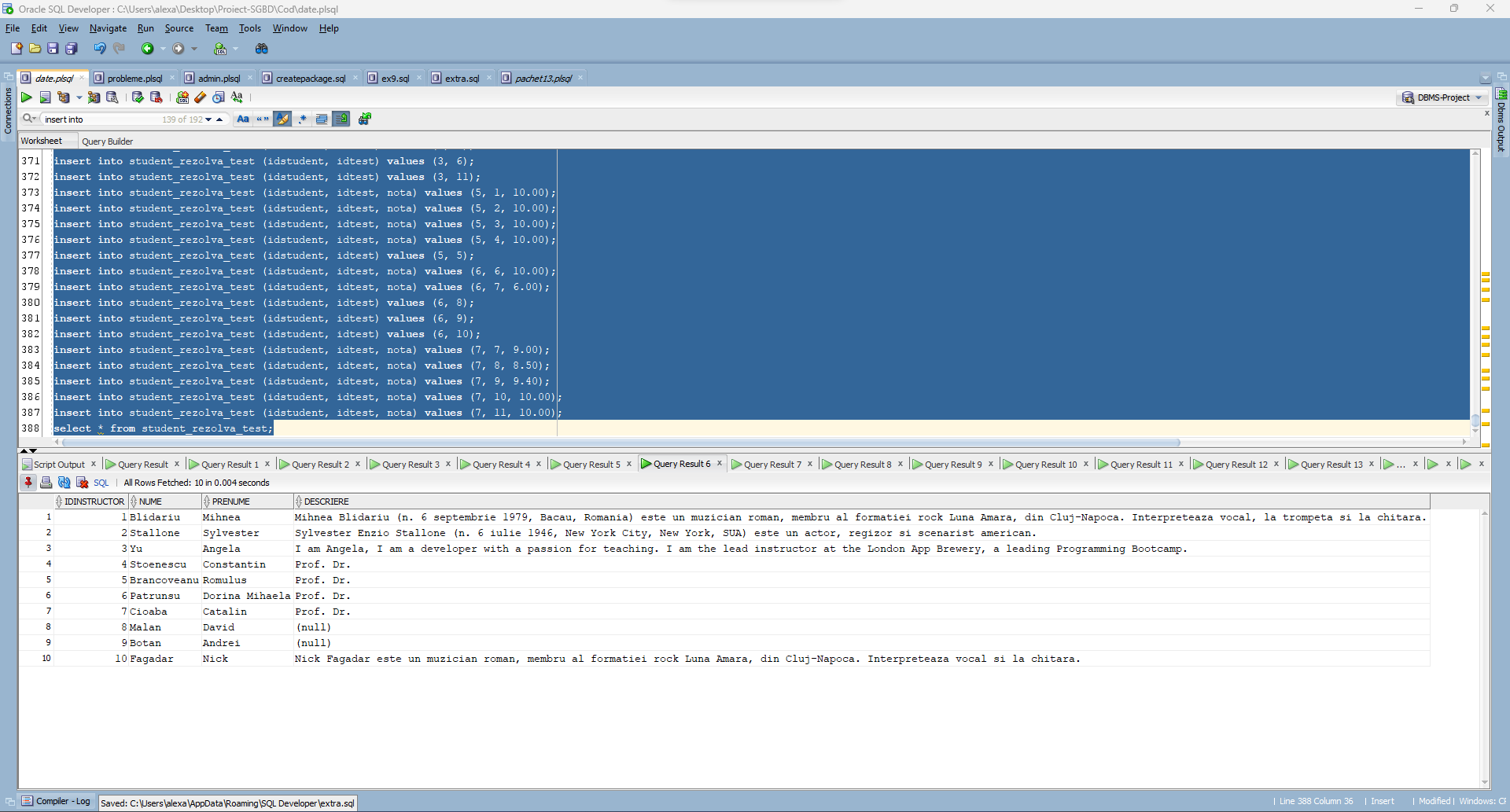
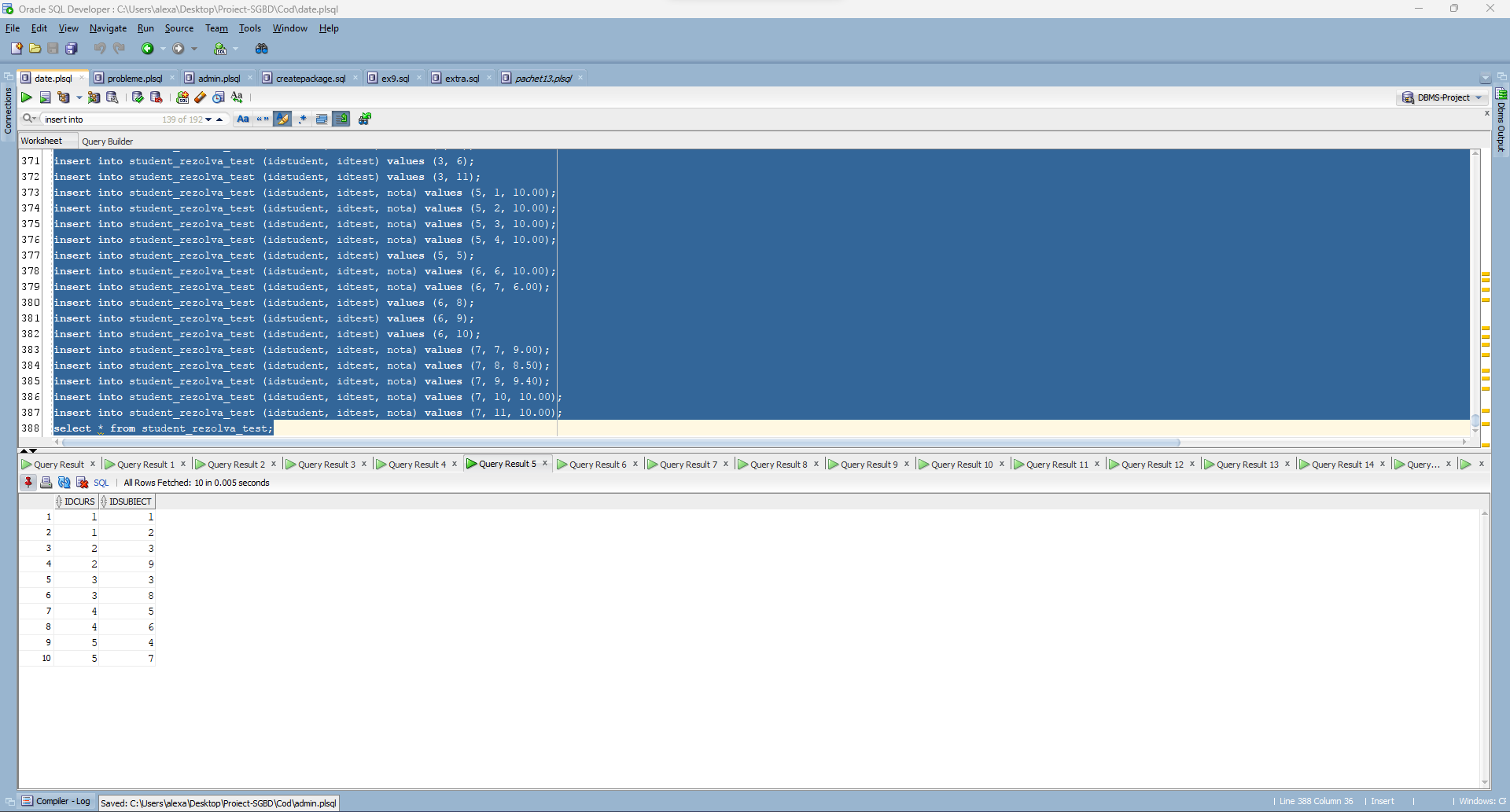
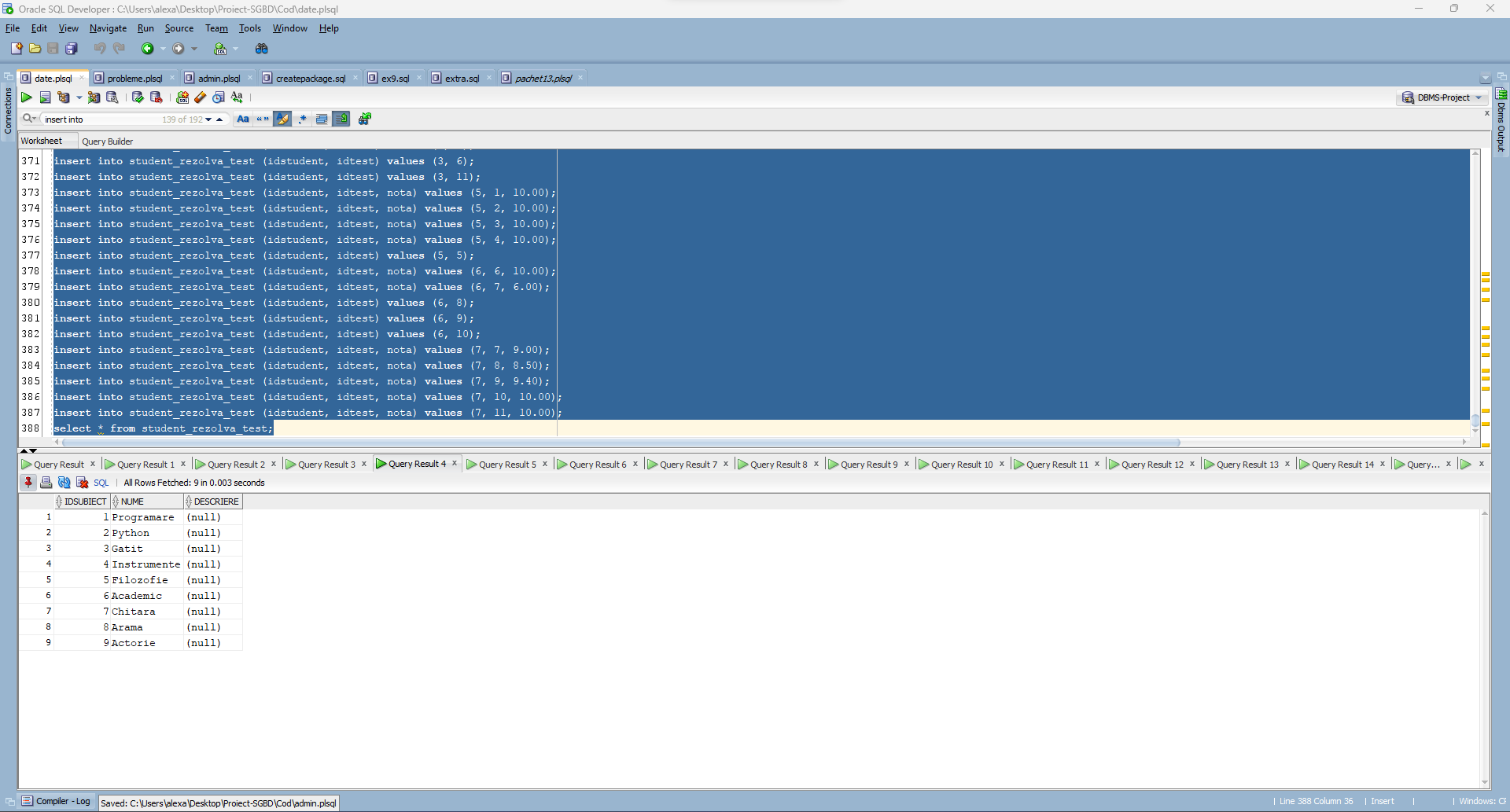
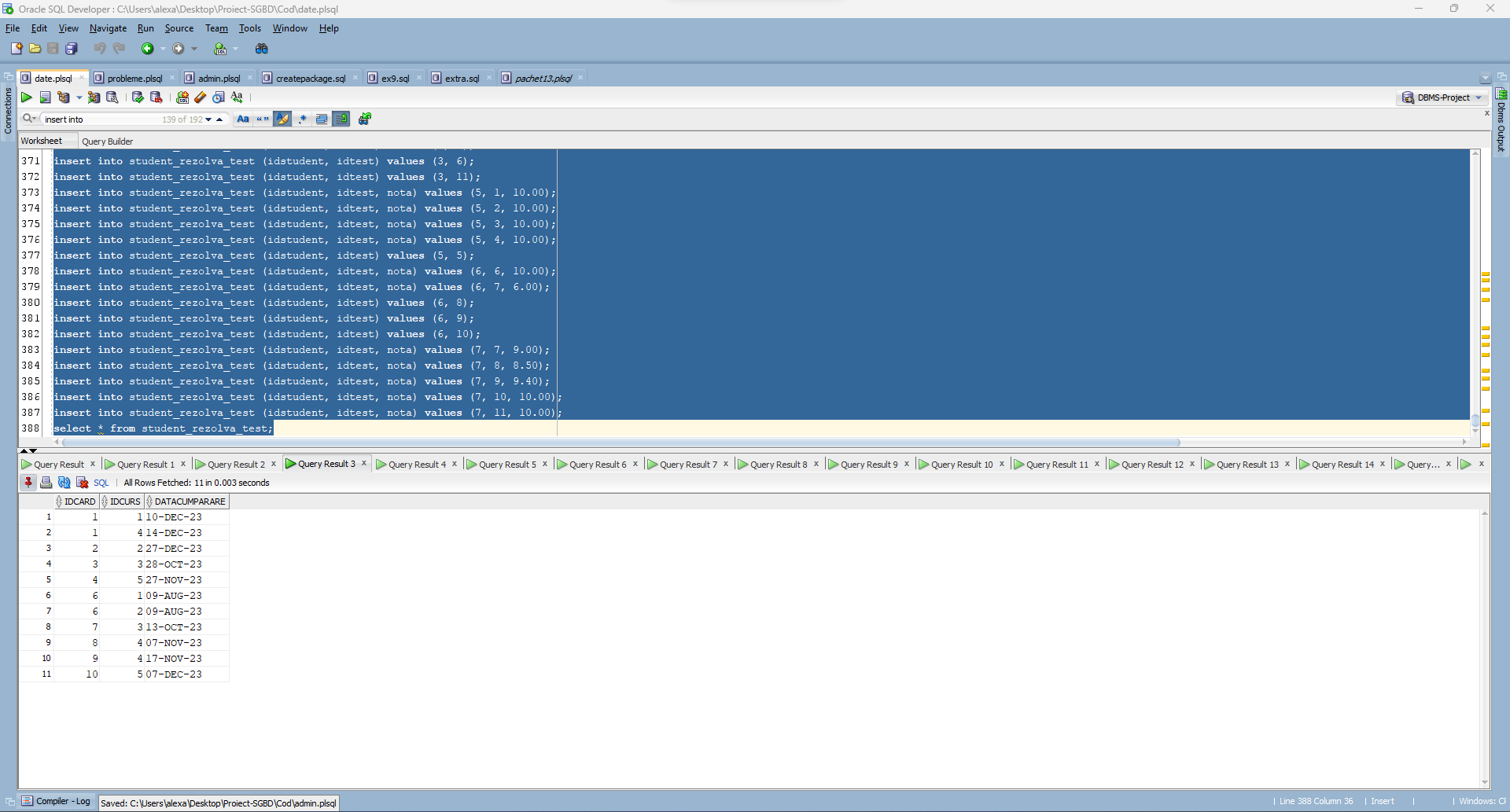
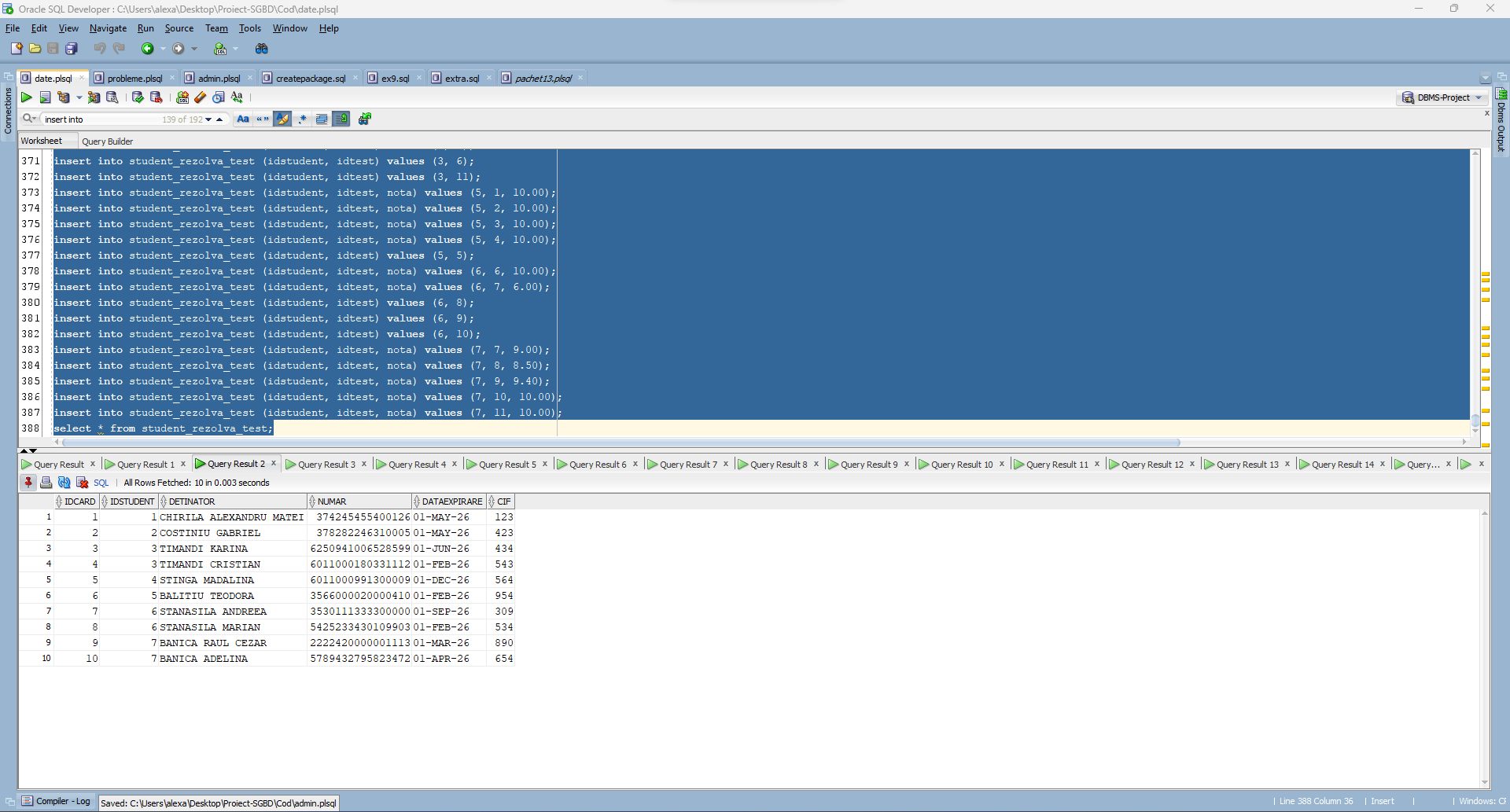
