

Gestiunea unei Universități

Broscoteanu Daria-Mihaela
Grupa 243

I. Prezența pe scurt baza de date (utilitatea ei)

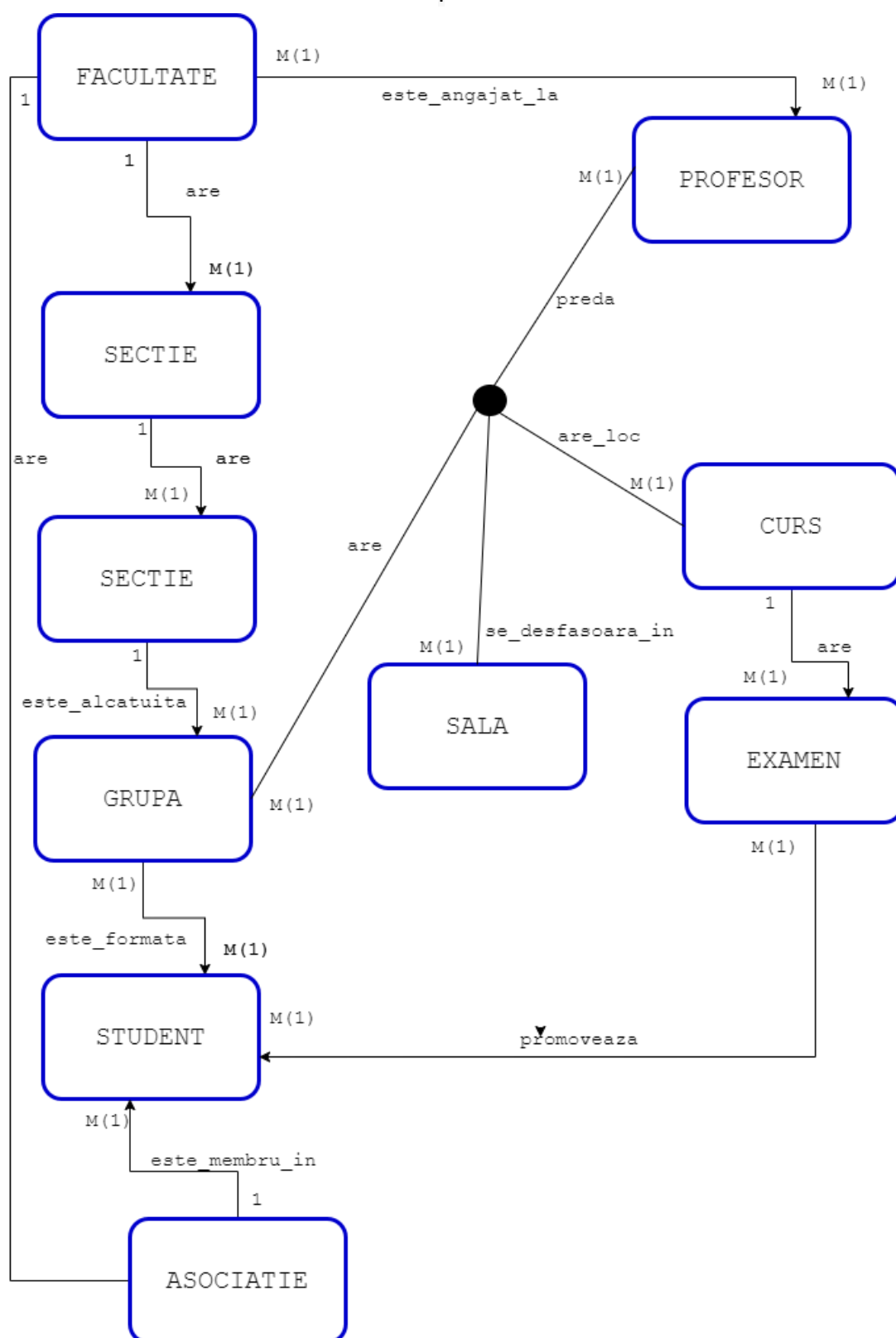
Baza de date conține informații cu privire la facultățile unei universități, secțiile și grupele acestora, cât și profesorii care predau în cadrul universității, ce cursuri și căror studenți le predau aceștia, dacă sunt voluntari în cadrul unei asociații sau nu, cât și notele obținute de elevi în cadrul unui curs care se va desfășura într-o anumită sală.

Scopul creării acestei baze de date este de a putea ține evidența studenților din cadrul universității, a situației lor școlare, cât și cursurilor predate în cadrul fiecărei facultăți, cât și a profesorului care susține cursul.

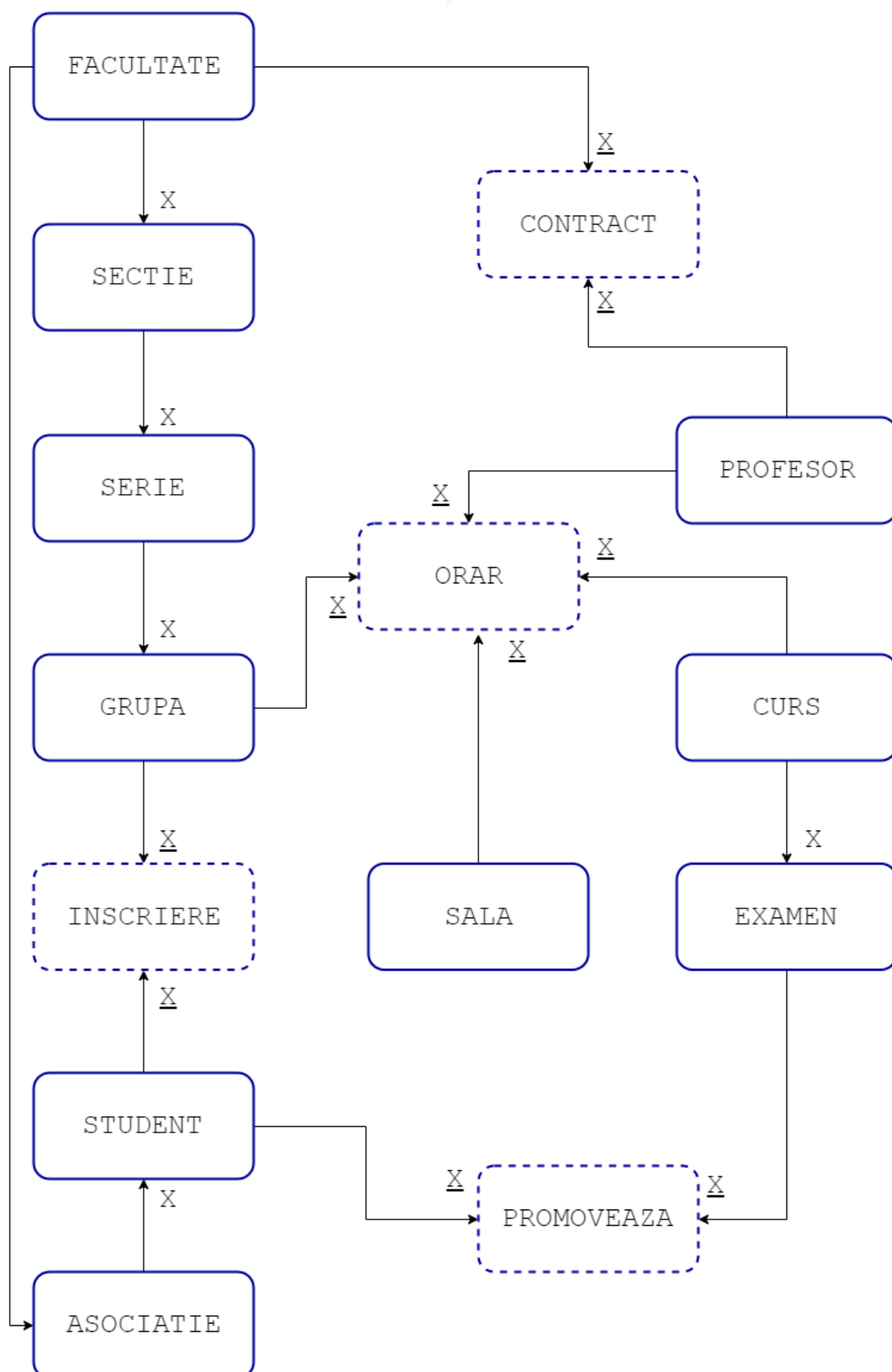
Fiecare student este repartizat la o secție, într-o serie și grupă, participând la anumite cursuri predate de profesori care au posibilitatea de a avea contracte cu mai multe facultăți din cadrul universității. Studenții pot lua parte la acțiuni de voluntariat în cadrul unei asociații studențești.

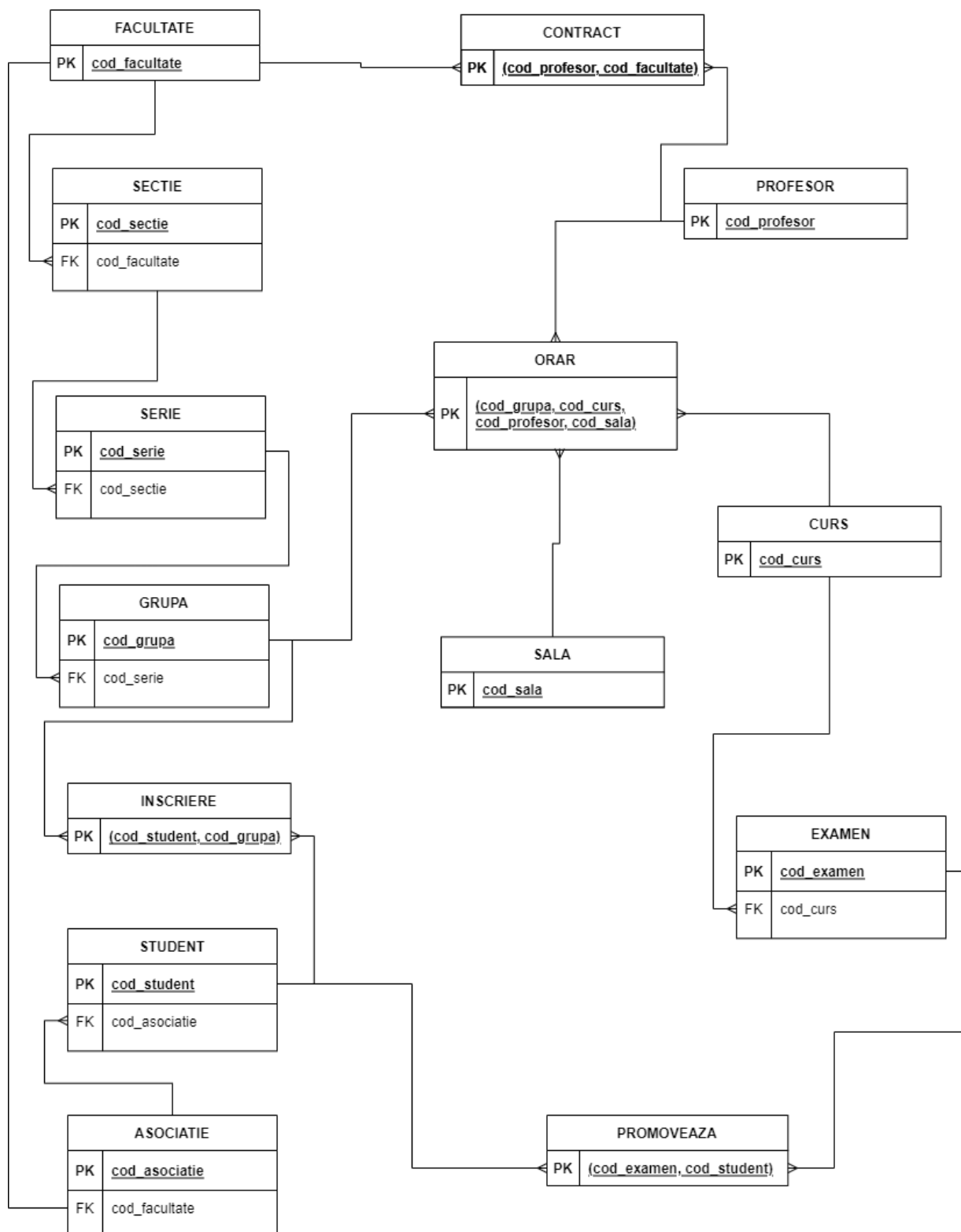
Cursurile la care iau parte studenții sunt susținute la nivel de grupă, în diferite săli ale facultății, de către profesori diferiți. Un profesor are contracte cu facultățile la care predă, acesta putând să predea diferite cursuri în cadrul unei facultăți. Un examen poate lua mai multe forme: proiect, examen scris sau interviu. Forma acestuia diferă în funcție de materia la care este susținut. Un student promovează un examen în cazul în care obține o notă mai mare sau egală cu 5.