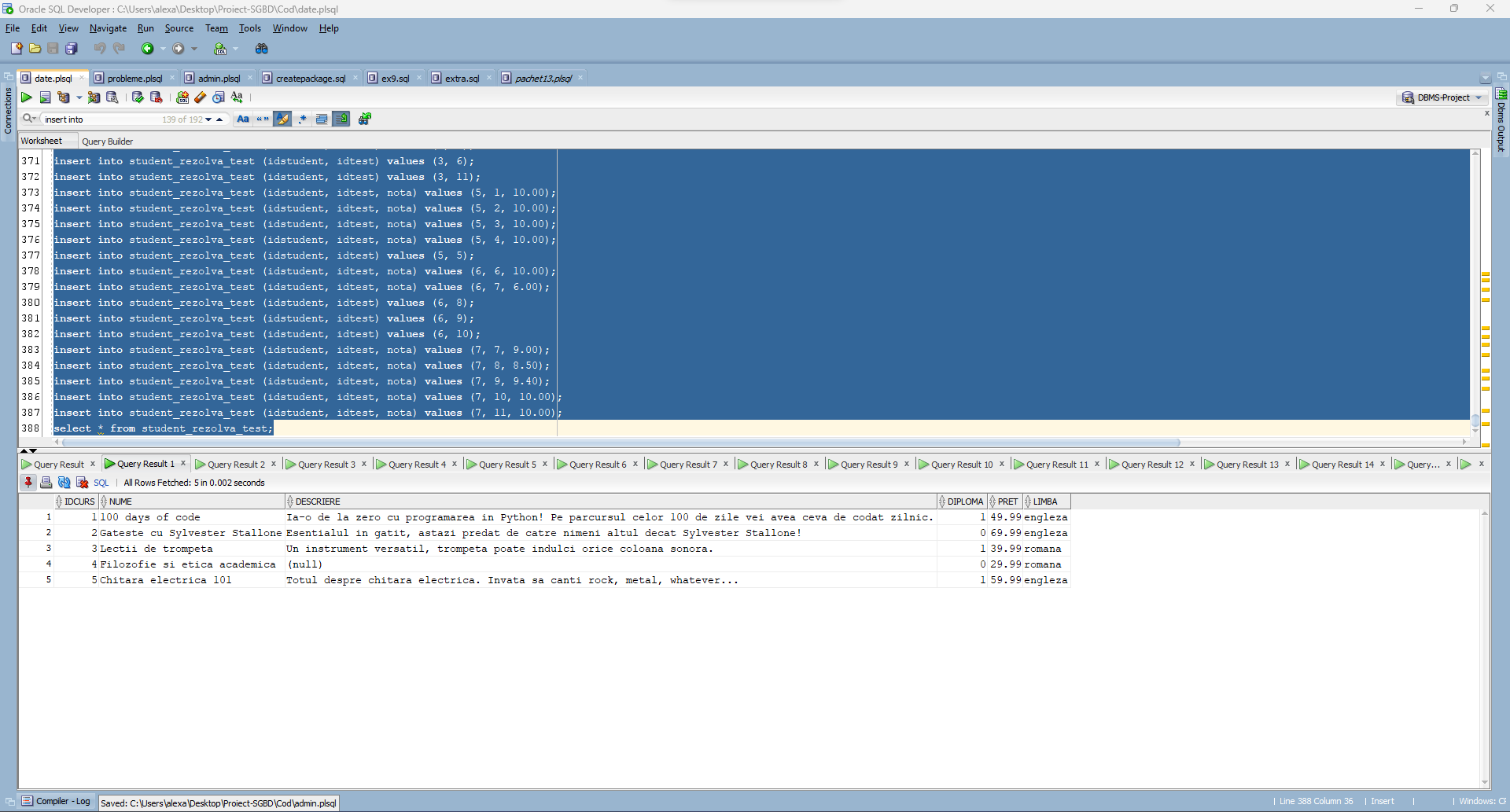
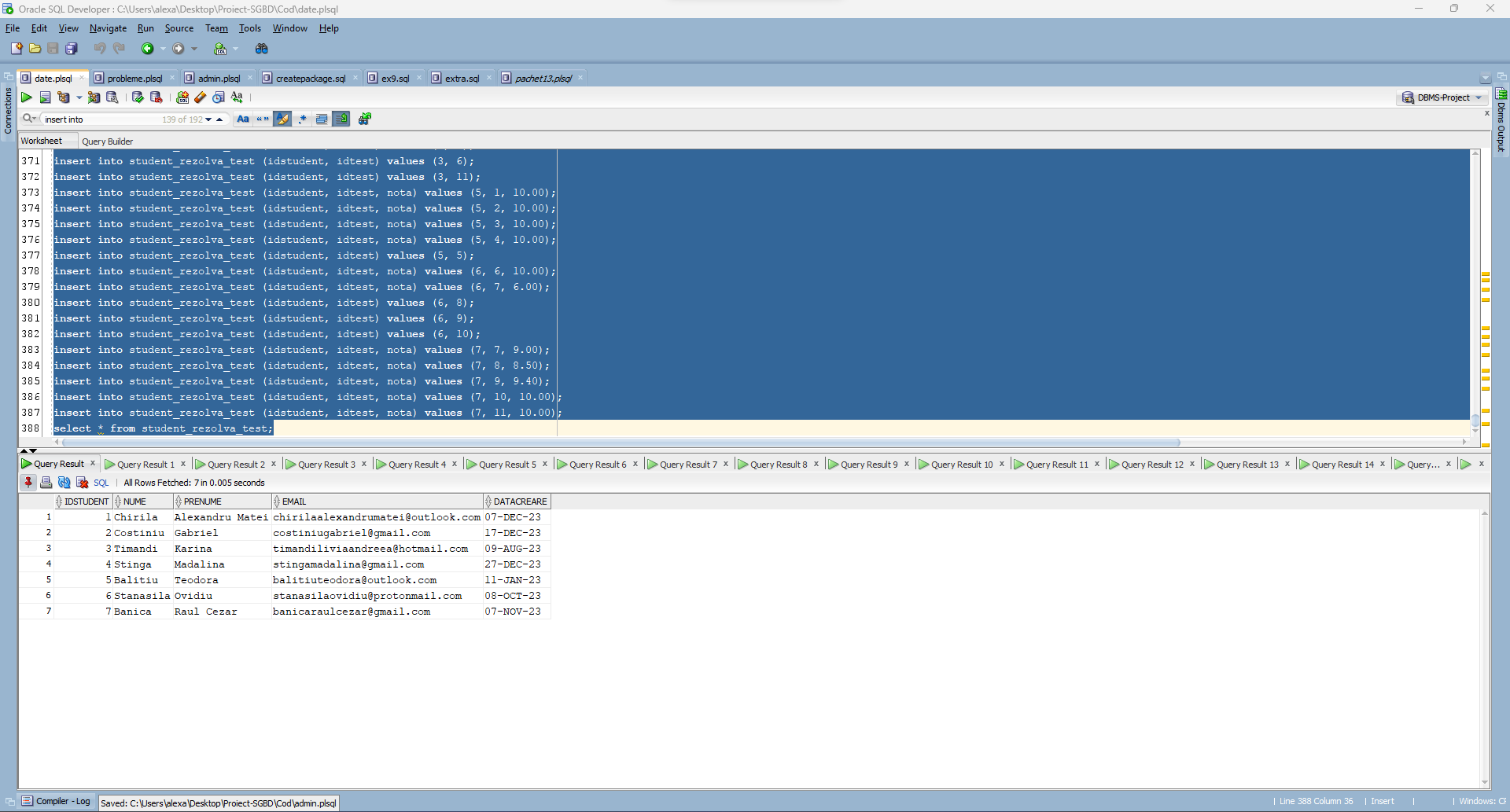
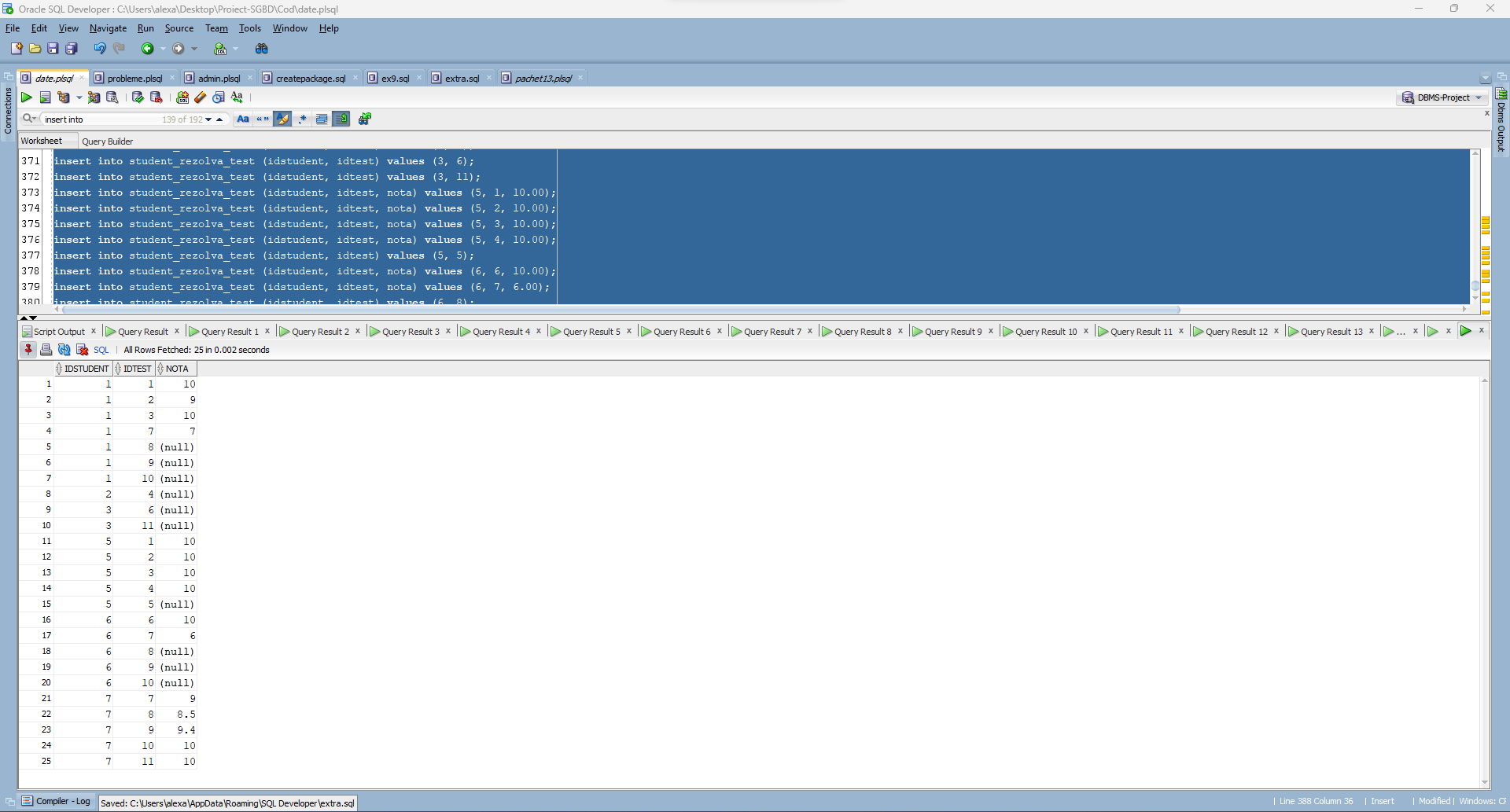
insert into student\_rezolva\_test (idstudent, idtest, nota) values (7, 8, 8.50);