2. Diagrama Entitate-Relație(ERD)



3. Diagrama Conceptuală





4. și 5. Implementați în Oracle diagrama conceptuală și inserări de date

```

-----FACULTATE-----
CREATE TABLE FACULTATE
(
    cod_facultate NUMBER(5) CONSTRAINT PKEY_FACULTATE PRIMARY KEY,
    denumire VARCHAR(100) CONSTRAINT denumire_facultate NOT NULL,
    adresa VARCHAR(100) CONSTRAINT adresa_facultate NOT NULL,
    telefon VARCHAR(20) CONSTRAINT telefon_facultate NOT NULL,
    mail VARCHAR(50) UNIQUE,
    fax VARCHAR(50),
    cod_postal VARCHAR(6)
);

SELECT *
FROM facultate;

INSERT INTO FACULTATE
VALUES(1,'Facultatea de Matematica si Informatica','Str. Academiei nr. 14',
'+4021-314.28.63', 'secretariat@fmi.unibuc.ro','+4021-314.28.63','010014');

INSERT INTO FACULTATE
VALUES(2,'Facultatea de Drept','Bd. M. Kogălniceanu, nr. 36-46', '+4021-312.49.48',
'informatii.drept@drept.unibuc.ro','+4021-312.49.48','050107');

INSERT INTO FACULTATE
VALUES(3,'Facultatea de Geografie','Bd. Nicolae Bălcescu Nr. 1', '+4021-305.38.10',
'secretariat@geo.unibuc.ro','+4021-305.38.10','010041');

INSERT INTO FACULTATE
VALUES(4,'Facultatea de Istorie','Bd. Regina Elisabeta nr. 4-12', '+4021-314.35.89',
'secretariat@istorie.unibuc.ro','+4021-314.35.89','030018');

INSERT INTO FACULTATE
VALUES(5,'Facultatea de Litere','Str. Edgar Quinet, nr. 5-7', '+4021-313.43.36',
'gabriela.dena@litere.unibuc.ro','+4021-313.43.36','010017');

COMMIT;

```

COD_FACULTATE	DENUMIRE	ADRESA	TELEFON	MAIL	FAX	COD_POSTAL
1	Facultatea de Matematica si Informatica	Str. Academiei nr. 14	+4021-314.28.63	secretariat@fmi.unibuc.ro	+4021-314.28.63	010014
2	Facultatea de Drept	Bd. M. Kogălniceanu, nr. 36-46	+4021-312.49.48	informatii.drept@drept.unibuc.ro	+4021-312.49.48	050107
3	Facultatea de Geografie	Bd. Nicolae Bălcescu Nr. 1	+4021-305.38.10	secretariat@geo.unibuc.ro	+4021-305.38.10	010041
4	Facultatea de Istorie	Bd. Regina Elisabeta nr. 4-12	+4021-314.35.89	secretariat@istorie.unibuc.ro	+4021-314.35.89	030018
5	Facultatea de Litere	Str. Edgar Quinet, nr. 5-7	+4021-313.43.36	gabriela.dena@litere.unibuc.ro	+4021-313.43.36	010017

```
-----ASOCIATIE-----  
-----  
  
CREATE TABLE ASOCIATIE(cod_asociatie NUMBER(5) CONSTRAINT PKEY_ASOCIATIE PRIMARY KEY,  
                        denumire VARCHAR(100) CONSTRAINT denumire_asociatie NOT NULL,  
                        data_infiintarii DATE CONSTRAINT data_infiintarii_const NOT  
NULL,  
                        cod_facultate NUMBER(5),  
                        CONSTRAINT fk_asoc FOREIGN KEY(cod_facultate) REFERENCES  
FACULTATE(cod_facultate)  
);  
  
Select *  
from asociatie;  
  
INSERT INTO ASOCIATIE  
VALUES(10, 'ASMI', TO_DATE('12-04-2009', 'DD-MM-YYYY'), 1);  
  
INSERT INTO ASOCIATIE  
VALUES(11, 'ASD', TO_DATE('17-06-2011', 'DD-MM-YYYY'), 2);  
  
INSERT INTO ASOCIATIE  
VALUES(12, 'ASG', TO_DATE('02-08-2012', 'DD-MM-YYYY'), 3);  
  
INSERT INTO ASOCIATIE  
VALUES(13, 'ASID', TO_DATE('18-05-2008', 'DD-MM-YYYY'), 4);  
  
INSERT INTO ASOCIATIE  
VALUES(14, 'ASL', TO_DATE('14-07-2010', 'DD-MM-YYYY'), 5);  
  
COMMIT;
```

COD_ASOCIATIE	DENUMIRE	DATA_INFIINTARII	COD_FACULTATE
10	ASMI	12-APR-09	1
11	ASD	17-JUN-11	2
12	ASG	02-AUG-12	3
13	ASID	18-MAY-08	4
14	ASL	14-JUL-10	5

```
CREATE TABLE SECTIE(cod_sectie NUMBER(5) CONSTRAINT PKEY_SECTIE PRIMARY KEY,  
                    denumire VARCHAR(100) CONSTRAINT denumire_serie NOT NULL,  
                    cod_facultate NUMBER(5),  
                    CONSTRAINT fk_sect FOREIGN KEY(cod_facultate) REFERENCES  
FACULTATE(cod_facultate)  
  
                    );
```

```
Select *  
from sectie;
```

```
INSERT INTO SECTIE  
VALUES(20,'Informatica', 1);
```

```
INSERT INTO SECTIE  
VALUES(21,'Matematica', 1);
```

```
INSERT INTO SECTIE  
VALUES(22,'Calculatoare si Tehnologia Informatiei', 1);
```

```
INSERT INTO SECTIE  
VALUES(23,'Drept Privat', 2);
```

```
INSERT INTO SECTIE  
VALUES (24,'Drept Public',2);
```

```
INSERT INTO SECTIE  
VALUES (25,' Drept Penal',2);
```

⚡ COD_SECTIE	⚡ DENUMIRE	⚡ COD_FACULTATE
20	Informatica	1
21	Matematica	1
22	Calculatoare si Tehnologia Informatiei	1
23	Drept Privat	2
24	Drept Public	2
25	Drept Penal	2


```
-----SERIE-----  
-----  
  
CREATE TABLE SERIE(cod_serie NUMBER(5) CONSTRAINT PKEY_SERIE PRIMARY KEY,  
                    denumire VARCHAR(100) CONSTRAINT denumire_sectie NOT NULL,  
                    cod_sectie NUMBER(5),  
                    CONSTRAINT fk_ser FOREIGN KEY( cod_sectie) REFERENCES  
SECTIE(cod_sectie)  
  
                    );  
  
Select *  
from serie;  
  
INSERT INTO SERIE  
VALUES(30,'13', 20);  
  
INSERT INTO SERIE  
VALUES(31,'14', 20);  
  
INSERT INTO SERIE  
VALUES(32,'15', 20);  
  
INSERT INTO SERIE  
VALUES(33,'16', 21);  
  
INSERT INTO SERIE  
VALUES (34,'11',22);  
  
INSERT INTO SERIE  
VALUES (35,'12',22);
```

	COD_SERIE	DENUMIRE	COD_SECTIE
1	30 13		20
2	31 14		20
3	32 15		20
4	33 16		21
5	34 11		22
6	35 12		22

```
-----GRUPA-----  
-----  
CREATE TABLE GRUPA(cod_grupa NUMBER(5) CONSTRAINT PKEY_GRUPA PRIMARY KEY,  
                    denumire VARCHAR(100) CONSTRAINT denumire_grupa NOT NULL,  
                    cod_serie NUMBER(5),  
                    CONSTRAINT fk_grupa FOREIGN KEY( cod_serie) REFERENCES  
SERIE(cod_serie)  
  
                    );  
  
select *  
from grupa;  
  
INSERT INTO GRUPA  
VALUES(40,'131', 30);  
  
INSERT INTO GRUPA  
VALUES(41,'132', 30);  
  
INSERT INTO GRUPA  
VALUES(42,'133', 30);  
  
INSERT INTO GRUPA  
VALUES(43,'141', 31);  
  
INSERT INTO GRUPA  
VALUES(44,'142', 31);  
  
INSERT INTO GRUPA  
VALUES(45,'143', 31);  
  
INSERT INTO GRUPA  
VALUES(46,'144', 31);  
  
INSERT INTO GRUPA  
VALUES(47,'151', 32);  
  
INSERT INTO GRUPA  
VALUES(48,'152', 32);  
  
COMMIT;
```

COD_GRUPA	DENUMIRE	COD_SERIE
40	131	30
41	132	30
42	133	30
43	141	31
44	142	31
45	143	31
46	144	31
47	151	32
48	152	32

```
-----STUDENT-----
-----
CREATE TABLE STUDENT(
    cod_student NUMBER(5) CONSTRAINT PKEY_STUDENT PRIMARY KEY,
    nume VARCHAR(100) CONSTRAINT nume_student NOT NULL,
    prenume VARCHAR(100) CONSTRAINT prenume_student NOT NULL,
    data_nasterii DATE CONSTRAINT data_nasterii_const NOT NULL,
    sex VARCHAR(10) CONSTRAINT sex_const NOT NULL,
    nationalitate VARCHAR(30) CONSTRAINT nat_const NOT NULL,
    telefon VARCHAR(20) CONSTRAINT telefon_student NOT NULL,
    mail VARCHAR(50) UNIQUE,
    cod_asociatie NUMBER(5),
    CONSTRAINT fk_student FOREIGN KEY( cod_asociatie) REFERENCES
ASOCIATIE(cod_asociatie)
);

Select *
from student;

INSERT INTO STUDENT
VALUES(51, 'Nimara', 'Dan',
TO_DATE('12-04-2000', 'dd-mm-yyyy'), 'masculin', 'roman', '0743234789', 'dnimara@gmail.com',
10);

INSERT INTO STUDENT
VALUES(52, 'Dima', 'Oana',
TO_DATE('26-01-2000', 'dd-mm-yyyy'), 'feminin', 'roman', '0757674789',
'dima.oana26@gmail.com', 10);

INSERT INTO STUDENT
VALUES(53, 'Miu', 'Adania',
TO_DATE('14-01-2001', 'dd-mm-yyyy'), 'feminin', 'roman', '0756789901',
'adania.miu@gmail.com', null);

INSERT INTO STUDENT
VALUES(54, 'Gherghescu', 'Andreea',
TO_DATE('27-09-2001', 'dd-mm-yyyy'), 'feminin', 'roman', '0778901456',
'gh.andreea@gmail.com', 10);

INSERT INTO STUDENT
VALUES(55, 'Pascu', 'Adrian',
TO_DATE('15-08-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0767891056',
'pascu.adi@gmail.com', 10);

INSERT INTO STUDENT
VALUES(56, 'Baciu', 'Daniel',
TO_DATE('24-06-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0748913234',
'dani.baciu@gmail.com', 10);

INSERT INTO STUDENT
VALUES(57, 'Guleama', 'Dan',
TO_DATE('17-07-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0767458910',
'dan.guleama@gmail.com', null);
```

Sisteme de Gestiune a Bazelor de Date
Anul II - Seria 24

```
INSERT INTO STUDENT
VALUES(58,'Marton', 'Sergiu',
TO_DATE('15-02-2001','dd-mm-yyyy'),'masculin','roman','0742561340',
'sergiu.marton@gmail.com', null);

INSERT INTO STUDENT
VALUES(59,'Vultur', 'Sofia',
TO_DATE('10-12-2001','dd-mm-yyyy'),'feminin','roman','0755678923',
'sofi.vultur@gmail.com', null);

INSERT INTO STUDENT
VALUES(60,'Fritz', 'Raluca',
TO_DATE('11-10-2001','dd-mm-yyyy'),'feminin','roman','0765678432',
'fritz.ralu@gmail.com', 10);
```

COD_STUDENT	NUME	PRENUME	DATA_NASTERII	SEX	NATIONALITATE	TELEFON	MAIL	COD_ASOCIATIE
51	Nimara	Dan	12-APR-00	masculin	roman	0743234789	dnimara@gmail.com	10
52	Dima	Oana	26-JAN-00	feminin	roman	0757674789	dima.oana26@gmail.com	10
53	Miu	Adania	14-JAN-01	feminin	roman	0756789901	adania.miu@gmail.com	(null)
54	Gherghescu	Andreea	27-SEP-01	feminin	roman	0778901456	gh.andreea@gmail.com	10
55	Pascu	Adrian	15-AUG-01	masculin	roman	0767891056	pascu.adi@gmail.com	10
56	Baciu	Daniel	24-JUN-01	masculin	roman	0748913234	dani.baciu@gmail.com	10
57	Guleama	Dan	17-JUL-01	masculin	roman	0767458910	dan.guleama@gmail.com	(null)
58	Marton	Sergiu	15-FEB-01	masculin	roman	0742561340	sergiu.marton@gmail.com	(null)
59	Vultur	Sofia	10-DEC-01	feminin	roman	0755678923	sofi.vultur@gmail.com	(null)
60	Fritz	Raluca	11-OCT-01	feminin	roman	0765678432	fritz.ralu@gmail.com	10

```

-----INSCRIERE-----
-----

CREATE TABLE INSCRIERE(cod_student NUMBER(5) CONSTRAINT pk_c_student REFERENCES
STUDENT(cod_student),
                        data_inscrierii DATE CONSTRAINT data_inscr NOT NULL,
                        cod_grupa NUMBER(5)CONSTRAINT pk_c_grupa REFERENCES GRUPA(cod_grupa)
,
                        CONSTRAINT pk_compus_inscr primary key(cod_grupa,cod_student)
);

select *
from inscriere;

INSERT INTO INSCRIERE
VALUES(51, TO_DATE('01-10-2020','dd-mm-yyyy'), 40);

INSERT INTO INSCRIERE
VALUES(52, TO_DATE('01-10-2020','dd-mm-yyyy'), 41);

INSERT INTO INSCRIERE
VALUES(53, TO_DATE('01-10-2020','dd-mm-yyyy'), 40);

INSERT INTO INSCRIERE
VALUES(54, TO_DATE('01-10-2019','dd-mm-yyyy'), 43);

INSERT INTO INSCRIERE
VALUES(60, TO_DATE('01-10-2019','dd-mm-yyyy'), 43);

INSERT INTO INSCRIERE
VALUES(59, TO_DATE('01-10-2019','dd-mm-yyyy'), 44);

INSERT INTO INSCRIERE
VALUES(58, TO_DATE('01-10-2019','dd-mm-yyyy'), 43);

INSERT INTO INSCRIERE
VALUES(57, TO_DATE('01-10-2019','dd-mm-yyyy'), 42);

INSERT INTO INSCRIERE
VALUES(56, TO_DATE('01-10-2018','dd-mm-yyyy'), 45);

INSERT INTO INSCRIERE
VALUES(55, TO_DATE('01-10-2018','dd-mm-yyyy'), 45);

```

COD_STUDENT	DATA_INSCRIERII	COD_GRUPA
51	01-OCT-20	40
52	01-OCT-20	41
53	01-OCT-20	40
54	01-OCT-19	43
60	01-OCT-19	43
59	01-OCT-19	44
58	01-OCT-19	43
57	01-OCT-19	42
56	01-OCT-18	45
55	01-OCT-18	45

```
-----PROFESOR-----  
-----  
  
CREATE TABLE PROFESOR(  
    cod_profesor NUMBER(5) CONSTRAINT PKEY_profesor PRIMARY KEY,  
    nume VARCHAR(100) CONSTRAINT nume_prof NOT NULL,  
    prenume VARCHAR(100) CONSTRAINT prenume_prof NOT NULL,  
    telefon VARCHAR(20) CONSTRAINT telefon_prof NOT NULL,  
    mail VARCHAR(50) UNIQUE  
  
    );  
  
select *  
from profesor;  
  
INSERT INTO PROFESOR  
VALUES(100, 'Popescu','Ion', '0756789546','ion.popescu@gmail.com');  
  
INSERT INTO PROFESOR  
VALUES(101, 'Ionescu','Irina', '0745678923','irina.ionescu@gmail.com');  
  
INSERT INTO PROFESOR  
VALUES(102, 'Avram','Bianca', '0745678213','bianca.avram@gmail.com');  
  
INSERT INTO PROFESOR  
VALUES(103, 'Branescu','Robert', '0789456234','robbranescu@gmail.com');  
  
INSERT INTO PROFESOR  
VALUES(104, 'Enache','Teodora', '0723563781','enache.teo@gmail.com');  
  
INSERT INTO PROFESOR  
VALUES(105, 'Boboc','Stefania', '0723157368','boboc.stefania@gmail.com');  
  
COMMIT;
```

⚡ COD_PROFESOR	⚡ NUME	⚡ PRENUME	⚡ TELEFON	⚡ MAIL
100	Popescu	Ion	0756789546	ion.popescu@gmail.com
101	Ionescu	Irina	0745678923	irina.ionescu@gmail.com
102	Avram	Bianca	0745678213	bianca.avram@gmail.com
103	Branescu	Robert	0789456234	robbranescu@gmail.com
104	Enache	Teodora	0723563781	enache.teo@gmail.com
105	Boboc	Stefania	0723157368	boboc.stefania@gmail.com

```

-----CONTRACT-----
-----

CREATE TABLE CONTRACT(cod_facultate NUMBER(5) CONSTRAINT pk_c_fac REFERENCES
FACULTATE(cod_facultate),
                        data_inceput DATE CONSTRAINT data_inc NOT NULL,
                        cod_profesor NUMBER(5) CONSTRAINT pk_c_prof REFERENCES
PROFESOR(cod_profesor),
                        salariu NUMBER CONSTRAINT sal_const NOT NULL,
                        CONSTRAINT pk_compus_cont primary key(cod_facultate,cod_profesor)
);

INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-08-2018','dd-mm-yyyy'), 100, 5400);

INSERT INTO CONTRACT
VALUES(2, TO_DATE('01-07-2018','dd-mm-yyyy'), 101, 5200);

INSERT INTO CONTRACT
VALUES(3, TO_DATE('01-06-2017','dd-mm-yyyy'), 100, 5400);

INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-06-2017','dd-mm-yyyy'), 102, 4500);

INSERT INTO CONTRACT
VALUES (1, TO_DATE('01-06-2017','dd-mm-yyyy'), 103, 4200);

INSERT INTO CONTRACT
VALUES (1,TO_DATE('01-06-2017','dd-mm-yyyy'), 104, 4700);

INSERT INTO CONTRACT
VALUES (1,TO_DATE('01-06-2019','dd-mm-yyyy'), 105, 4700);

INSERT INTO CONTRACT
VALUES (4,TO_DATE('01-06-2016','dd-mm-yyyy'), 104, 5400);

INSERT INTO CONTRACT
VALUES (5,TO_DATE('02-08-2015','dd-mm-yyyy'), 102, 5700);

INSERT INTO CONTRACT
VALUES (5,TO_DATE('13-07-2016','dd-mm-yyyy'), 104, 3600);

INSERT INTO CONTRACT
VALUES (2,TO_DATE('19-06-2017','dd-mm-yyyy'), 102, 3800);

```

⚡ COD_FACULTATE	⚡ DATA_INCEPUT	⚡ COD_PROFESOR	⚡ SALARIU
1	01-AUG-18	100	5400
2	01-JUL-18	101	5200
3	01-JUN-17	100	5400
1	01-JUN-17	102	4500
1	01-JUN-17	103	4200
1	01-JUN-17	104	4700
1	01-JUN-19	105	4700
4	01-JUN-16	104	5400
5	02-AUG-15	102	5700
5	13-JUL-16	104	3600
2	19-JUN-17	102	3800

```
-----SALA-----  
-----  
  
CREATE TABLE SALA(cod_sala NUMBER(5) CONSTRAINT PKEY_sala PRIMARY KEY,  
                   denumire VARCHAR(30) CONSTRAINT denumire_sala NOT NULL,  
                   locatie VARCHAR(50) CONSTRAINT locatie_sala NOT NULL  
                   );  
  
select *  
from sala;  
  
INSERT INTO SALA  
VALUES (200,'Amfiteatrul Titulescu', 'Str. Alexandru Lapusneanu, nr.11');  
  
INSERT INTO SALA  
VALUES (201,'Amfiteatrul Haret', 'Str. Ion Minulescu, nr.12');  
  
INSERT INTO SALA  
VALUES (202,'Amfiteatrul Pompeiu', 'Str. Stefan cel Mare, nr.201');  
  
INSERT INTO SALA  
VALUES (203,'Amfiteatrul Moisil', 'Str. Lebedei, nr.304');  
  
INSERT INTO SALA  
VALUES (204,'Amfiteatrul Lalescu', 'Str. Herastrau, nr.215');  
  
INSERT INTO SALA  
VALUES (205,'Amfiteatrul Ghika', 'Splaiul Independentei, nr.5');  
  
INSERT INTO SALA  
VALUES (206,'Amfiteatrul Barbilian', 'Str. Unirii, nr.20');
```

COD_SALA	DENUMIRE	LOCATIE
200	Amfiteatrul Titulescu	Str. Alexandru Lapusneanu, nr.11
201	Amfiteatrul Haret	Str. Ion Minulescu, nr.12
202	Amfiteatrul Pompeiu	Str. Stefan cel Mare, nr.201
203	Amfiteatrul Moisil	Str. Lebedei, nr.304
204	Amfiteatrul Lalescu	Str. Herastrau, nr.215
205	Amfiteatrul Ghika	Splaiul Independentei, nr.5
206	Amfiteatrul Barbilian	Str. Unirii, nr.20


```
-----CURS-----  
-----  
  
CREATE TABLE CURS(cod_curs NUMBER(5) CONSTRAINT PKEY_curs PRIMARY KEY,  
                    denumire VARCHAR(50) CONSTRAINT denumire_curs NOT NULL  
                    );  
  
select *  
from curs;  
  
INSERT INTO CURS  
VALUES (300, 'Arhitectura sistemelor de calcul');  
  
INSERT INTO CURS  
VALUES (301, 'Gandire Critica si Etica Academica');  
  
INSERT INTO CURS  
VALUES (302, 'Limba si Literatura Engleza');  
  
INSERT INTO CURS  
VALUES (303, 'Programare Orientata pe Obiecte');  
  
INSERT INTO CURS  
VALUES (304, 'Educatie Fizica');  
  
INSERT INTO CURS  
VALUES (305, 'Baze de Date');  
  
INSERT INTO CURS  
VALUES (306, 'Istoria Religiilor');  
  
INSERT INTO CURS  
VALUES (307, 'Chimie organica');
```

COD_...	DENUMIRE
300	Arhitectura sistemelor de calcul
301	Gandire Critica si Etica Academica
302	Limba si Literatura Engleza
303	Programare Orientata pe Obiecte
304	Educatie Fizica
305	Baze de Date
306	Istoria Religiilor
307	Chimie organica

```
-----EXAMEN-----
CREATE TABLE EXAMEN(cod_examen NUMBER(5) CONSTRAINT PKEY_examen PRIMARY KEY,
                    forma VARCHAR(30) CONSTRAINT forma_const NOT NULL,
                    cod_curs NUMBER(5),
                    CONSTRAINT fk_examen FOREIGN KEY( cod_curs) REFERENCES
CURS(cod_curs)
                    );

select *
from examen;

CREATE SEQUENCE SEQ_EXAM
INCREMENT by 10
START WITH 400
MAXVALUE 10000
NOCYCLE;

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Examen Scris', 300);

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Examen Scris', 301);

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Examen Scris', 302);

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Interviu', 303);

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Interviu', 304);

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Proiect', 305);

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Proiect', 306);

INSERT INTO EXAMEN
VALUES (SEQ_EXAM.NEXTVAL, 'Examen Scris', 307);
```

COD_EXAMEN	FORMA	COD_CURS
400	Examen Scris	300
410	Examen Scris	301
420	Examen Scris	302
430	Interviu	303
440	Interviu	304
450	Proiect	305
460	Proiect	306
470	Examen Scris	307

```
-----PROMOVEAZA-----  
-----  
  
CREATE TABLE PROMOVEAZA(  
    nota NUMBER(5,2) CONSTRAINT nota_const NOT NULL,  
    cod_student NUMBER(5) CONSTRAINT pk_c_s REFERENCES  
STUDENT(cod_student),  
    cod_examen NUMBER(5) CONSTRAINT pk_c_e REFERENCES  
EXAMEN(cod_examen),  
    CONSTRAINT pk_compus_prom primary key(cod_student,cod_examen)  
);  
  
select *  
from promoveaza;  
  
INSERT INTO PROMOVEAZA  
VALUES (10,51, 410);  
  
INSERT INTO PROMOVEAZA  
VALUES (10,52, 470);  
  
INSERT INTO PROMOVEAZA  
VALUES (8.5,53, 480);  
  
INSERT INTO PROMOVEAZA  
VALUES (7.5,54, 480);  
  
INSERT INTO PROMOVEAZA  
VALUES (9.5,55, 430);  
  
INSERT INTO PROMOVEAZA  
VALUES (8,56, 470);  
  
INSERT INTO PROMOVEAZA  
VALUES (7.5,53, 460);  
  
INSERT INTO PROMOVEAZA  
VALUES (9.3,55,460);  
  
INSERT INTO PROMOVEAZA  
VALUES (10,60, 470);  
  
INSERT INTO PROMOVEAZA  
VALUES (10,60, 420);  
  
INSERT INTO PROMOVEAZA  
VALUES (8.7, 51, 430);  
  
INSERT INTO PROMOVEAZA  
VALUES (5.6, 60, 460);  
  
INSERT INTO PROMOVEAZA
```

Sisteme de Gestiuine a Bazelor de Date
Anul II - Seria 24

```
VALUES (8.7, 59, 420);
```

```
INSERT INTO PROMOVEAZA  
VALUES (7.7, 57, 470);
```

NOTA	COD_STUDENT	COD_EXAMEN
10	51	410
10	52	470
9.5	55	430
8	56	470
7.5	53	460
9.3	55	460
10	60	470
10	60	420
8.7	51	430
5.6	60	460
8.7	59	420
7.7	57	470

```
-----ORAR-----  
  
CREATE TABLE ORAR(cod_grupa NUMBER(5) CONSTRAINT pk_c_gr REFERENCES GRUPA(cod_grupa),  
                   cod_curs  NUMBER(5) CONSTRAINT pk_c_curs REFERENCES CURS(cod_curs),  
                   cod_profesor NUMBER(5) CONSTRAINT pk_c_ REFERENCES  
PROFESOR(cod_profesor),  
                   cod_sala NUMBER(5) CONSTRAINT pk_c_sala REFERENCES SALA(cod_sala),  
                   CONSTRAINT pk_compus_orar primary key(cod_grupa,  
cod_curs,cod_profesor,cod_sala)  
                   );  
  
select *  
from orar;  
  
INSERT INTO ORAR  
VALUES (40,300,100,200);  
  
INSERT INTO ORAR  
VALUES (41,303,102,201);  
  
INSERT INTO ORAR  
VALUES (42,305,105,205);  
  
INSERT INTO ORAR  
VALUES (43,300,103,202);  
  
INSERT INTO ORAR  
VALUES (44,302,102,203);  
  
INSERT INTO ORAR  
VALUES (45,303,101,202);  
  
INSERT INTO ORAR  
VALUES (46,305,101,201);  
  
COMMIT;
```

⚡ COD_GRUPA	⚡ COD_CURS	⚡ COD_PROFESOR	⚡ COD_SALA
40	300	100	200
41	303	102	201
42	305	105	205
43	300	103	202
44	302	102	203
45	303	101	202
46	305	101	201