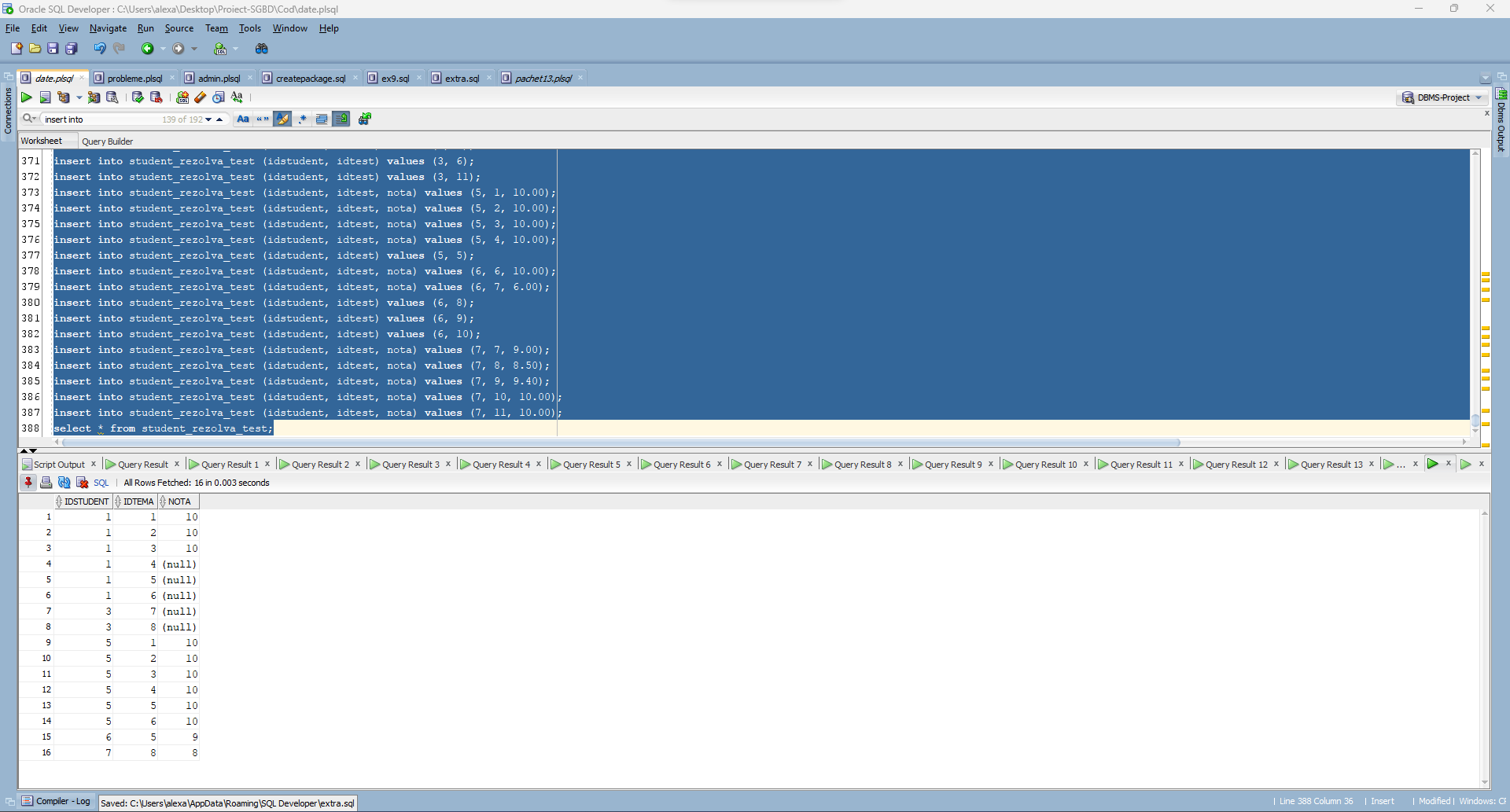
insert into student\_rezolva\_test (idstudent, idtest, nota) values (7, 9, 9.40);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (7, 10, 10.00);