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.

```
-- pentru o facultate al carei nume este dat, pentru fiecare amfiteatru, afisati
studentii ai caror grupe desfasoara ore in amfiteatrul curent sau afisati 'nu exista'
-- daca pentru grupa x nu exista studenti care se aiba ore in amfiteatrul y

CREATE OR REPLACE PROCEDURE Ex6 (denumirefac FACULTATE.denumire%TYPE)
AS
    TYPE tabl_idx IS TABLE OF sala%rowtype INDEX BY PLS_INTEGER;
    amfiteatre tabl_idx;

    TYPE tip_lista_nested IS TABLE OF grupa%rowtype;
    grupe tip_lista_nested := tip_lista_nested();

    TYPE tabl_index IS TABLE OF VARCHAR(200) INDEX BY PLS_INTEGER;
    v_numa tabl_index;

    numar NUMBER(6);
BEGIN
    SELECT *
    BULK COLLECT INTO amfiteatre
    FROM sala;

    SELECT COUNT(*)
    INTO numar
    FROM grupa g, sectie sect, serie ser, facultate f
    WHERE g.cod_serie = ser.cod_serie and ser.cod_sectie = sect.cod_sectie and
    sect.cod_facultate = f.cod_facultate
    AND UPPER(f.denumire) LIKE UPPER(denumirefac) AND ROWNUM <= 1000;

    grupe.extend(numar + 1);

    SELECT g.cod_grupa, g.denumire, g.cod_serie
    BULK COLLECT INTO grupe
    FROM grupa g, sectie sect, serie ser, facultate f
    WHERE g.cod_serie = ser.cod_serie and ser.cod_sectie = sect.cod_sectie and
    sect.cod_facultate = f.cod_facultate
    AND UPPER(f.denumire) LIKE UPPER('Facultatea de Matematica%');

    FOR i IN amfiteatre.first..amfiteatre.last LOOP
        DBMS_OUTPUT.PUT_LINE('Amfiteatrul ' || amfiteatre(i).denumire);
        DBMS_OUTPUT.PUT_LINE('-----');

        FOR j IN grupe.first..grupe.last LOOP
            DBMS_OUTPUT.PUT_LINE('Grupa ' || grupe(j).denumire);
            DBMS_OUTPUT.PUT_LINE('-----');
            SELECT s.numa || ' ' || s.prenume
            BULK COLLECT INTO v_numa
            FROM student s, ora o, inscriere ins
            WHERE s.cod_student = ins.cod_student and ins.cod_grupa = grupe(j).cod_grupa
```

```
and o.cod_grupa = grupe(j).cod_grupa and amfiteatre(i).cod_sala = o.cod_sala;
    IF v_nume.count > 0 THEN
        numar := 0;
        FOR k in v_nume.first..v_nume.last LOOP
            numar := numar + 1;
            DBMS_OUTPUT.PUT_LINE(numar || ' ' || v_nume(k));
        END LOOP;
    END IF;
    IF v_nume.count = 0
    THEN DBMS_OUTPUT.PUT_LINE('Nu exista');
    END IF;
    DBMS_OUTPUT.NEW_LINE;
    DBMS_OUTPUT.NEW_LINE;
END LOOP;
END LOOP;

END;
/

BEGIN
    Ex6('Facultatea de Matematica si Informatica');
END;
/
```

-- nu este tot outputul

The screenshot displays the SQL Developer interface. The 'Query Builder' window shows the PL/SQL code being executed. The 'Results' window on the right shows the output of the procedure, which lists groups and their members. The 'Log' window at the bottom shows the execution status.

Query Builder:

```
DBMS_OUTPUT.NEW_LINE;
DBMS_OUTPUT.NEW_LINE;
END LOOP;
END LOOP;

END;
/

BEGIN
    Ex6('Facultatea de Matematica si Informatica');
END;
/
```

Results:

Amfiteatrul	Amfiteatrul Titulescu
Grupa 131	
1. Nimara Dan	
2. Miu Adania	
Grupa 132	
Nu exista	
Grupa 133	
Nu exista	
Grupa 141	
Nu exista	
Grupa 142	
Nu exista	
Grupa 143	
Nu exista	
Grupa 144	
Nu exista	
Grupa 151	
Nu exista	

Log:

```
PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.

Procedure EX6 compiled

Procedure EX6 compiled

PL/SQL procedure successfully completed.

PL/SQL procedure successfully completed.
```

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.

```
-- pentru examenele care au ca forma 'Examen Scris'
-- afisati studentii care le-au promovat cu nota mai mare de 7
CREATE OR REPLACE PROCEDURE Ex7 (tip_examen examen.forma%TYPE, nota_examen
promoveaza.nota%TYPE)
AS
    CURSOR cursuri (cod curs.cod_curs%TYPE) IS
        SELECT denumire
        FROM curs
        WHERE cod_curs = cod;

    CURSOR examene (tip_examen examen.forma%TYPE) IS
        SELECT cod_examen, cod_curs
        FROM examen
        WHERE UPPER(forma) LIKE UPPER(tip_examen);

    CURSOR studenti (cod examen.cod_examen%TYPE ) IS
        SELECT s.num || ' ' || s.prenume || ' a obtinut nota ' || p.nota as
result
        FROM promoveaza p, student s
        WHERE p.cod_student = s.cod_student AND p.cod_examen = cod AND p.nota >=
nota_examen;

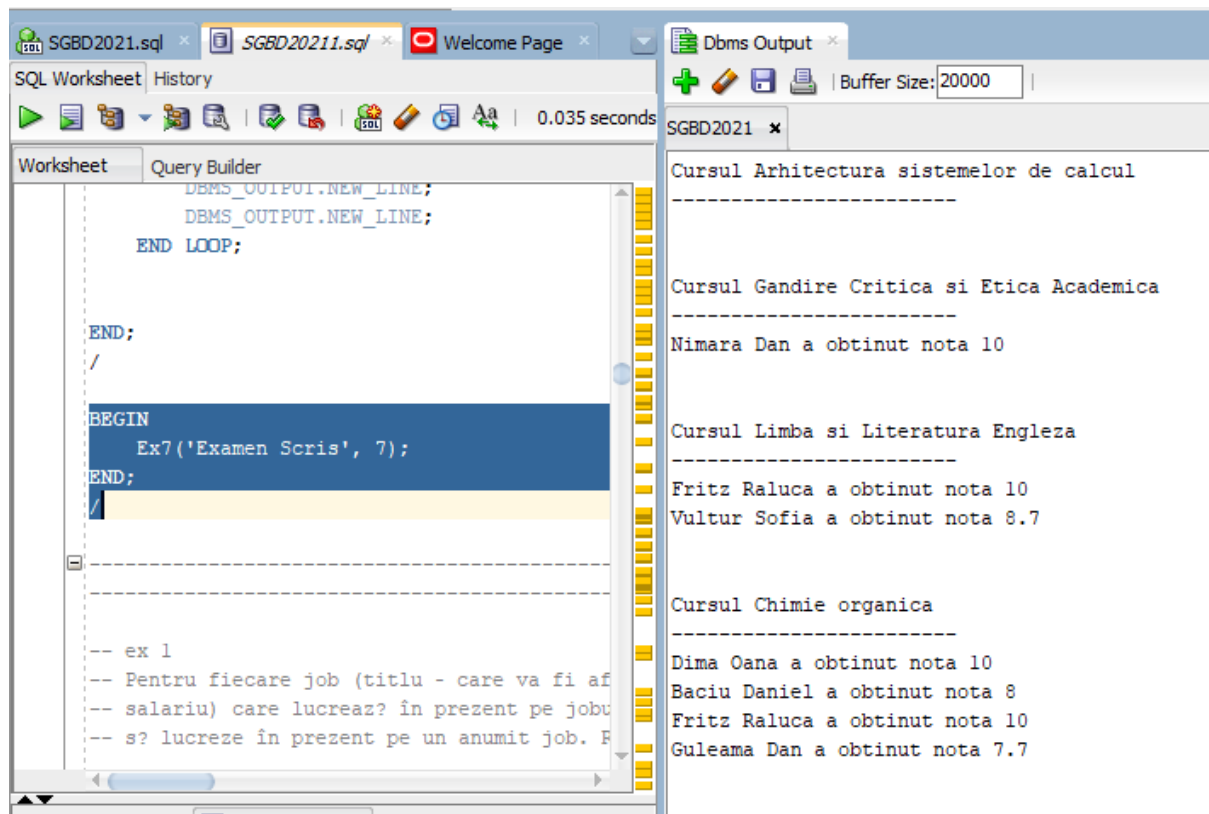
    den curs.denumire%TYPE;
    cod curs.cod_curs%TYPE;
BEGIN
    FOR examen in examene(tip_examen) LOOP
        OPEN cursuri(examen.cod_curs);
        FETCH cursuri INTO den;
        DBMS_OUTPUT.PUT_LINE('Cursul ' || den);
        DBMS_OUTPUT.PUT_LINE('-----');
        CLOSE cursuri;

        FOR student IN studenti(examen.cod_examen) LOOP
            DBMS_OUTPUT.PUT_LINE(student.result);
        END LOOP;

        DBMS_OUTPUT.NEW_LINE;
        DBMS_OUTPUT.NEW_LINE;
    END LOOP;

END;
/

BEGIN
    Ex7('Examen Scris', 7);
END;
/
```

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.

-- pentru exemplul acesta, o sa mai introduc cateva date in tabelele profesor si contract

```
INSERT INTO PROFESOR
VALUES(106, 'Georgescu','Ion', '0745678923','ion.geo@gmail.com');

INSERT INTO PROFESOR
VALUES(107, 'Visan','Irina', '0726357985','irina.visan@gmail.com');

INSERT INTO PROFESOR
VALUES(108, 'Paduraru','Bianca', '0722537253','bianca.pad@gmail.com');

INSERT INTO PROFESOR
VALUES(109, 'Irinescu','Robert', '0721548293','robertiri@gmail.com');

INSERT INTO PROFESOR
VALUES(110, 'Ene','Teodora', '0723234545','ene.teo@gmail.com');

INSERT INTO PROFESOR
```

```
VALUES(111, 'Baciu','Stefania', '0719323354','bstefania@gmail.com');
COMMIT;

INSERT INTO PROFESOR
VALUES(112, 'Iorga','Flavius', '0745678291','iflav@gmail.com');

INSERT INTO PROFESOR
VALUES(113, 'Enescu','Violeta', '0745678291','vio.enescu@gmail.com');
COMMIT;

---
INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-08-2018','dd-mm-yyyy'), 113, 5400);

INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-08-2018','dd-mm-yyyy'), 111, 5400);

INSERT INTO CONTRACT
VALUES(4, TO_DATE('01-07-2018','dd-mm-yyyy'), 108, 5200);

INSERT INTO CONTRACT
VALUES(3, TO_DATE('01-06-2017','dd-mm-yyyy'), 107, 5400);

INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-06-2017','dd-mm-yyyy'), 109, 4500);

INSERT INTO CONTRACT
VALUES (1, TO_DATE('01-06-2017','dd-mm-yyyy'), 110, 4200);

INSERT INTO CONTRACT
VALUES (1,TO_DATE('01-06-2017','dd-mm-yyyy'), 106, 4700);

INSERT INTO CONTRACT
VALUES (1,TO_DATE('01-06-2019','dd-mm-yyyy'), 107, 4700);

INSERT INTO CONTRACT
VALUES (4,TO_DATE('01-06-2016','dd-mm-yyyy'), 108, 5400);

INSERT INTO CONTRACT
VALUES (2,TO_DATE('02-08-2015','dd-mm-yyyy'), 111, 5700);

INSERT INTO CONTRACT
VALUES (5,TO_DATE('13-07-2016','dd-mm-yyyy'), 111, 3600);

INSERT INTO CONTRACT
VALUES (5,TO_DATE('19-06-2017','dd-mm-yyyy'), 108, 3800);

select * from contract;
select * from profesor;
```

COD_FACULTATE	DATA_INCEPUT	COD_PROFESOR	SALARIU
1	01-AUG-18	111	5400
2	01-JUL-18	108	5200
3	01-JUN-17	107	5400
1	01-JUN-17	109	4500
1	01-JUN-17	110	4200
1	01-JUN-17	106	4700
1	01-JUN-19	107	4700
4	01-JUN-16	108	5400
5	02-AUG-15	111	5700
2	02-AUG-15	111	5700
5	19-JUN-17	108	3800
1	01-AUG-18	113	5400
1	01-AUG-18	100	5400
2	01-JUL-18	101	5200
3	01-JUN-17	100	5400
1	01-JUN-17	102	4500
1	01-JUN-17	103	4200
1	01-JUN-17	104	4700
1	01-JUN-19	105	4700

COD_PROF...	NUME	PRENUME	TELEFON	MAIL
106	Georgescu	Ion	0745678923	ion.geo@gmail.com
107	Visan	Irina	0726357985	irina.visan@gmail.com
108	Paduraru	Bianca	0722537253	bianca.pad@gmail.com
109	Irinescu	Robert	0721548293	robertiri@gmail.com
110	Ene	Teodora	0723234545	ene.teo@gmail.com
111	Baciu	Stefania	0719323354	bstefania@gmail.com
112	Iorga	Flavius	0745678291	iflav@gmail.com
113	Enescu	Violeta	0745678291	vio.enescu@gmail.com
100	Popescu	Ion	0756789546	ion.popescu@gmail.com
101	Ionescu	Irina	0745678923	irina.ionescu@gmail.com
102	Avram	Bianca	0745678213	bianca.avram@gmail.com
103	Branescu	Robert	0789456234	robbranescu@gmail.com
104	Enache	Teodora	0723563781	enache.teo@gmail.com
105	Boboc	Stefania	0723157368	boboc.stefania@gmail.com

Cerinta:

- pentru un prenume al unui profesor dat, determinati numarul de facultati la care acesta este angajat
- daca prenumele lui apare de cel putin doua ori, selectand doar facultatile care cifra 1 in codul postal

```
CREATE OR REPLACE FUNCTION Ex8(cod profesor.cod_profesor%TYPE) RETURN NUMBER
IS
    nr_facultati NUMBER;
    TYPE tip_tabel IS TABLE OF contract%rowtype INDEX BY PLS_INTEGER;
    tabel tip_tabel;
    pren profesor.prenume%type;
    --exceptii
    NEGATIVE_NUMBER EXCEPTION;
    NO_DATA_FOUND1 EXCEPTION;
    NO_DATA_FOUND2 EXCEPTION;

BEGIN
    IF cod < 0 THEN -- codul profesorului nu e valid
        RAISE NEGATIVE_NUMBER;
    END IF;

    SELECT *
    BULK COLLECT INTO tabel
    FROM contract
    WHERE cod_profesor = cod;

    IF SQL%NOTFOUND THEN
        RAISE NO_DATA_FOUND1;
    END IF;

    SELECT prenume
    INTO pren
    FROM profesor
    WHERE cod_profesor = cod;

    SELECT COUNT(f.cod_facultate)
    INTO nr_facultati
    FROM facultate f
    JOIN contract c ON (c.cod_facultate = f.cod_facultate)
    WHERE c.cod_profesor = cod
    AND (SELECT COUNT(prenume)FROM profesor WHERE UPPER(prenume) LIKE UPPER(pren) ) >= 2
    AND cod_postal LIKE '%1%';

    IF nr_facultati = 0 THEN
        RAISE NO_DATA_FOUND2;
    ELSE RETURN nr_facultati;
    END IF;

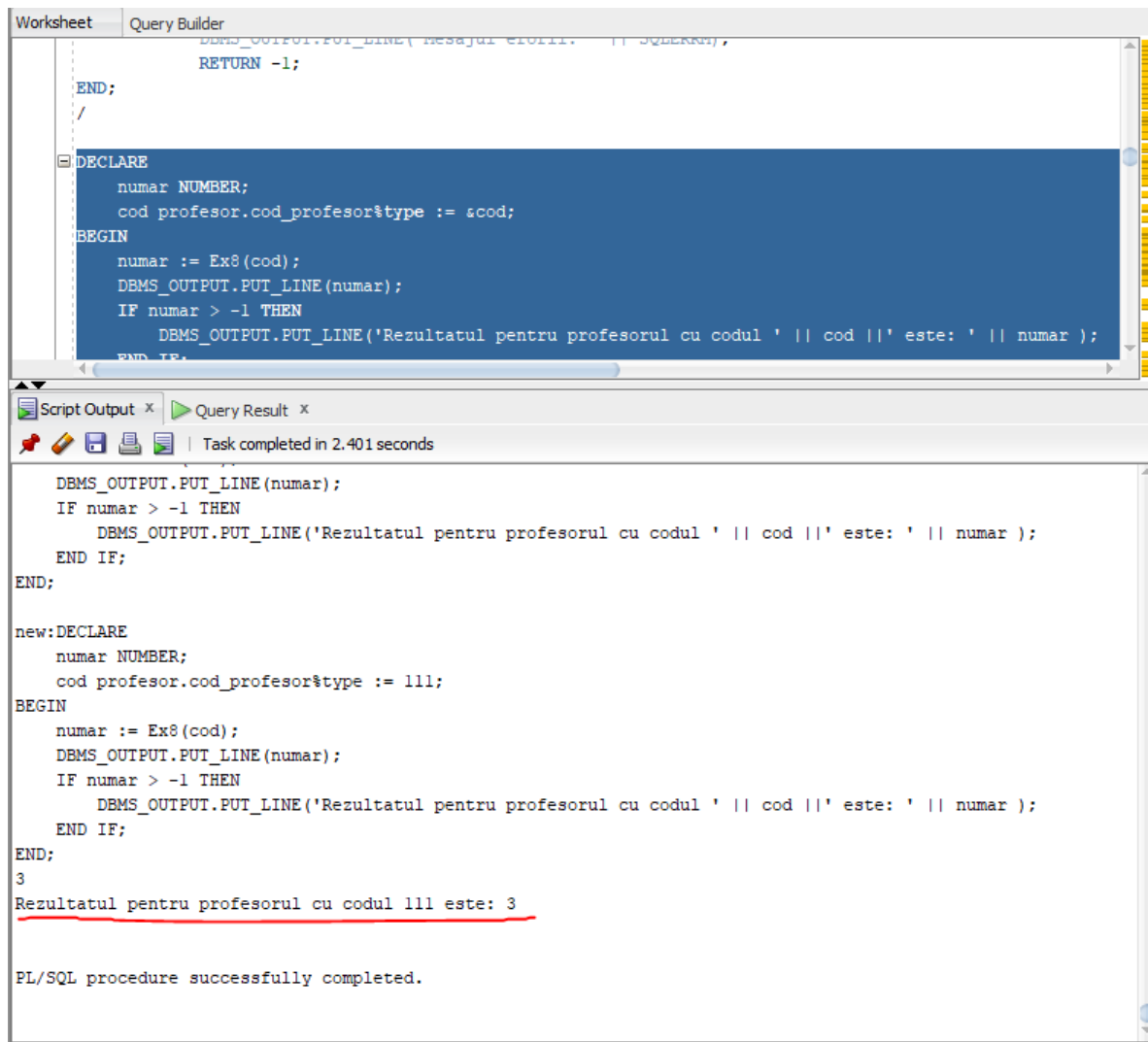
    EXCEPTION
        WHEN NEGATIVE_NUMBER THEN
            DBMS_OUTPUT.PUT_LINE('Codul profesorului nu poate sa fie negativ!');
            RETURN -1;
        WHEN NO_DATA_FOUND1 THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista profesorul cu codul ' || cod || ' in tabela
contract!');
            RETURN -1;
        WHEN NO_DATA_FOUND2 THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista facultati care sa indeplineasca acele
conditii!');
```

```
        RETURN -1;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' || SQLCODE);
        DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' || SQLERRM);
        RETURN -1;
END;
/

DECLARE
    numar NUMBER;
    cod profesor.cod_profesor%type := &cod;
BEGIN
    numar := Ex8(cod);
    DBMS_OUTPUT.PUT_LINE(numar);
    IF numar > -1 THEN
        DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: '
|| numar );
    END IF;
END;
/
```

OUTPUT-URI:

-- pentru cod = 111



The screenshot displays the Oracle SQL Developer interface. The top pane, titled 'Query Builder', contains a PL/SQL script. The bottom pane, titled 'Script Output', shows the execution results of the script. The script defines a procedure that takes a cod value and returns a number. The output shows the procedure being executed with cod = 111, resulting in the number 3.

```
DBMS_OUTPUT.PUT_LINE('Mesajul este: ' || SQLERRM);
RETURN -1;
END;
/

DECLARE
  numar NUMBER;
  cod profesor.cod_profesor$type := scod;
BEGIN
  numar := Ex8(cod);
  DBMS_OUTPUT.PUT_LINE(numar);
  IF numar > -1 THEN
    DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
  END IF;
END;
```

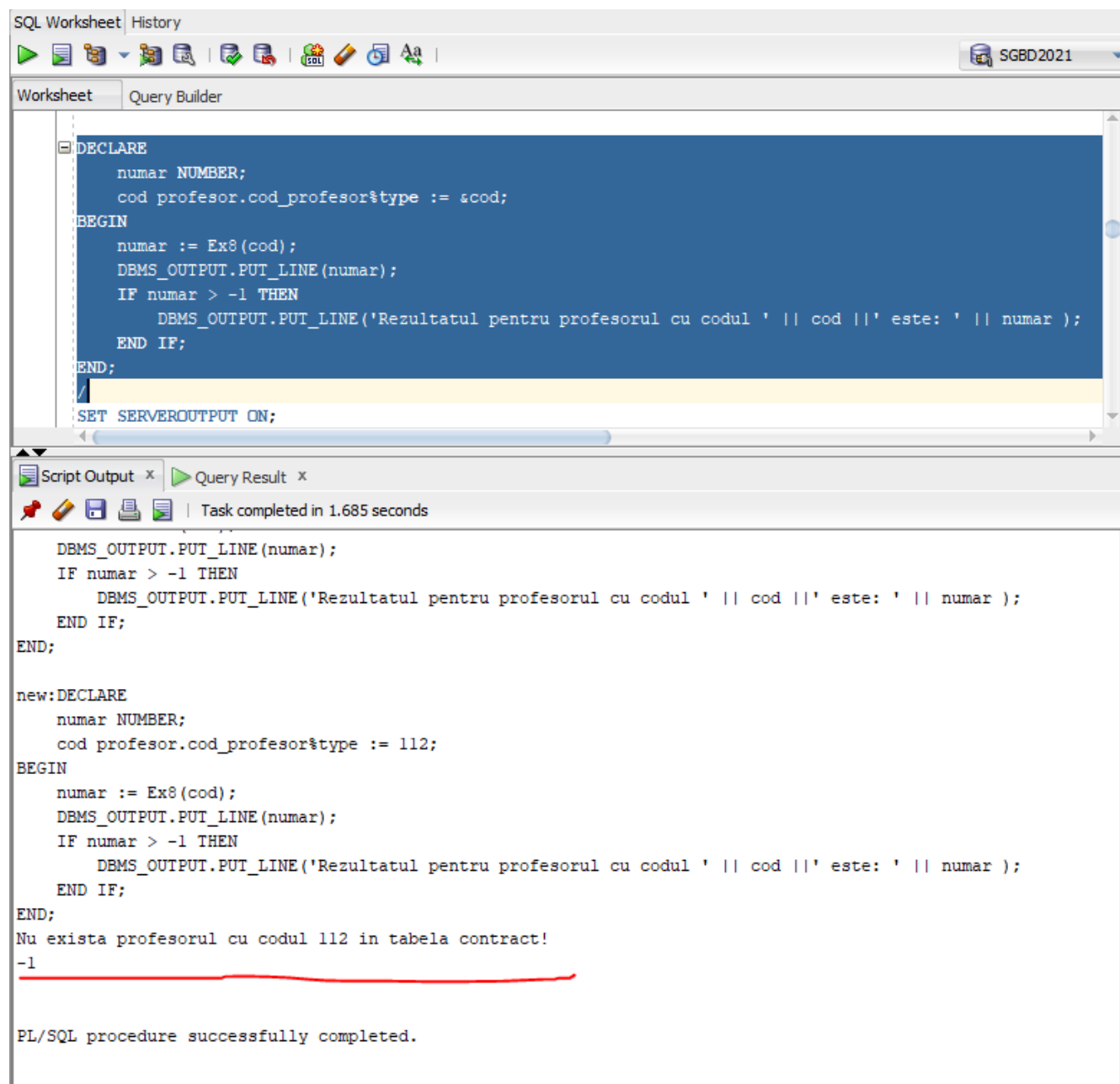
Task completed in 2.401 seconds

```
DBMS_OUTPUT.PUT_LINE(numar);
IF numar > -1 THEN
  DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
END IF;
END;

new:DECLARE
  numar NUMBER;
  cod profesor.cod_profesor$type := 111;
BEGIN
  numar := Ex8(cod);
  DBMS_OUTPUT.PUT_LINE(numar);
  IF numar > -1 THEN
    DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
  END IF;
END;
3
Rezultatul pentru profesorul cu codul 111 este: 3

PL/SQL procedure successfully completed.
```

-- pentru cod = 112



The screenshot shows an SQL IDE with a 'Worksheet' tab. The main editor contains a PL/SQL procedure. Below the editor, the 'Script Output' tab is active, displaying the execution results. The procedure is as follows:

```
DECLARE
    numar NUMBER;
    cod profesor.cod_profesor$type := &cod;
BEGIN
    numar := Ex8(cod);
    DBMS_OUTPUT.PUT_LINE(numar);
    IF numar > -1 THEN
        DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
    END IF;
END;
```

The 'Script Output' tab shows the following output:

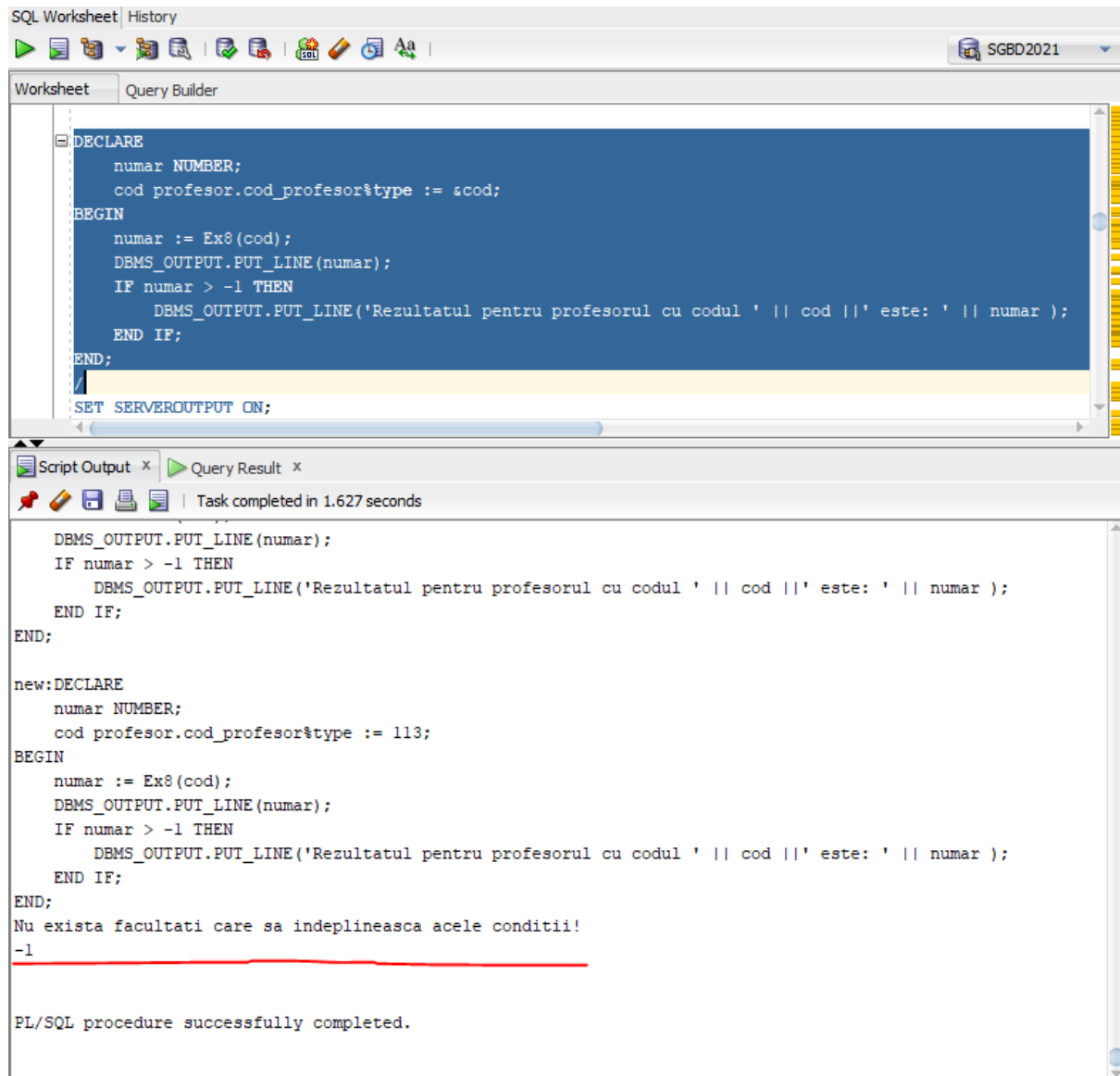
```
DBMS_OUTPUT.PUT_LINE (numar);
IF numar > -1 THEN
    DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
END IF;
END;
```

new:DECLARE
numar NUMBER;
cod profesor.cod_profesor\$type := 112;
BEGIN
numar := Ex8(cod);
DBMS_OUTPUT.PUT_LINE(numar);
IF numar > -1 THEN
DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar);
END IF;
END;