insert into student\_rezolva\_test (idstudent, idtest, nota) values (7, 11, 10.00);

select \* from student\_rezolva\_test;





**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.**

**Pentru un student al carui ID se da de la tastatura sa se afiseze cursurile pe care le-a cumparat pentru fiecare curs cumparat sa se enumere intrebarile din testele aferente fiecarui capitol al cursului, cat si raspunsurile corecte.**

CREATE OR REPLACE PROCEDURE afisare\_sarcini\_de\_lucru (v\_id student.idstudent%type) as

TYPE date\_curs IS RECORD (id curs.idcurs%type, nume curs.nume%type);

TYPE date\_intrebare IS RECORD (enunt intrebare.enunt%type, raspuns intrebare.raspunscorect%type);

TYPE date\_capitol IS RECORD (id capitol.idcapitol%type, titlu capitol.titlu%type);

TYPE t\_curs IS TABLE OF date\_curs INDEX BY PLS\_INTEGER;

TYPE t\_capitol IS TABLE OF date\_capitol;

TYPE t\_intrebare IS TABLE OF date\_intrebare;

TYPE t\_test IS VARRAY(100) of test.IDTEST%type;

nume\_curs t\_curs;

titluri\_capitol t\_capitol := t\_capitol();

intrebari t\_intrebare := t\_intrebare();

teste t\_test := t\_test();

v\_idcurs curs.IDCURS%type;

nume\_s student.nume%type;

datacreare\_s student.datacreare%type;

BEGIN

-- cursurile detinute de studentul dat ca parametru

select nume || ' ' || prenume, datacreare into nume\_s, datacreare\_s from student where idstudent = v\_id;

DBMS\_OUTPUT.PUT\_LINE('Studentul ' || nume\_s || ', care s-a intregistrat la data de ' || datacreare\_s || ' a cumparat urmatoarele cursuri: ');

select cu.idcurs, cu.nume bulk COLLECT INTO nume\_curs

from card ca, CARD\_CUMPARA\_CURS ccc, curs cu

where v\_id = ca.IDSTUDENT and ccc.IDCARD = ca.IDCARD and cu.IDCURS = ccc.IDCURS;

for i in nume\_curs.first..nume\_curs.last LOOP

DBMS\_OUTPUT.PUT\_LINE('Cursul ' || nume\_curs(i).nume || ' are urmatoarele capitole: ');

v\_idcurs := nume\_curs(i).id;

select idcapitol, titlu bulk collect into titluri\_capitol

from capitol

where idcurs = v\_idcurs;

for j in titluri\_capitol.first..titluri\_capitol.last loop

DBMS\_OUTPUT.PUT(' ' || titluri\_capitol(j).titlu || ', care ');

-- ia testele aferente capitolului

select idtest bulk collect into teste

from TEST

where IDCAPITOL = titluri\_capitol(j).id;

if teste.count != 0 then

dbms\_output.PUT\_LINE('are ' || teste.count || ' teste:');

for k in teste.first..teste.last LOOP

-- ia intrebarile din test

dbms\_output.put\_line(' Testul ' || k || ':');

select enunt, raspunscorect bulk collect into intrebari

from INTREBARE

where idtest = teste(k);

for l in intrebari.first..intrebari.last LOOP

-- TODO see how to correct output

DBMS\_OUTPUT.PUT\_line(' Intrebarea ' || l || ': ' || intrebari(l).enunt);

DBMS\_OUTPUT.PUT\_LINE(' Raspunsul corect este: ' || intrebari(l).raspuns);

end loop;

end loop;

ELSE

DBMS\_OUTPUT.PUT\_LINE('nu are teste.');