Nu exista profesorul cu codul 112 in tabela contract!
-1

PL/SQL procedure successfully completed.

-- pentru cod = 113



The screenshot shows an SQL Worksheet interface with a 'Query Builder' tab. The main window displays a PL/SQL procedure. Below the editor, the 'Script Output' and 'Query Result' tabs are visible. The 'Query Result' tab shows the execution output, including the procedure code with a hardcoded value of 113, the execution result, and a confirmation message.

```
DECLARE
    numar NUMBER;
    cod profesor.cod_profesor$type := scod;
BEGIN
    numar := Ex8(cod);
    DBMS_OUTPUT.PUT_LINE(numar);
    IF numar > -1 THEN
        DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
    END IF;
END;

SET SERVEROUTPUT ON;
```

Task completed in 1.627 seconds

```
DBMS_OUTPUT.PUT_LINE(numar);
IF numar > -1 THEN
    DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
END IF;
END;
```

new:DECLARE

```
    numar NUMBER;
    cod profesor.cod_profesor$type := 113;
BEGIN
    numar := Ex8(cod);
    DBMS_OUTPUT.PUT_LINE(numar);
    IF numar > -1 THEN
        DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
    END IF;
END;
```

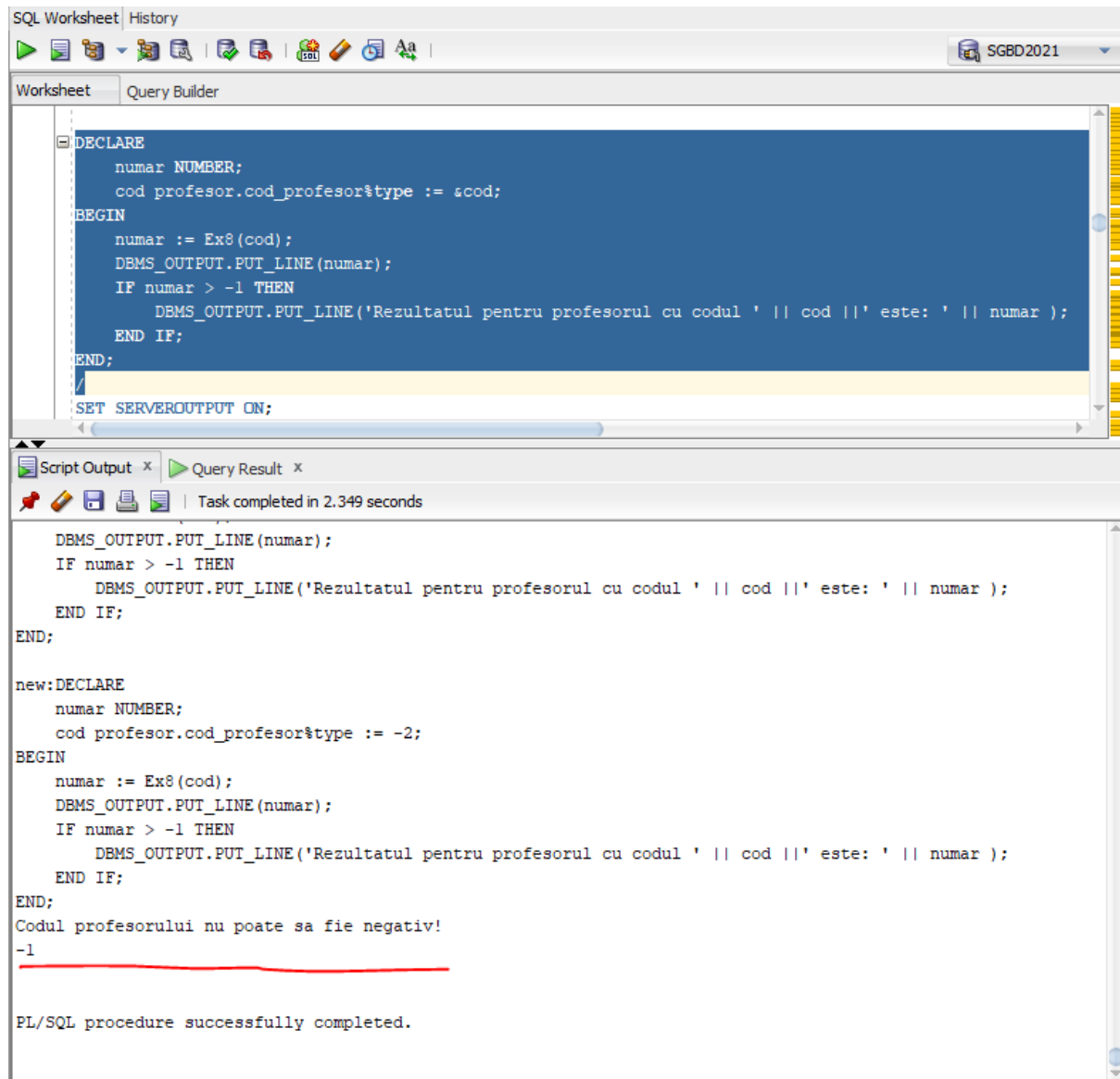
END;

Nu exista facultati care sa indeplineasca acele conditii!

-1

PL/SQL procedure successfully completed.

- pentru cod = -2



The screenshot shows an SQL Worksheet interface with a 'Query Builder' tab. The main editor contains a PL/SQL block. Below the editor, the 'Script Output' and 'Query Result' tabs are visible. The 'Script Output' tab shows the execution results, including a message indicating that the PL/SQL procedure was successfully completed.

```
DECLARE
    numar NUMBER;
    cod profesor.cod_profesor$type := scod;
BEGIN
    numar := Ex8(cod);
    DBMS_OUTPUT.PUT_LINE(numar);
    IF numar > -1 THEN
        DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar );
    END IF;
END;

SET SERVEROUTPUT ON;
```

DBMS_OUTPUT.PUT_LINE(numar);
IF numar > -1 THEN
 DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar);
END IF;
END;

new:DECLARE
 numar NUMBER;
 cod profesor.cod_profesor\$type := -2;
BEGIN
 numar := Ex8(cod);
 DBMS_OUTPUT.PUT_LINE(numar);
 IF numar > -1 THEN
 DBMS_OUTPUT.PUT_LINE('Rezultatul pentru profesorul cu codul ' || cod || ' este: ' || numar);
 END IF;
END;
Codul profesorului nu poate sa fie negativ!
-1

PL/SQL procedure successfully completed.

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. Cerinta:

-- pentru un student al carui nume este dat, afisez cursurile la care e inscris, luandu le in calcul doar pe cele desfasurate in amfiteatre

-- cu litera i in denumire

-- mai fac 2 inserari in tabela student pentru a exemplifica eroarea de too many rows

```
INSERT INTO STUDENT
VALUES(61,'Nimara', 'Irina',
TO_DATE('19-03-2000','dd-mm-yyyy'),'feminin','roman','0756428912',
'irina_nim@gmail.com', 10);
COMMIT;
INSERT INTO STUDENT
VALUES(62,'Teodorescu', 'Karla',
TO_DATE('19-03-2002','dd-mm-yyyy'),'feminin','roman','0753212342', 'karlateo@gmail.com',
10);
COMMIT;
CREATE OR REPLACE PROCEDURE Ex9(ume_student student.nume%type)
AS
    TYPE tabel_index IS TABLE OF curs.denumire%type INDEX BY PLS_INTEGER;
    den_curs tabel_index;
    TYPE tabel_index_studenti IS TABLE OF student%rowtype INDEX BY PLS_INTEGER;
    studenti tabel_index_studenti;

    NO_DATA_FOUND1 EXCEPTION;
    NO_DATA_FOUND2 EXCEPTION;
    TOO_MANY_ROWS1 EXCEPTION;

BEGIN
    SELECT *
    BULK COLLECT INTO studenti
    FROM student
    WHERE UPPER(nume) = UPPER(ume_student);
    IF SQL%NOTFOUND THEN
        RAISE NO_DATA_FOUND1;
    END IF;

    IF studenti.count >= 2 THEN
        RAISE TOO_MANY_ROWS1;
    END IF;

    SELECT c.denumire
```

```
BULK COLLECT INTO den_curs
FROM curs c JOIN orar o ON (o.cod_curs = c.cod_curs)
JOIN sala s ON (o.cod_sala = s.cod_sala)
JOIN grupa g ON (o.cod_grupa = g.cod_grupa)
JOIN inscriere i ON (i.cod_grupa = g.cod_grupa)
JOIN student stud ON (stud.cod_student = i.cod_student)
WHERE UPPER(stud.num) = UPPER(num_student)
AND s.denumire LIKE '%i%';

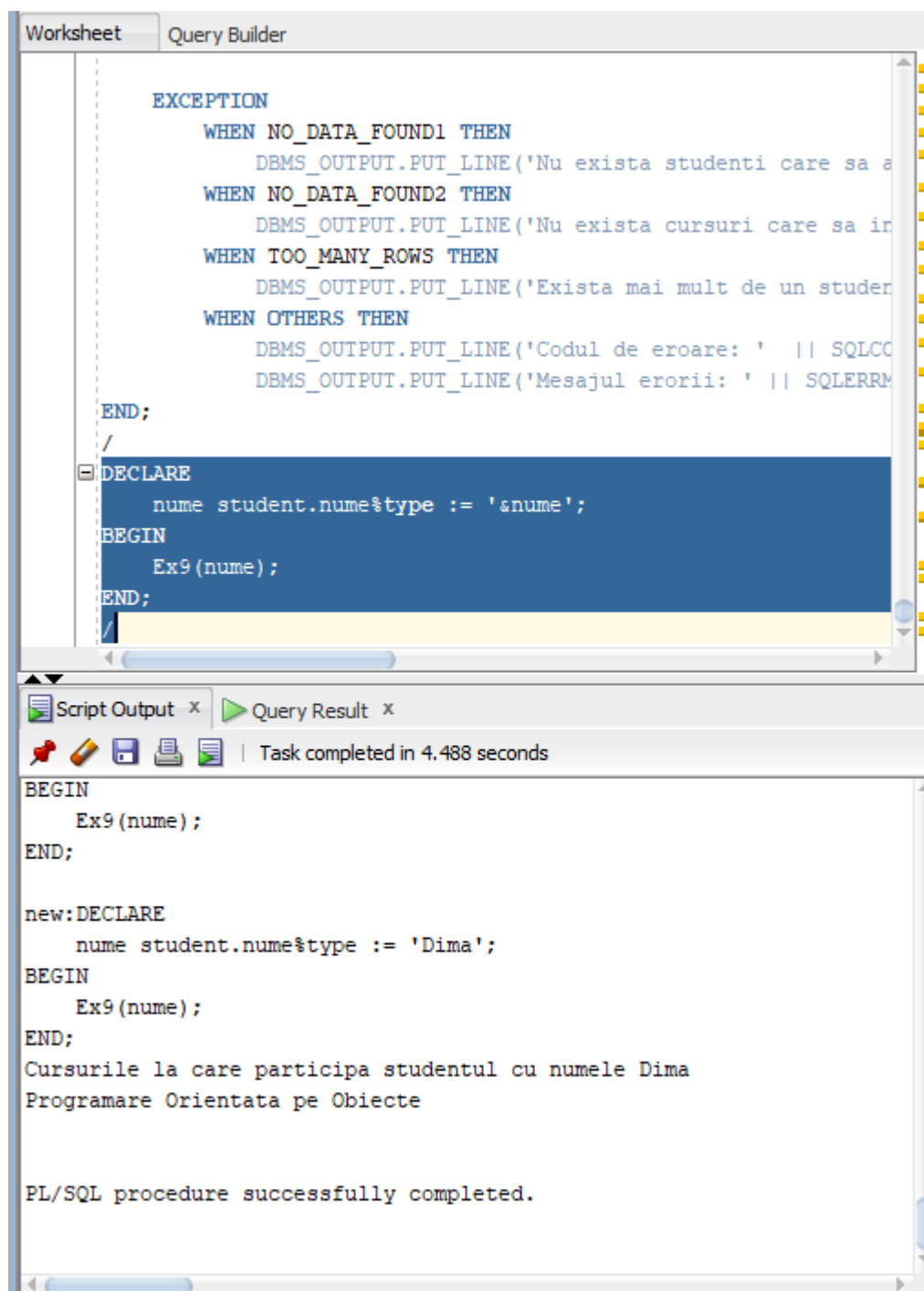
IF den_curs.count = 0 THEN
    RAISE NO_DATA_FOUND2;
END IF;

DBMS_OUTPUT.PUT_LINE('Cursurile la care participa studentul cu numele ' ||
num_student);
FOR i IN den_curs.FIRST..den_curs.LAST LOOP
    DBMS_OUTPUT.PUT_LINE(den_curs(i));
END LOOP;

EXCEPTION
    WHEN NO_DATA_FOUND1 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista studenti care sa aiba acest nume de
familie!');
    WHEN NO_DATA_FOUND2 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista cursuri care sa indeplineasca acele
conditii!');
    WHEN TOO_MANY_ROWS1 THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai mult de un student cu acel nume!');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' || SQLCODE);
        DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' || SQLERRM);
END;
/
DECLARE
    nume student.num%type := '&nume';
BEGIN
    Ex9(nume);
END;
/
```