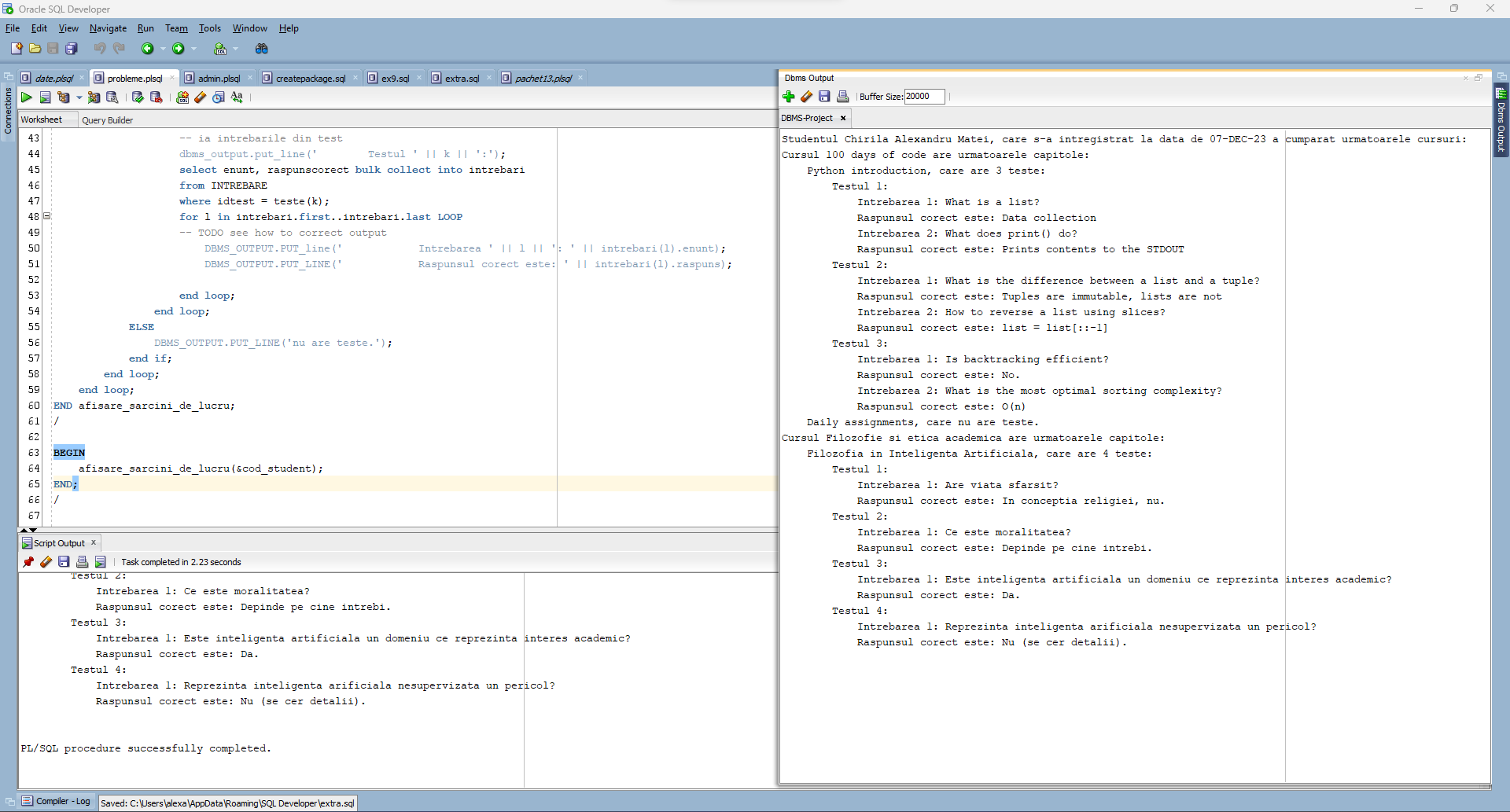
end if;

end loop;

end loop;

END afisare\_sarcini\_de\_lucru;

/



**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.**

**Pentru fiecare student, sa se afiseze cardurile pe care si le-a inregistrat, si ce cursuri a cumparat cu fiecare.**

CREATE OR REPLACE PROCEDURE afisare\_cumparaturi as

id\_s student.idstudent%type;

nume\_s student.nume%type;

nume\_c curs.nume%type;

pret\_c curs.pret%type;

-- cursor explicit

cursor studenti is

select idstudent, nume || ' ' || prenume

from student;

cursor cursuri (id\_c card.idcard%type) IS

select idcurs from CARD\_CUMPARA\_CURS where idcard = id\_c;

BEGIN

open studenti;

loop

fetch studenti into id\_s, nume\_s;

exit when studenti%notfound;

dbms\_output.put\_line('Studentul ' || nume\_s || ' are inregistrate urmatoarele carduri:');

-- cursor implicit

for i in (select idcard, detinator, numar from card where idstudent = id\_s) LOOP

DBMS\_OUTPUT.PUT\_LINE(' ' || i.detinator || ', cu codul ' || i.numar || ', de pe care a cumparat urmatoarele cursuri:');

for j in cursuri(i.idcard) LOOP

select nume, pret into nume\_c, pret\_c

from curs where idcurs = j.idcurs;

DBMS\_OUTput.PUT\_LINE(' ' || nume\_c || ', care costa ' || pret\_c || ' lei.');

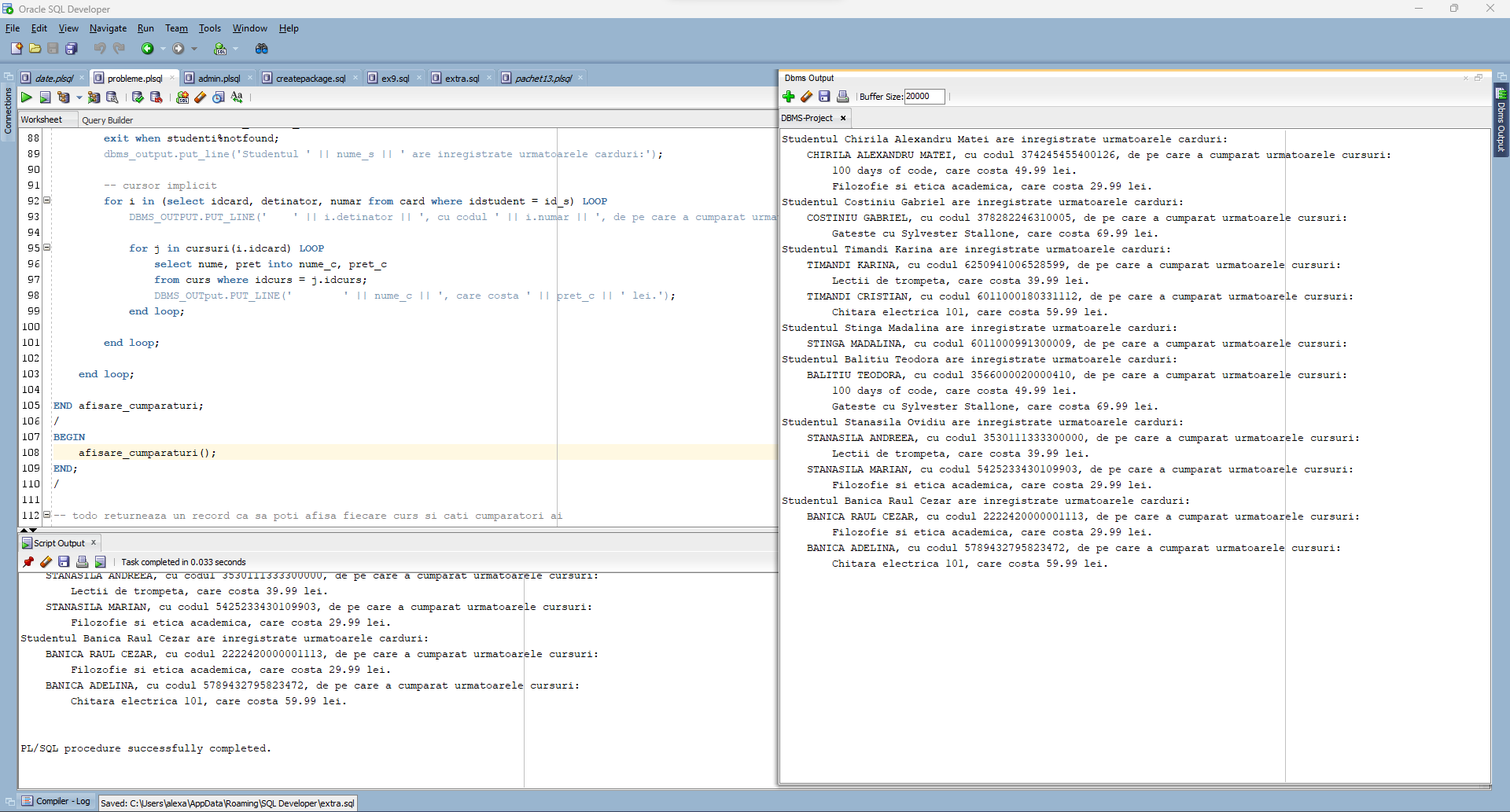
end loop;

end loop;

end loop;

END afisare\_cumparaturi;

/



**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.**

**Pentru un instructor al carui nume se da la tastatura sa se afiseze cate copii ale tuturor cursurilor pe care le preda au fost cumparate, pentru fiecare curs.**

CREATE OR REPLACE FUNCTION numar\_clienti (nume\_instr instructor.nume%type) RETURN number AS

ret\_nrvanzari number;

v\_nrinst int;

v\_idinst int;

nu\_exista\_instructor EXCEPTION;

PRAGMA EXCEPTION\_INIT (nu\_exista\_instructor, -20000);

mai\_multi\_instructori EXCEPTION;

PRAGMA EXCEPTION\_INIT (mai\_multi\_instructori, -20001);

BEGIN

select count(\*) into v\_nrinst from instructor where nume = nume\_instr;

if v\_nrinst > 1 then

raise\_application\_error(-20000, 'Sunt mai multi instructori cu numele dat!');

elsif v\_nrinst < 1 then

raise\_application\_error(-20001, 'Nu exista instructori cu numele dat!');

end if;

dbms\_output.put\_line('Despre instructorul ' || nume\_instr || ' stim urmatoarele: ');

for linie in (select c.nume, count(ccc.idcard) cumparari

from instructor i, instructor\_preda\_curs ipc, card\_cumpara\_curs ccc, curs c

where i.nume = nume\_instr and i.idinstructor = ipc.idinstructor and ipc.idcurs = ccc.idcurs and ipc.idcurs = c.idcurs

group by c.nume) loop

ret\_nrvanzari := ret\_nrvanzari + linie.cumparari;

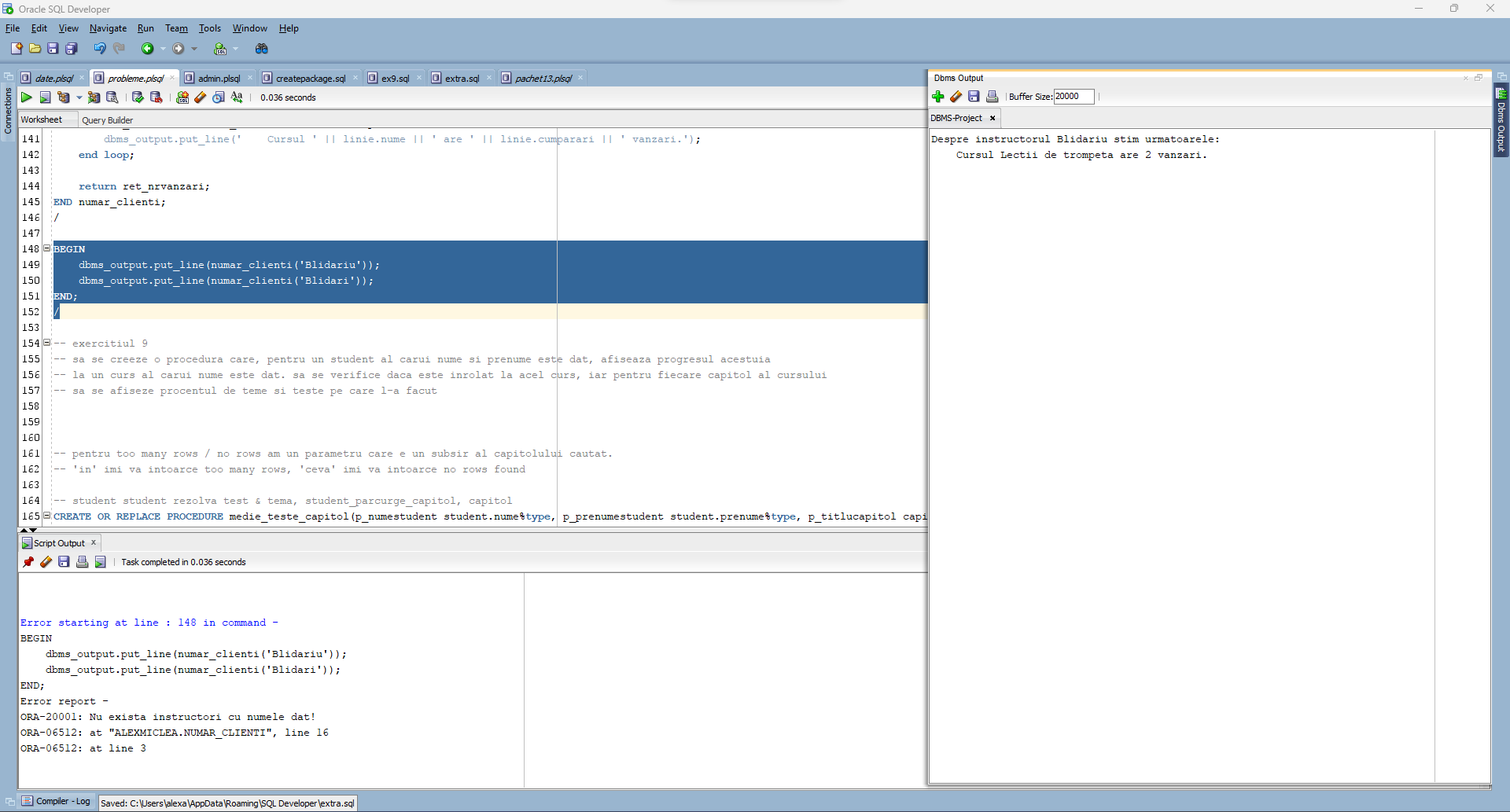
dbms\_output.put\_line(' Cursul ' || linie.nume || ' are ' || linie.cumparari || ' vanzari.');

end loop;

return ret\_nrvanzari;

END numar\_clienti;

/



**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.**

**Sa se creeze o procedura care, pentru un student al carui nume si prenume este dat, afiseaza progresul acestuia la un curs al carui nume este dat. sa se verifice daca este inrolat la acel curs, iar pentru fiecare capitol al cursului sa se afiseze procentul de teme si teste pe care l-a facut.**

CREATE OR REPLACE PROCEDURE medie\_teste\_capitol(p\_numestudent student.nume%type, p\_prenumestudent student.prenume%type, p\_titlucapitol capitol.titlu%type) AS

v\_aux number;

v\_numestudent student.nume%type;

v\_prenumestudent student.prenume%type;

v\_titlucapitol capitol.titlu%type;

v\_medienote number(4,2);

BEGIN

select s.nume, s.prenume, c.titlu, avg(nvl(srtest.nota,0)) medie into v\_numestudent, v\_prenumestudent, v\_titlucapitol, v\_medienote

from student s, capitol c, student\_parcurge\_capitol spc, student\_rezolva\_test srtest, test

where lower(s.nume) = lower(p\_numestudent) and lower(s.prenume) = lower(p\_prenumestudent) and spc.idstudent = s.idstudent and spc.idcapitol = c.idcapitol

and srtest.idstudent = s.idstudent

and test.idtest = srtest.idtest

and test.idcapitol = c.idcapitol

and instr(lower(c.titlu), lower(p\_titlucapitol), 1) > 0

group by s.nume, s.prenume, c.titlu;

dbms\_output.put\_line('Studentul ' || v\_numestudent || ' ' || v\_prenumestudent || ' are media ' || v\_medienote || ' la testele din capitolul ' || v\_titlucapitol);

EXCEPTION

when no\_data\_found then

select count(\*) into v\_aux

from student

where lower(student.nume) = lower(p\_numestudent) and lower(student.prenume) = lower(p\_prenumestudent);

if v\_aux = 0 then

dbms\_output.put\_line('Studentul cu numele si prenumele dat nu exista!');

end if;

select count(distinct c.titlu) into v\_aux

from student s, capitol c, student\_parcurge\_capitol spc, test

where spc.idstudent = s.idstudent and spc.idcapitol = c.idcapitol

and instr(lower(c.titlu), lower(p\_titlucapitol), 1) > 0

and spc.efectuat = 1

and test.idcapitol = c.idcapitol

and lower(s.nume) = lower(p\_numestudent) and lower(s.prenume) = lower(p\_prenumestudent);

if v\_aux = 0 then

dbms\_output.put\_line('Nu s-a putut gasi un capitol cu subsirul dat!');

end if;

when too\_many\_rows then

select count(\*) into v\_aux

from student

where lower(student.nume) = lower(p\_numestudent) and lower(student.prenume) = lower(p\_prenumestudent);

if v\_aux > 1 then

dbms\_output.put\_line('Exista mai multi studenti cu numele si prenumele dat!');

end if;

select count(distinct c.titlu) into v\_aux

from student s, capitol c, student\_parcurge\_capitol spc, test

where spc.idstudent = s.idstudent and spc.idcapitol = c.idcapitol

and instr(lower(c.titlu), lower(p\_titlucapitol), 1) > 0

and spc.efectuat = 1

and test.idcapitol = c.idcapitol

and lower(s.nume) = lower(p\_numestudent) and lower(s.prenume) = lower(p\_prenumestudent);

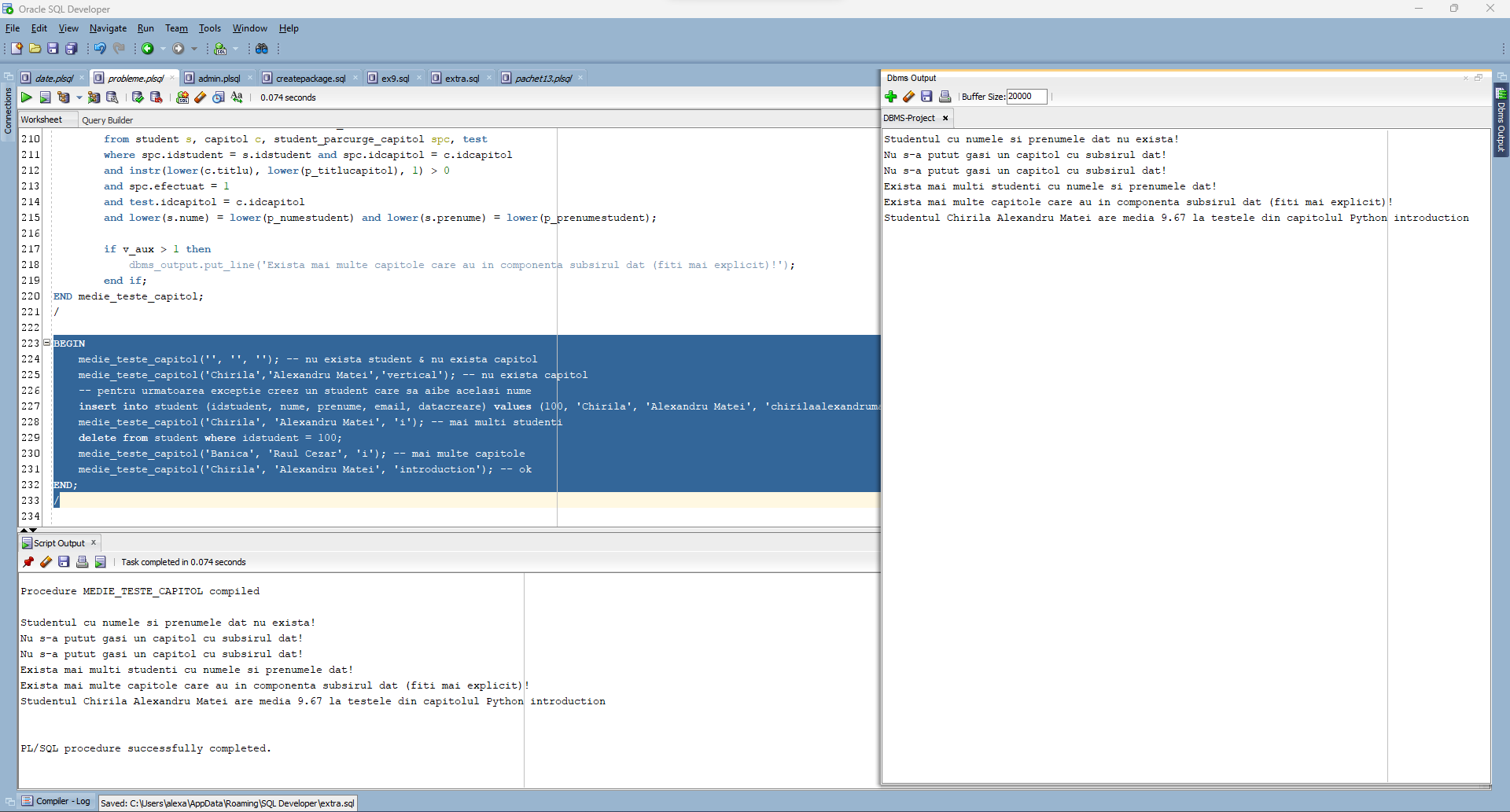
if v\_aux > 1 then

dbms\_output.put\_line('Exista mai multe capitole care au in componenta subsirul dat (fiti mai explicit)!');

end if;

END medie\_teste\_capitol;

/



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

**Implementati un trigger prin care doar administratorul bazei de date poate modifica, insera sau sterge un curs din baza de date.**

create or replace trigger t\_admincurs

before insert or update or delete on curs

declare

begin

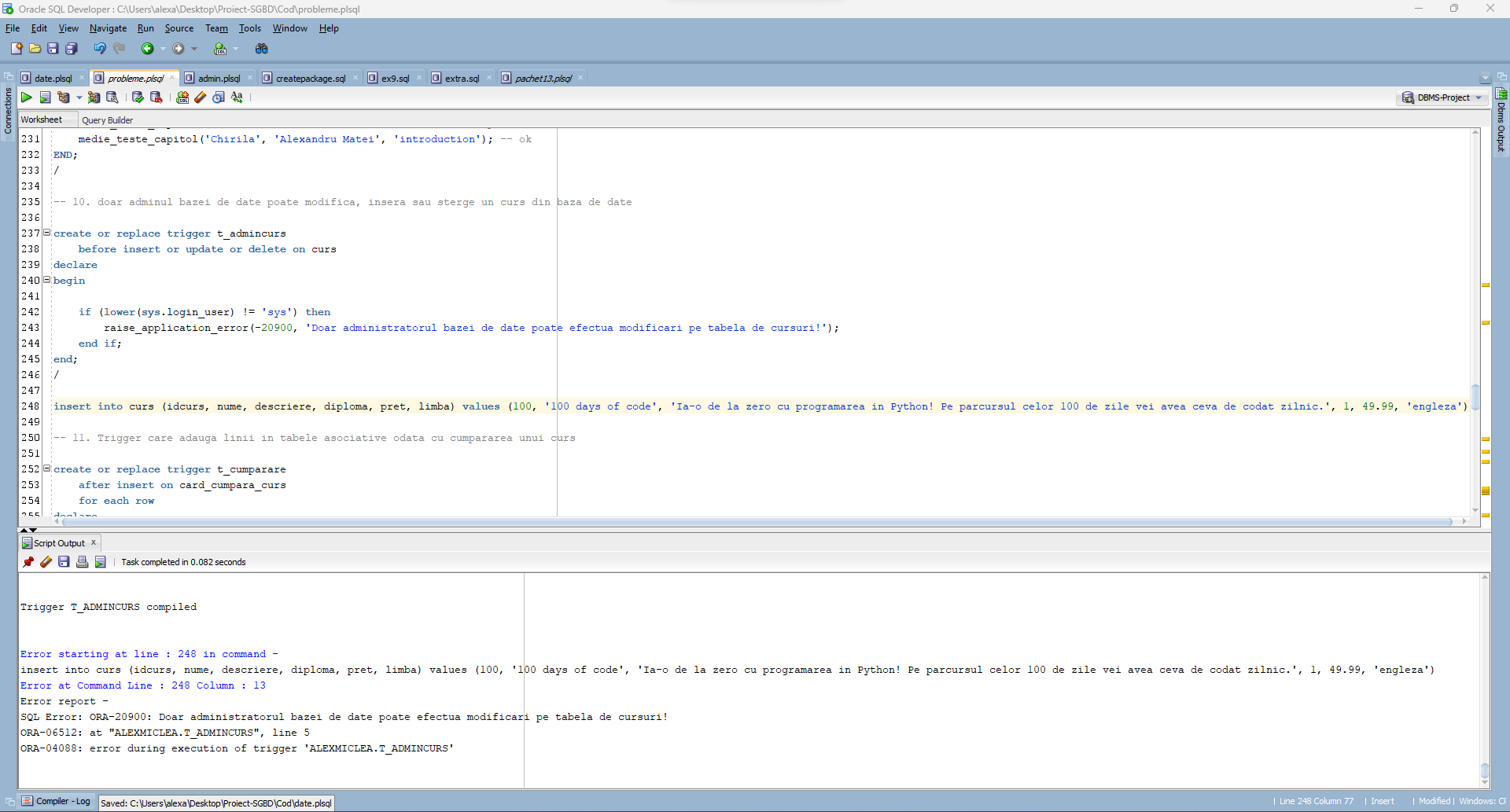
if (lower(sys.login\_user) != 'sys') then

raise\_application\_error(-20900, 'Doar administratorul bazei de date poate efectua modificari pe tabela de cursuri!');

end if;

end;