OUTPUT:

-- nume = Dima



The screenshot displays the Oracle SQL Developer interface. The top pane, titled 'Query Builder', contains a PL/SQL script. The script includes an exception block with several 'WHEN' clauses for error handling, followed by a 'DECLARE' section, a 'BEGIN' block, and an 'END;' statement. The 'DECLARE' section defines a variable 'nume' of type 'student.nume%type' and assigns it the value 'Dima'. The 'BEGIN' block calls a procedure 'Ex9(nume)'. The bottom pane, titled 'Script Output', shows the execution results. It includes the same script code as the top pane, followed by the output of the 'Ex9(nume)' procedure, which lists the courses 'Cursurile la care participa studentul cu numele Dima' and 'Programare Orientata pe Obiecte'. The final line of the output is 'PL/SQL procedure successfully completed.'

```
EXCEPTION
    WHEN NO_DATA_FOUND1 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista studenti care sa a
    WHEN NO_DATA_FOUND2 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista cursuri care sa in
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai mult de un studen
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' || SQLCC
        DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' || SQLERRM
END;
/
DECLARE
    nume student.nume%type := '&nume';
BEGIN
    Ex9(nume);
END;
```

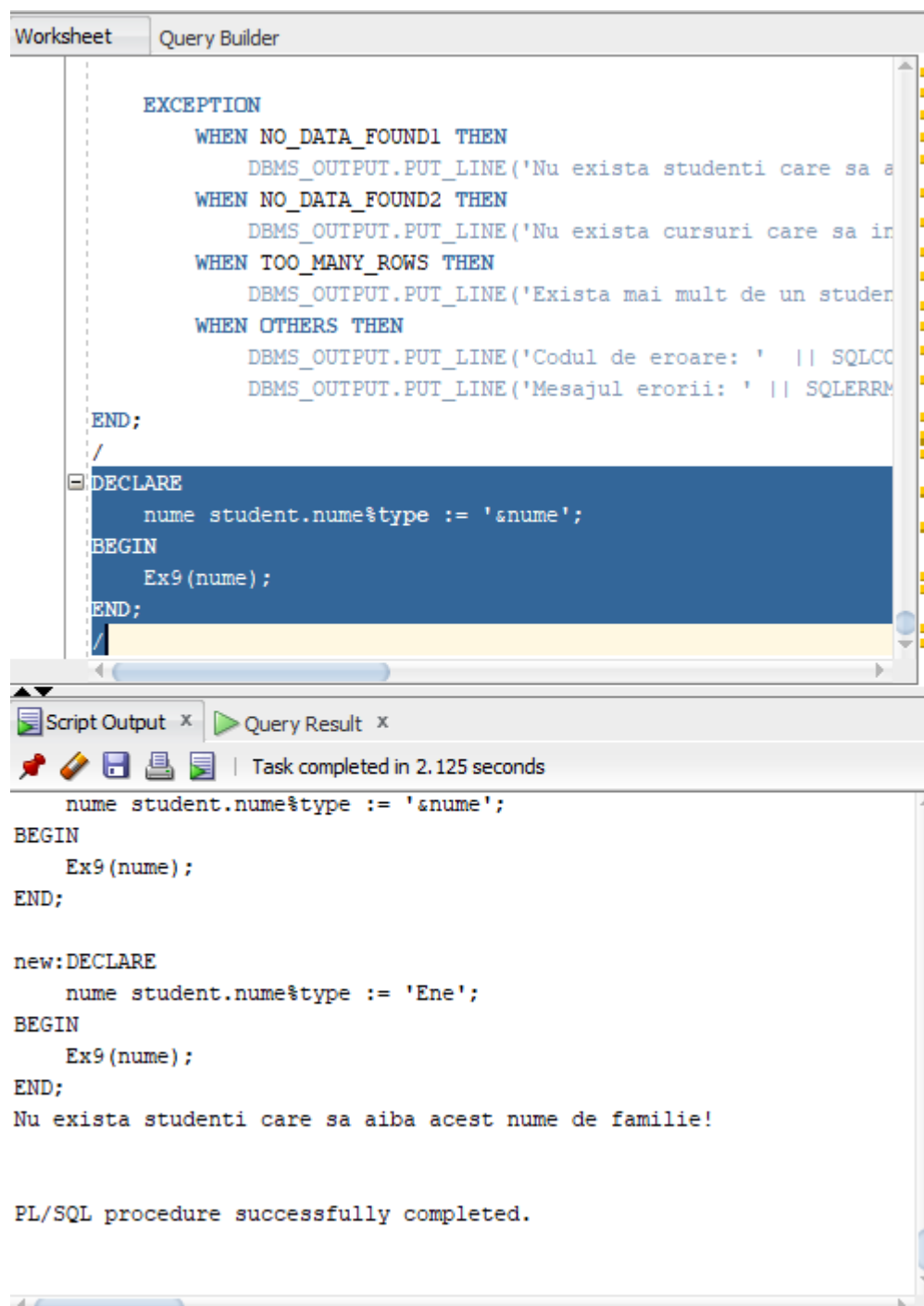
Script Output x Query Result x
Task completed in 4.488 seconds

```
BEGIN
    Ex9(nume);
END;

new:DECLARE
    nume student.nume%type := 'Dima';
BEGIN
    Ex9(nume);
END;
Cursurile la care participa studentul cu numele Dima
Programare Orientata pe Obiecte

PL/SQL procedure successfully completed.
```

-- nume = Ene



The screenshot shows a database query editor with two tabs: 'Worksheet' and 'Query Builder'. The 'Worksheet' tab is active, displaying a PL/SQL script. The script includes an exception block for handling errors, followed by a 'DECLARE' section, a 'BEGIN' block, and an 'END;' statement. The 'BEGIN' block contains a call to a procedure 'Ex9' with the parameter 'nume'. The 'DECLARE' section declares a variable 'nume' of type 'student.nume\$type' and assigns it the value 'Ene'. The 'BEGIN' block also contains a call to 'Ex9' with the parameter 'nume'. The 'END;' statement marks the end of the procedure. The 'Query Result' tab is also visible, showing the output of the script execution. The output includes the declaration of 'nume', the call to 'Ex9', and a message indicating that the PL/SQL procedure was successfully completed.

```
EXCEPTION
    WHEN NO_DATA_FOUND1 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista studenti care sa a
    WHEN NO_DATA_FOUND2 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista cursuri care sa in
    WHEN TOO_MANY_ROWS THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai mult de un studen
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' || SQLCO
        DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' || SQLERRM
END;
/
DECLARE
    nume student.nume$type := '&nume';
BEGIN
    Ex9 (nume);
END;
```

Script Output x Query Result x

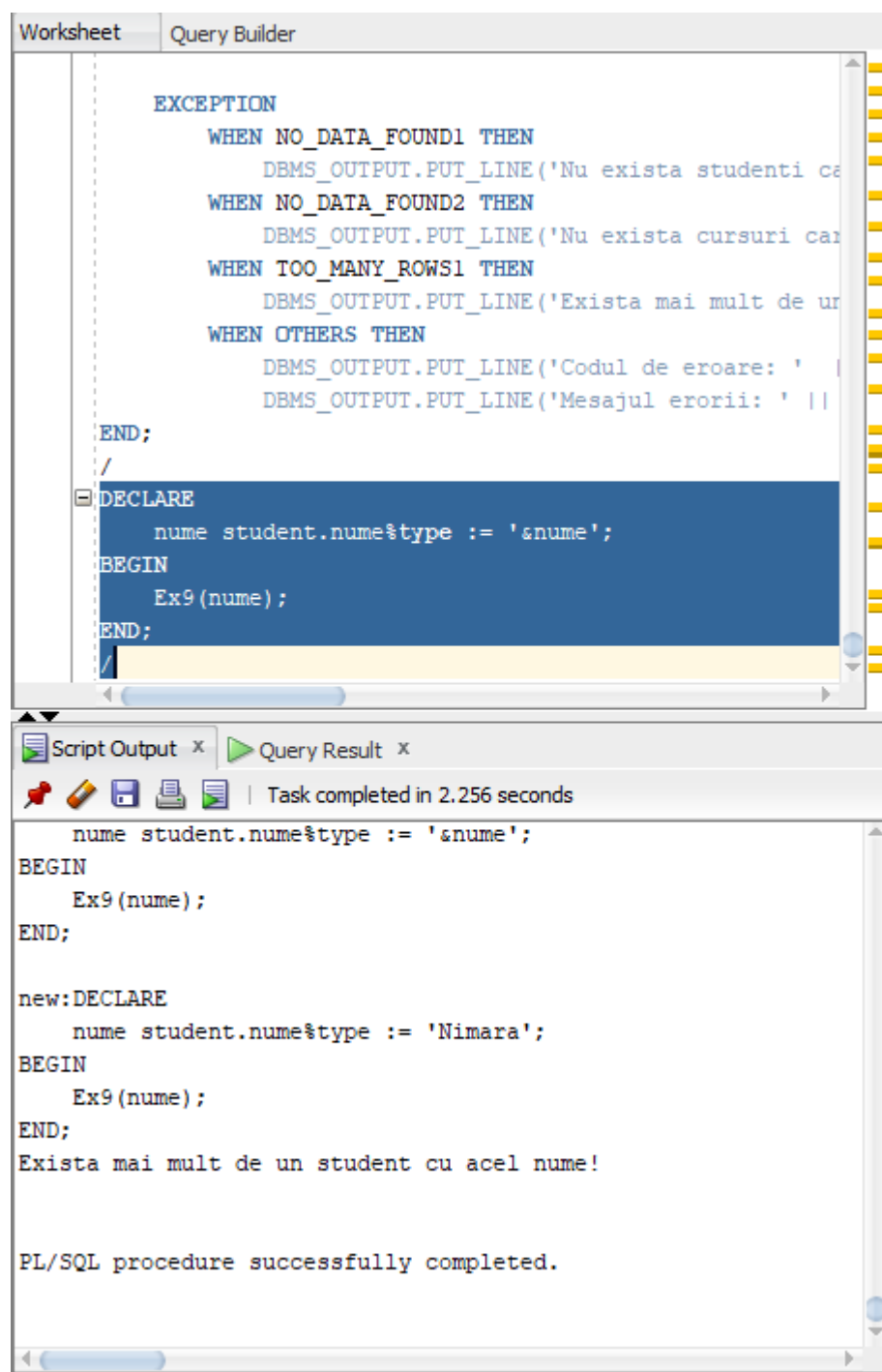
Task completed in 2.125 seconds

```
nume student.nume$type := '&nume';
BEGIN
    Ex9 (nume);
END;

new:DECLARE
    nume student.nume$type := 'Ene';
BEGIN
    Ex9 (nume);
END;
Nu exista studenti care sa aiba acest nume de familie!

PL/SQL procedure successfully completed.
```

-- nume = Nimara



The screenshot displays the Oracle SQL Developer interface. The top pane, titled 'Query Builder', contains a PL/SQL script. The script includes an exception block with several 'WHEN' clauses for error handling, followed by a 'DECLARE' section, a 'BEGIN' block, and an 'END;' statement. The 'DECLARE' section defines a variable 'nume' of type 'student.nume%type' and assigns it the value '&nume'. The 'BEGIN' block calls a procedure 'Ex9' with the variable 'nume' as an argument. The bottom pane, titled 'Script Output', shows the execution results. It displays the same PL/SQL script that was executed, followed by the message 'PL/SQL procedure successfully completed.' and a confirmation message 'Exista mai mult de un student cu acel nume!'.

```
EXCEPTION
    WHEN NO_DATA_FOUND1 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista studenti cu numele ' || nume);
    WHEN NO_DATA_FOUND2 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista cursuri cu numele ' || nume);
    WHEN TOO_MANY_ROWS1 THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai mult de un student cu numele ' || nume);
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' || SQLCODE);
        DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' || SQLERRM);
END;
/
DECLARE
    nume student.nume%type := '&nume';
BEGIN
    Ex9 (nume);
END;
```