/



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

**Implementati un trigger care adauga in baza de date linii de interes in tabelele asociative (ex: student rezolva tema) atunci cand un curs este cumparat cu un card.**

create or replace trigger t\_cumparare

after insert on card\_cumpara\_curs

for each row

declare

v\_numarlinii int;

v\_idstudent int;

v\_idcapitol int;

type v\_iduri is varray(10) of int;

v\_idcapitole v\_iduri;

v\_idteste v\_iduri;

v\_idteme v\_iduri;

begin

if inserting then

select c.idstudent into v\_idstudent

from card c

where c.idcard = :NEW.idcard;

select idcapitol bulk collect into v\_idcapitole

from capitol

where idcurs = :NEW.idcurs;

for i in v\_idcapitole.first..v\_idcapitole.last loop

insert into student\_parcurge\_capitol (idstudent, idcapitol, efectuat) values (v\_idstudent, i, 0);

select count(\*) into v\_numarlinii

from test where idcapitol = i;

if (v\_numarlinii > 0) then

select idtest bulk collect into v\_idteste

from test

where idcapitol = i;

end if;

select count(\*) into v\_numarlinii

from tema where idcapitol = i;

if (v\_numarlinii > 0) then

select idtema bulk collect into v\_idteme

from tema

where idcapitol = i;

end if;

end loop;

insert into student\_noteaza\_curs (idstudent, idcurs) values (v\_idstudent, :NEW.idcurs);

for i in v\_idteste.first..v\_idteste.last loop

insert into student\_rezolva\_test (idstudent, idtest) values (v\_idstudent, i);

end loop;

for i in v\_idteme.first..v\_idteme.last loop

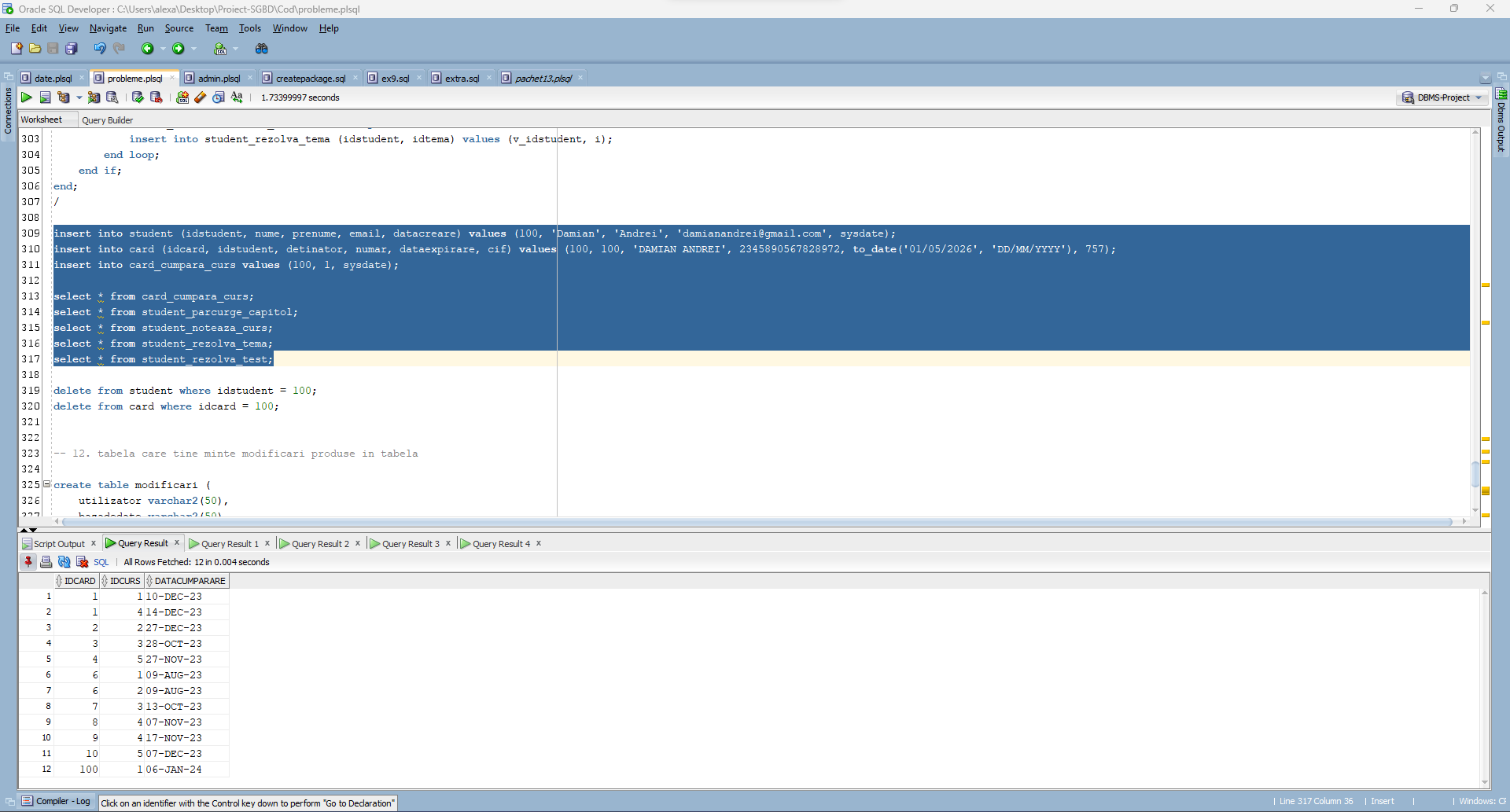
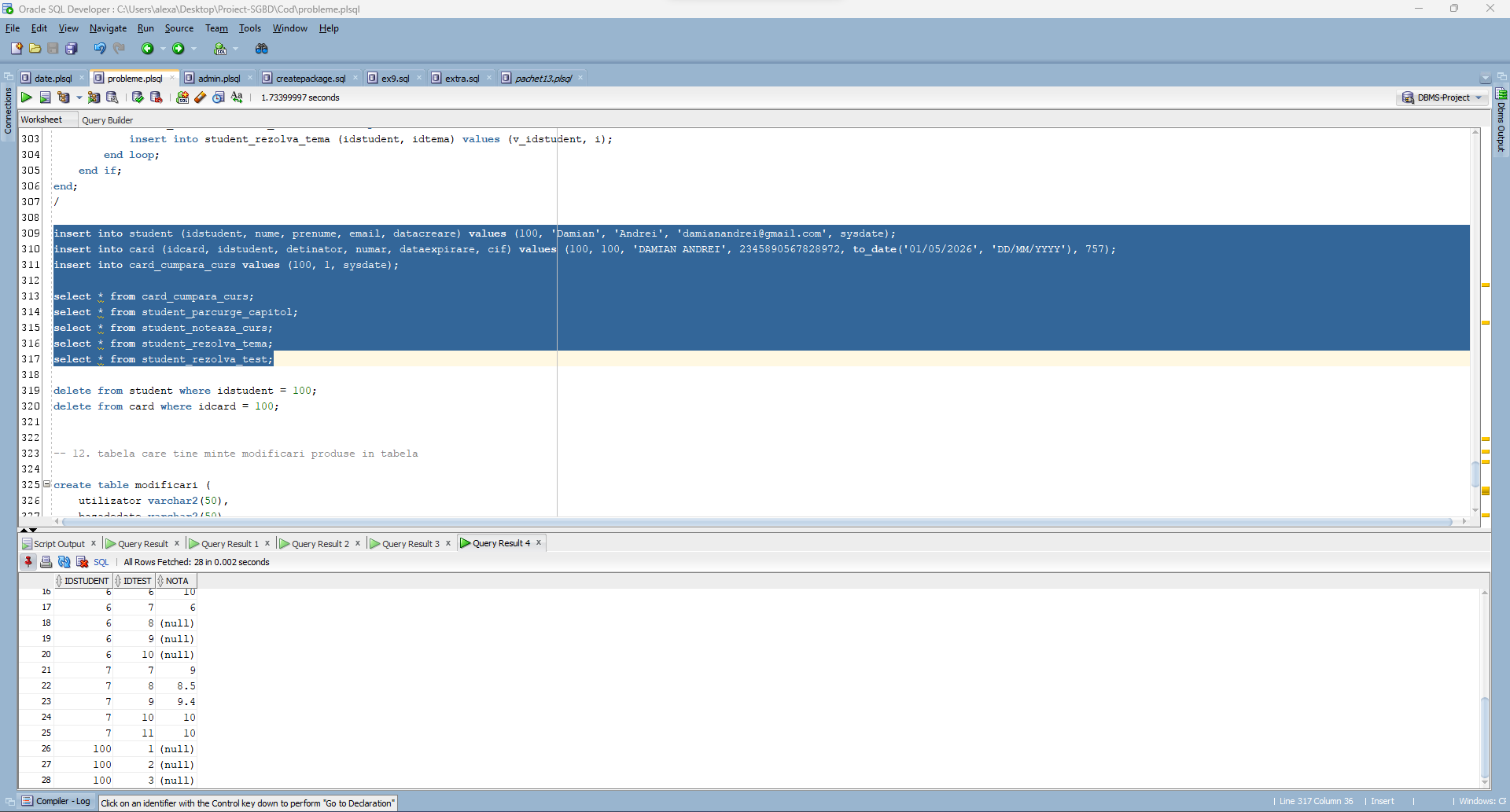
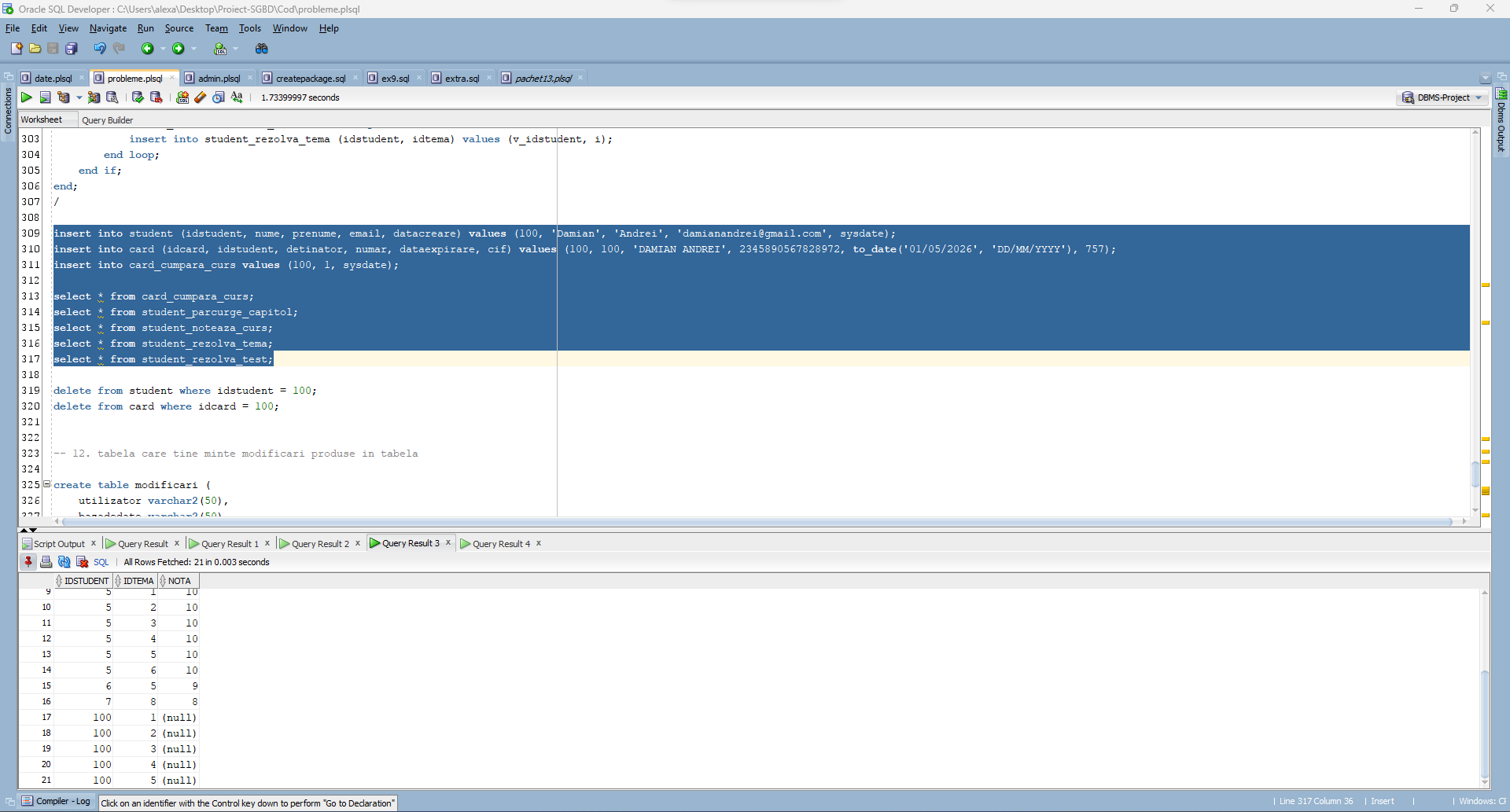
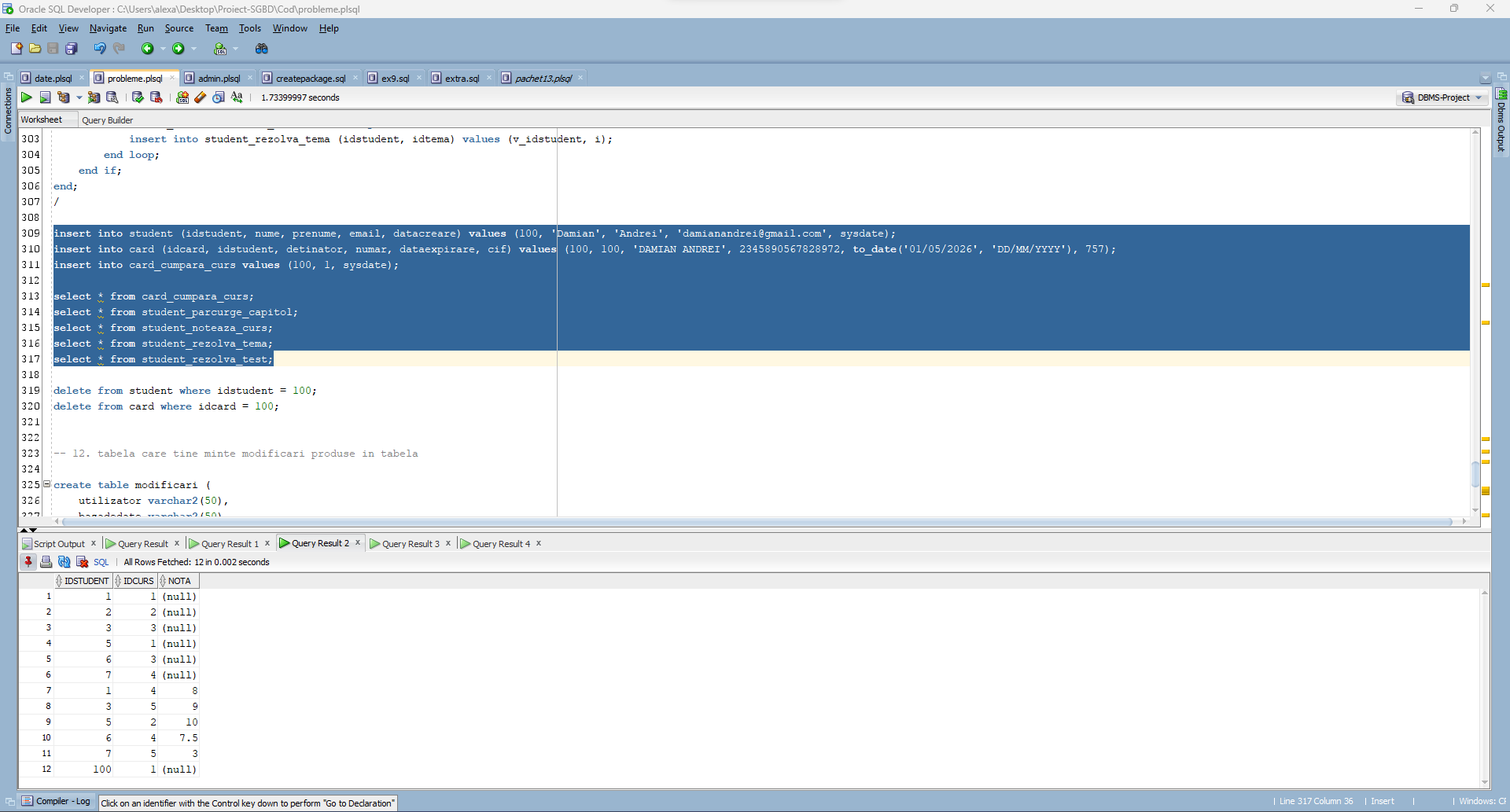
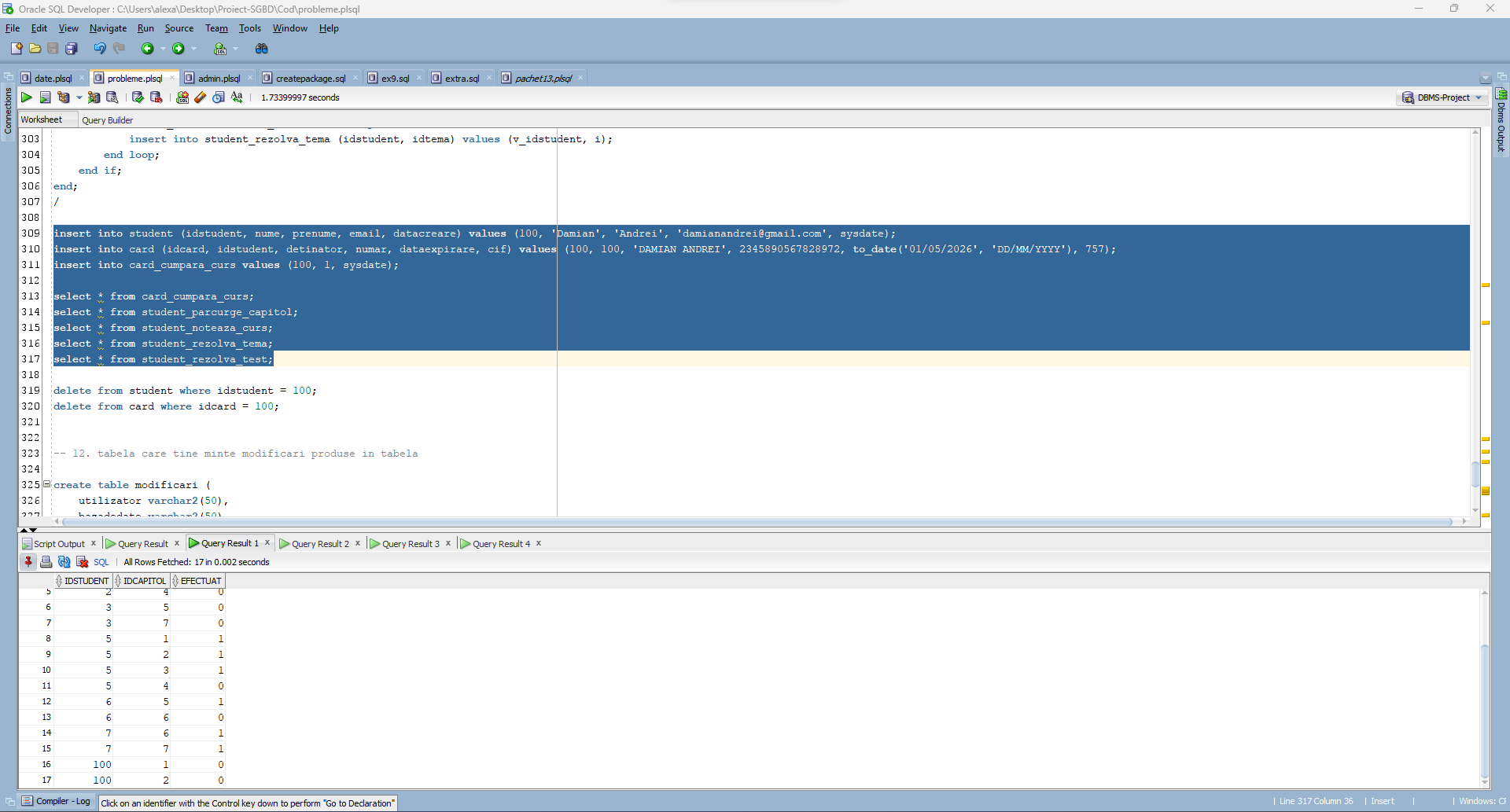
insert into student\_rezolva\_tema (idstudent, idtema) values (v\_idstudent, i);

end loop;

end if;

end;

/



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

**Definiti o tabela in care sa se tina cont de datele despre o modificare adusa bazei de date. Implementati un trigger care sa introduca date in aceasta tabela.**

create table modificari (

utilizator varchar2(50),

bazadedate varchar2(50),

modificare varchar2(50),

numeobiect varchar2(50),

dataefectuare date default sysdate

);

select \* from modificari;

create or replace trigger t\_modificari

after alter or create or drop on schema

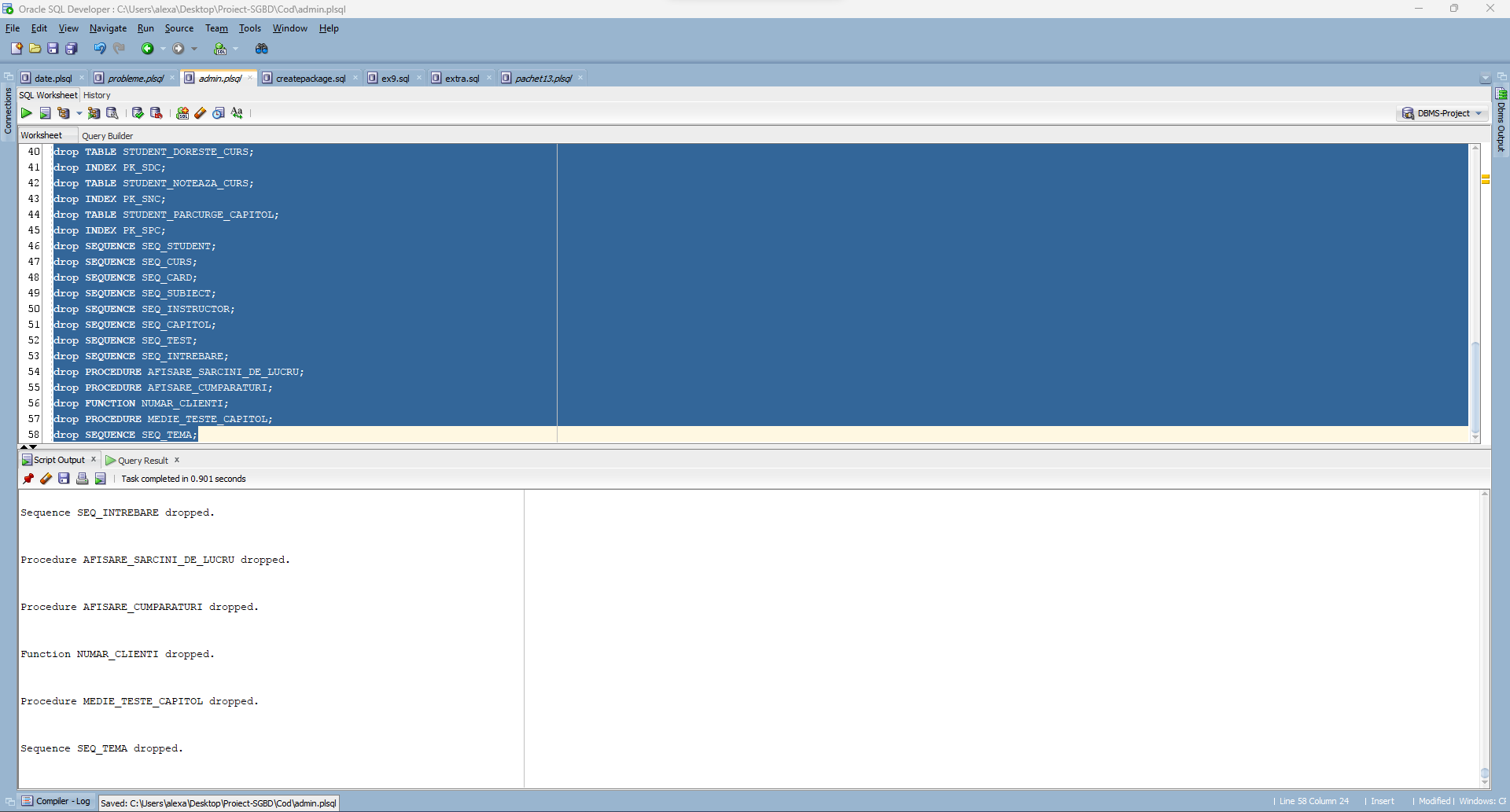
begin

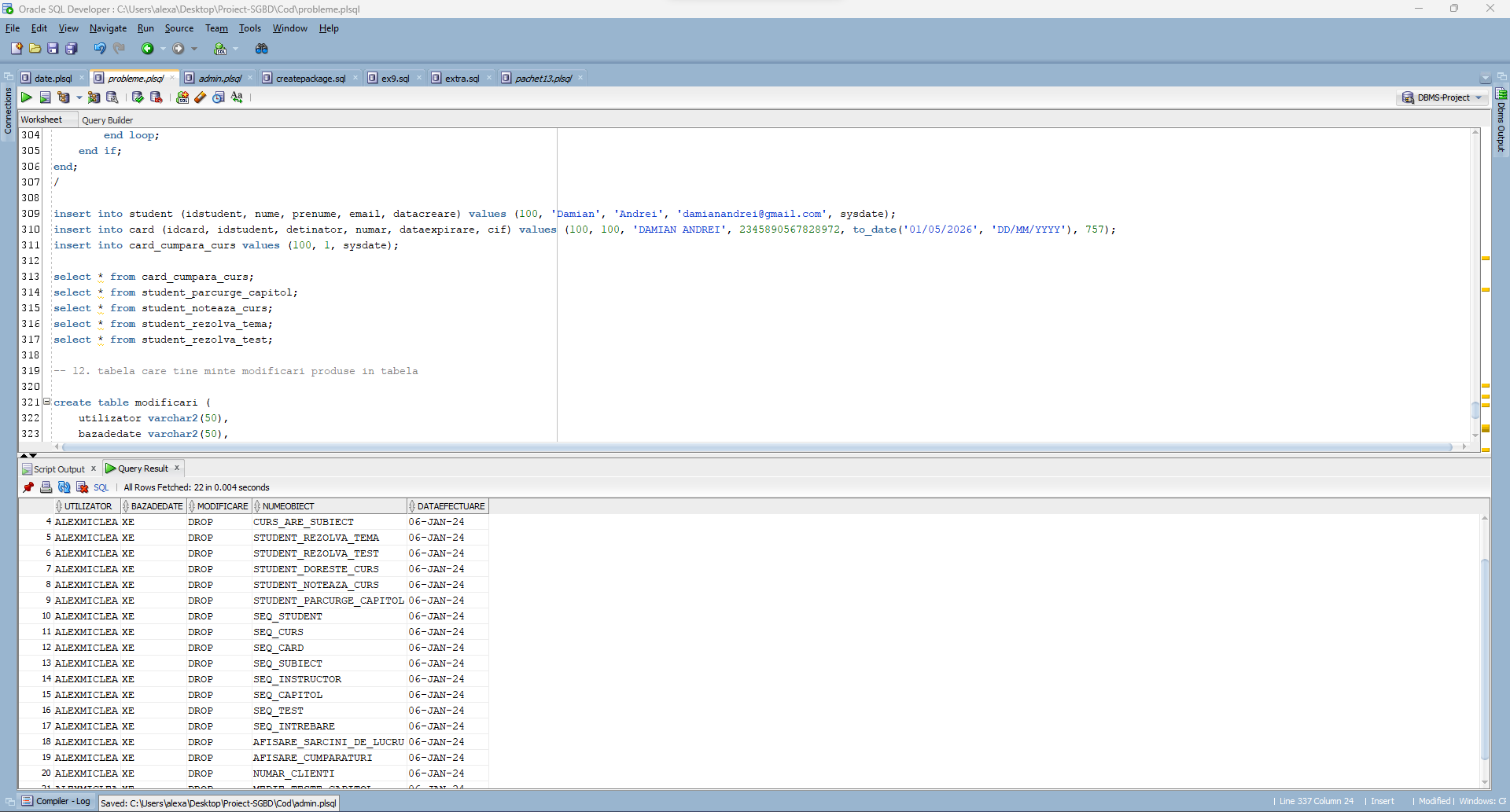
insert into modificari (utilizator, bazadedate, modificare, numeobiect)

values (sys.login\_user, sys.database\_name, sys.sysevent, sys.dictionary\_obj\_name);

end;

/





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

create or replace package tema\_sgbd as

PROCEDURE afisare\_sarcini\_de\_lucru (v\_id student.idstudent%type);

PROCEDURE afisare\_cumparaturi;

FUNCTION numar\_clienti (nume\_instr instructor.nume%type) RETURN number;

PROCEDURE medie\_teste\_capitol(p\_numestudent student.nume%type, p\_prenumestudent student.prenume%type, p\_titlucapitol capitol.titlu%type);

end tema\_sgbd;

create or replace package body tema\_sgbd as

PROCEDURE afisare\_sarcini\_de\_lucru (v\_id student.idstudent%type) as

TYPE date\_curs IS RECORD (id curs.idcurs%type, nume curs.nume%type);

TYPE date\_intrebare IS RECORD (enunt intrebare.enunt%type, raspuns intrebare.raspunscorect%type);

TYPE date\_capitol IS RECORD (id capitol.idcapitol%type, titlu capitol.titlu%type);

TYPE t\_curs IS TABLE OF date\_curs INDEX BY PLS\_INTEGER;

TYPE t\_capitol IS TABLE OF date\_capitol;

TYPE t\_intrebare IS TABLE OF date\_intrebare;

TYPE t\_test IS VARRAY(100) of test.IDTEST%type;

nume\_curs t\_curs;

titluri\_capitol t\_capitol := t\_capitol();

intrebari t\_intrebare := t\_intrebare();

teste t\_test := t\_test();

v\_idcurs curs.IDCURS%type;

nume\_s student.nume%type;

datacreare\_s student.datacreare%type;

BEGIN

-- cursurile detinute de studentul dat ca parametru

select nume || ' ' || prenume, datacreare into nume\_s, datacreare\_s from student where idstudent = v\_id;

DBMS\_OUTPUT.PUT\_LINE('Studentul ' || nume\_s || ', care s-a intregistrat la data de ' || datacreare\_s || ' a cumparat urmatoarele cursuri: ');

select cu.idcurs, cu.nume bulk COLLECT INTO nume\_curs

from card ca, CARD\_CUMPARA\_CURS ccc, curs cu

where v\_id = ca.IDSTUDENT and ccc.IDCARD = ca.IDCARD and cu.IDCURS = ccc.IDCURS;

for i in nume\_curs.first..nume\_curs.last LOOP

DBMS\_OUTPUT.PUT\_LINE('Cursul ' || nume\_curs(i).nume || ' are urmatoarele capitole: ');

v\_idcurs := nume\_curs(i).id;

select idcapitol, titlu bulk collect into titluri\_capitol

from capitol

where idcurs = v\_idcurs;

for j in titluri\_capitol.first..titluri\_capitol.last loop

DBMS\_OUTPUT.PUT(' ' || titluri\_capitol(j).titlu || ', care ');

-- ia testele aferente capitolului

select idtest bulk collect into teste

from TEST

where IDCAPITOL = titluri\_capitol(j).id;

if teste.count != 0 then

dbms\_output.PUT\_LINE('are ' || teste.count || ' teste:');

for k in teste.first..teste.last LOOP

-- ia intrebarile din test

dbms\_output.put\_line(' Testul ' || k || ':');

select enunt, raspunscorect bulk collect into intrebari

from INTREBARE

where idtest = teste(k);

for l in intrebari.first..intrebari.last LOOP

-- TODO see how to correct output

DBMS\_OUTPUT.PUT\_line(' Intrebarea ' || l || ': ' || intrebari(l).enunt);

DBMS\_OUTPUT.PUT\_LINE(' Raspunsul corect este: ' || intrebari(l).raspuns);

end loop;

end loop;

ELSE

DBMS\_OUTPUT.PUT\_LINE('nu are teste.');

end if;

end loop;

end loop;

END afisare\_sarcini\_de\_lucru;

PROCEDURE afisare\_cumparaturi as

id\_s student.idstudent%type;

nume\_s student.nume%type;

nume\_c curs.nume%type;

pret\_c curs.pret%type;

-- cursor explicit

cursor studenti is

select idstudent, nume || ' ' || prenume

from student;

cursor cursuri (id\_c card.idcard%type) IS

select idcurs from CARD\_CUMPARA\_CURS where idcard = id\_c;

BEGIN

open studenti;

loop

fetch studenti into id\_s, nume\_s;

exit when studenti%notfound;

dbms\_output.put\_line('Studentul ' || nume\_s || ' are inregistrate urmatoarele carduri:');

-- cursor implicit

for i in (select idcard, detinator, numar from card where idstudent = id\_s) LOOP

DBMS\_OUTPUT.PUT\_LINE(' ' || i.detinator || ', cu codul ' || i.numar || ', de pe care a cumparat urmatoarele cursuri:');

for j in cursuri(i.idcard) LOOP

select nume, pret into nume\_c, pret\_c

from curs where idcurs = j.idcurs;

DBMS\_OUTput.PUT\_LINE(' ' || nume\_c || ', care costa ' || pret\_c || ' lei.');

end loop;

end loop;

end loop;

END afisare\_cumparaturi;

FUNCTION numar\_clienti (nume\_instr instructor.nume%type) RETURN number AS

ret\_nrvanzari number;

v\_nrinst int;

v\_idinst int;

nu\_exista\_instructor EXCEPTION;

PRAGMA EXCEPTION\_INIT (nu\_exista\_instructor, -20000);

mai\_multi\_instructori EXCEPTION;

PRAGMA EXCEPTION\_INIT (mai\_multi\_instructori, -20001);

BEGIN

select count(\*) into v\_nrinst from instructor where nume = nume\_instr;

if v\_nrinst > 1 then

raise\_application\_error(-20000, 'Sunt mai multi instructori cu numele dat!');

elsif v\_nrinst < 1 then

raise\_application\_error(-20001, 'Nu exista instructori cu numele dat!');

end if;

dbms\_output.put\_line('Despre instructorul ' || nume\_instr || ' stim urmatoarele: ');

for linie in (select c.nume, count(ccc.idcard) cumparari

from instructor i, instructor\_preda\_curs ipc, card\_cumpara\_curs ccc, curs c

where i.nume = nume\_instr and i.idinstructor = ipc.idinstructor and ipc.idcurs = ccc.idcurs and ipc.idcurs = c.idcurs

group by c.nume) loop

ret\_nrvanzari := ret\_nrvanzari + linie.cumparari;

dbms\_output.put\_line(' Cursul ' || linie.nume || ' are ' || linie.cumparari || ' vanzari.');

end loop;

return ret\_nrvanzari;

END numar\_clienti;

PROCEDURE medie\_teste\_capitol(p\_numestudent student.nume%type, p\_prenumestudent student.prenume%type, p\_titlucapitol capitol.titlu%type) AS

v\_aux number;

v\_numestudent student.nume%type;

v\_prenumestudent student.prenume%type;

v\_titlucapitol capitol.titlu%type;

v\_medienote number(4,2);

BEGIN

select s.nume, s.prenume, c.titlu, avg(nvl(srtest.nota,0)) medie into v\_numestudent, v\_prenumestudent, v\_titlucapitol, v\_medienote

from student s, capitol c, student\_parcurge\_capitol spc, student\_rezolva\_test srtest, test

where lower(s.nume) = lower(p\_numestudent) and lower(s.prenume) = lower(p\_prenumestudent) and spc.idstudent = s.idstudent and spc.idcapitol = c.idcapitol

and srtest.idstudent = s.idstudent

and test.idtest = srtest.idtest

and test.idcapitol = c.idcapitol

and instr(lower(c.titlu), lower(p\_titlucapitol), 1) > 0

group by s.nume, s.prenume, c.titlu;

dbms\_output.put\_line('Studentul ' || v\_numestudent || ' ' || v\_prenumestudent || ' are media ' || v\_medienote || ' la testele din capitolul ' || v\_titlucapitol);

EXCEPTION

when no\_data\_found then

select count(\*) into v\_aux

from student

where lower(student.nume) = lower(p\_numestudent) and lower(student.prenume) = lower(p\_prenumestudent);

if v\_aux = 0 then

dbms\_output.put\_line('Studentul cu numele si prenumele dat nu exista!');

end if;

select count(distinct c.titlu) into v\_aux

from student s, capitol c, student\_parcurge\_capitol spc, test

where spc.idstudent = s.idstudent and spc.idcapitol = c.idcapitol

and instr(lower(c.titlu), lower(p\_titlucapitol), 1) > 0

and spc.efectuat = 1

and test.idcapitol = c.idcapitol

and lower(s.nume) = lower(p\_numestudent) and lower(s.prenume) = lower(p\_prenumestudent);

if v\_aux = 0 then

dbms\_output.put\_line('Nu s-a putut gasi un capitol cu subsirul dat!');

end if;

when too\_many\_rows then

select count(\*) into v\_aux

from student

where lower(student.nume) = lower(p\_numestudent) and lower(student.prenume) = lower(p\_prenumestudent);

if v\_aux > 1 then

dbms\_output.put\_line('Exista mai multi studenti cu numele si prenumele dat!');

end if;

select count(distinct c.titlu) into v\_aux

from student s, capitol c, student\_parcurge\_capitol spc, test

where spc.idstudent = s.idstudent and spc.idcapitol = c.idcapitol

and instr(lower(c.titlu), lower(p\_titlucapitol), 1) > 0

and spc.efectuat = 1

and test.idcapitol = c.idcapitol

and lower(s.nume) = lower(p\_numestudent) and lower(s.prenume) = lower(p\_prenumestudent);

if v\_aux > 1 then

dbms\_output.put\_line('Exista mai multe capitole care au in componenta subsirul dat (fiti mai explicit)!');

end if;

END medie\_teste\_capitol;

END tema\_sgbd;

/