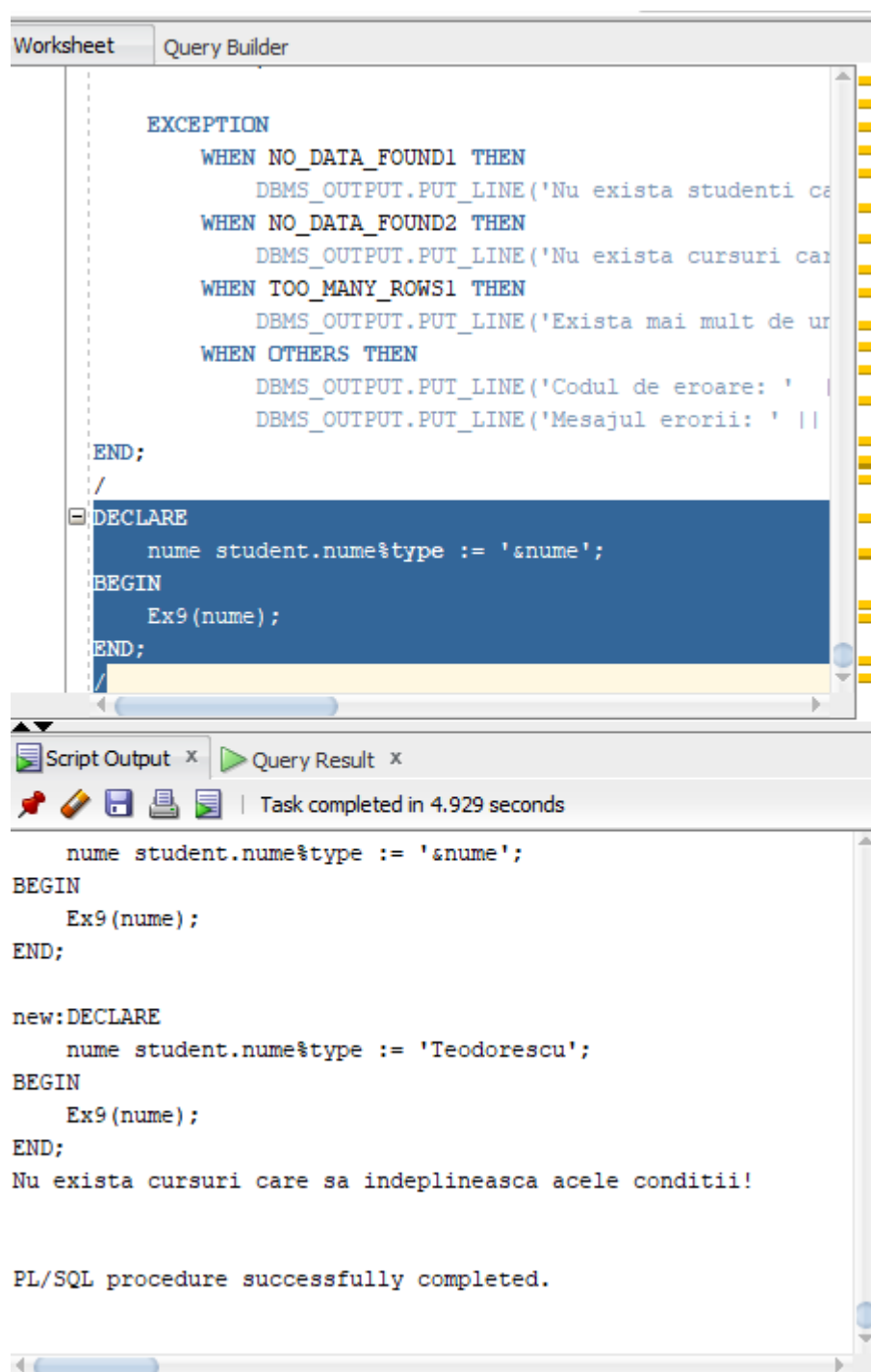
Task completed in 2.256 seconds

```
nume student.nume%type := '&nume';
BEGIN
    Ex9 (nume);
END;

new:DECLARE
    nume student.nume%type := 'Nimara';
BEGIN
    Ex9 (nume);
END;
Exista mai mult de un student cu acel nume!

PL/SQL procedure successfully completed.
```

-- nume = Teodorescu



The screenshot shows a database query builder window with two tabs: 'Worksheet' and 'Query Builder'. The 'Query Builder' tab is active, displaying a PL/SQL script. The script includes an exception block with four conditions: NO_DATA_FOUND1, NO_DATA_FOUND2, TOO_MANY_ROWS1, and OTHERS. Each condition has a corresponding message to be displayed using DBMS_OUTPUT.PUT_LINE. The script ends with an END; statement. Below the exception block, there is a DECLARE section for a variable 'nume' of type 'student.nume\$type', followed by a BEGIN section where the variable is assigned the value 'Teodorescu' and a procedure 'Ex9' is called. The script concludes with an END; statement. The 'Worksheet' tab is visible in the background, showing the same script. Below the query builder, there is a 'Script Output' tab and a 'Query Result' tab. The 'Script Output' tab is active, displaying the execution results. The results show the variable 'nume' assigned the value 'Teodorescu', the procedure 'Ex9' being called, and the message 'Nu exista cursuri care sa indeplineasca acele conditii!' being displayed. The final message is 'PL/SQL procedure successfully completed.'.

```
EXCEPTION
    WHEN NO_DATA_FOUND1 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista studenti ca');
    WHEN NO_DATA_FOUND2 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista cursuri ca');
    WHEN TOO_MANY_ROWS1 THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai mult de ur');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' ||
        DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' ||

END;
/

DECLARE
    nume student.nume$type := '&nume';
BEGIN
    Ex9 (nume);
END;
```

Script Output x Query Result x

Task completed in 4.929 seconds

```
nume student.nume$type := '&nume';
BEGIN
    Ex9 (nume);
END;

new:DECLARE
    nume student.nume$type := 'Teodorescu';
BEGIN
    Ex9 (nume);
END;
Nu exista cursuri care sa indeplineasca acele conditii!

PL/SQL procedure successfully completed.
```

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

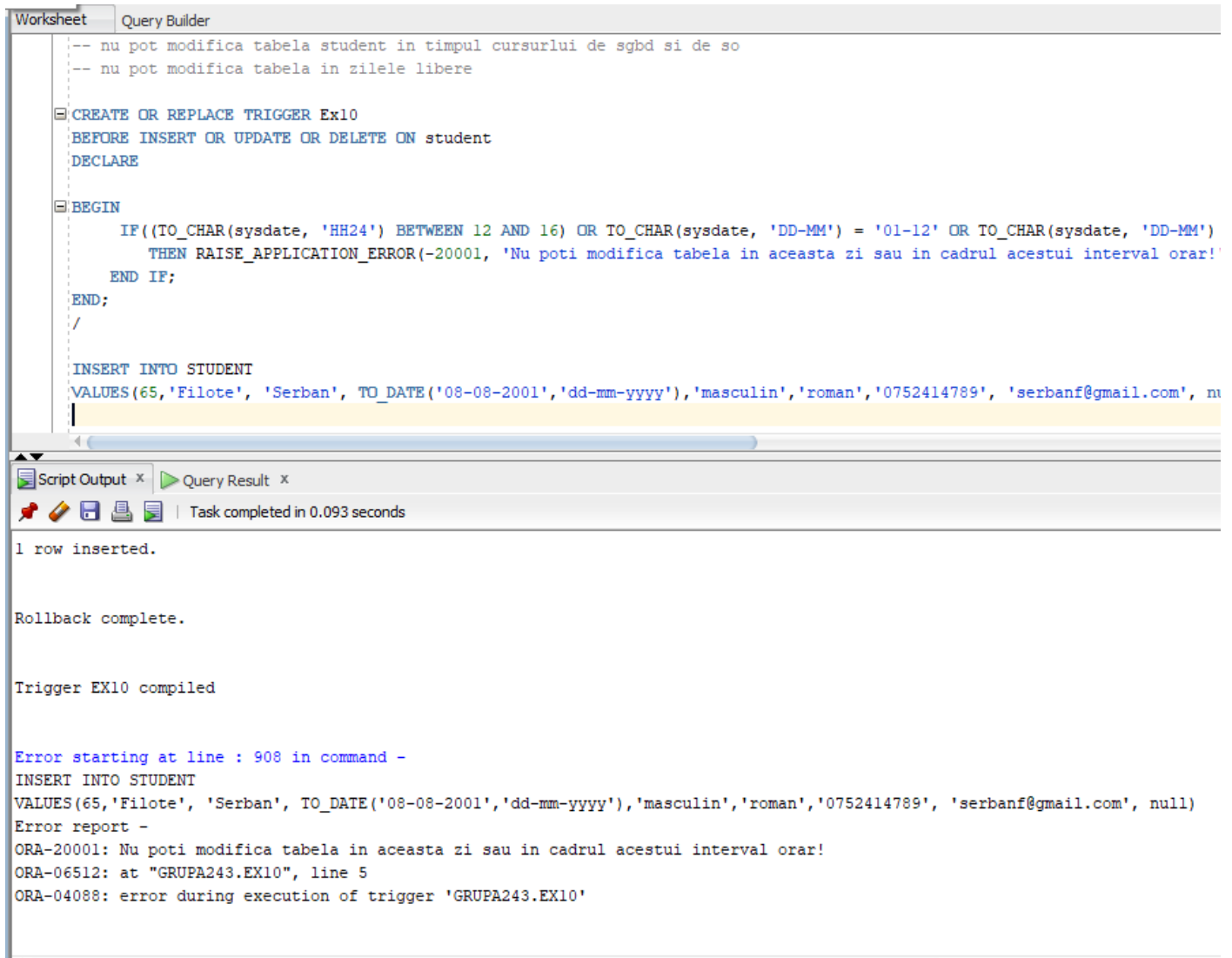
```
-- ex 10
-- voi realiza un trigger care se va declansa atunci cand vreau sa modific tabela
contract cu conditiile
-- nu pot modifica tabela student in timpul cursului de sgbd si de so
-- nu pot modifica tabela in zilele libere

CREATE OR REPLACE TRIGGER Ex10
BEFORE INSERT OR UPDATE OR DELETE ON student
DECLARE

BEGIN
    IF((TO_CHAR(sysdate, 'HH24') BETWEEN 12 AND 16) OR TO_CHAR(sysdate, 'DD-MM') =
'01-12' OR TO_CHAR(sysdate, 'DD-MM') = '25-12' OR TO_CHAR(sysdate, 'DD-MM') = '01-01')
        THEN RAISE_APPLICATION_ERROR(-20001, 'Nu poti modifica tabela in aceasta zi sau
in cadrul acestui interval orar!');
    END IF;
END;
/

INSERT INTO STUDENT
VALUES(65, 'Filote', 'Serban',
TO_DATE('08-08-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0752414789', 'serbanf@gmail.com',
null);
```


Sisteme de Gestiune a Bazelor de Date
Anul II - Seria 24



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Query Builder', contains a PL/SQL script. The script starts with two comments: '-- nu pot modifica tabela student in timpul cursului de sgbd si de so' and '-- nu pot modifica tabela in zilele libere'. It then defines a trigger named 'Ex10' that fires 'BEFORE INSERT OR UPDATE OR DELETE ON student'. The trigger body begins with 'BEGIN' and contains an 'IF' statement that checks if the current time is between 12 and 16, or if the date is '01-12'. If true, it raises an application error with message 'Nu poti modifica tabela in aceasta zi sau in cadrul acestui interval orar!'. The script ends with 'END;' and a slash. Below the trigger definition, an 'INSERT INTO STUDENT' statement is shown, with values: (65, 'Filote', 'Serban', TO_DATE('08-08-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0752414789', 'serbanf@gmail.com', null). The bottom pane, titled 'Script Output', shows the execution results. It indicates that the task completed in 0.093 seconds. The output shows '1 row inserted.', 'Rollback complete.', and 'Trigger EX10 compiled'. An error message follows: 'Error starting at line : 908 in command - INSERT INTO STUDENT VALUES(65, 'Filote', 'Serban', TO_DATE('08-08-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0752414789', 'serbanf@gmail.com', null)'. The error report shows 'ORA-20001: Nu poti modifica tabela in aceasta zi sau in cadrul acestui interval orar!', 'ORA-06512: at "GRUPA243.EX10", line 5', and 'ORA-04088: error during execution of trigger 'GRUPA243.EX10''.

```
-- nu pot modifica tabela student in timpul cursului de sgbd si de so
-- nu pot modifica tabela in zilele libere

CREATE OR REPLACE TRIGGER Ex10
BEFORE INSERT OR UPDATE OR DELETE ON student
DECLARE

BEGIN
    IF((TO_CHAR(sysdate, 'HH24') BETWEEN 12 AND 16) OR TO_CHAR(sysdate, 'DD-MM') = '01-12' OR TO_CHAR(sysdate, 'DD-MM')
        THEN RAISE_APPLICATION_ERROR(-20001, 'Nu poti modifica tabela in aceasta zi sau in cadrul acestui interval orar!')
    END IF;
END;
/

INSERT INTO STUDENT
VALUES(65, 'Filote', 'Serban', TO_DATE('08-08-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0752414789', 'serbanf@gmail.com', null)
```

Script Output x Query Result x

Task completed in 0.093 seconds

1 row inserted.

Rollback complete.

Trigger EX10 compiled

Error starting at line : 908 in command -
INSERT INTO STUDENT
VALUES(65, 'Filote', 'Serban', TO_DATE('08-08-2001', 'dd-mm-yyyy'), 'masculin', 'roman', '0752414789', 'serbanf@gmail.com', null)
Error report -
ORA-20001: Nu poti modifica tabela in aceasta zi sau in cadrul acestui interval orar!
ORA-06512: at "GRUPA243.EX10", line 5
ORA-04088: error during execution of trigger 'GRUPA243.EX10'

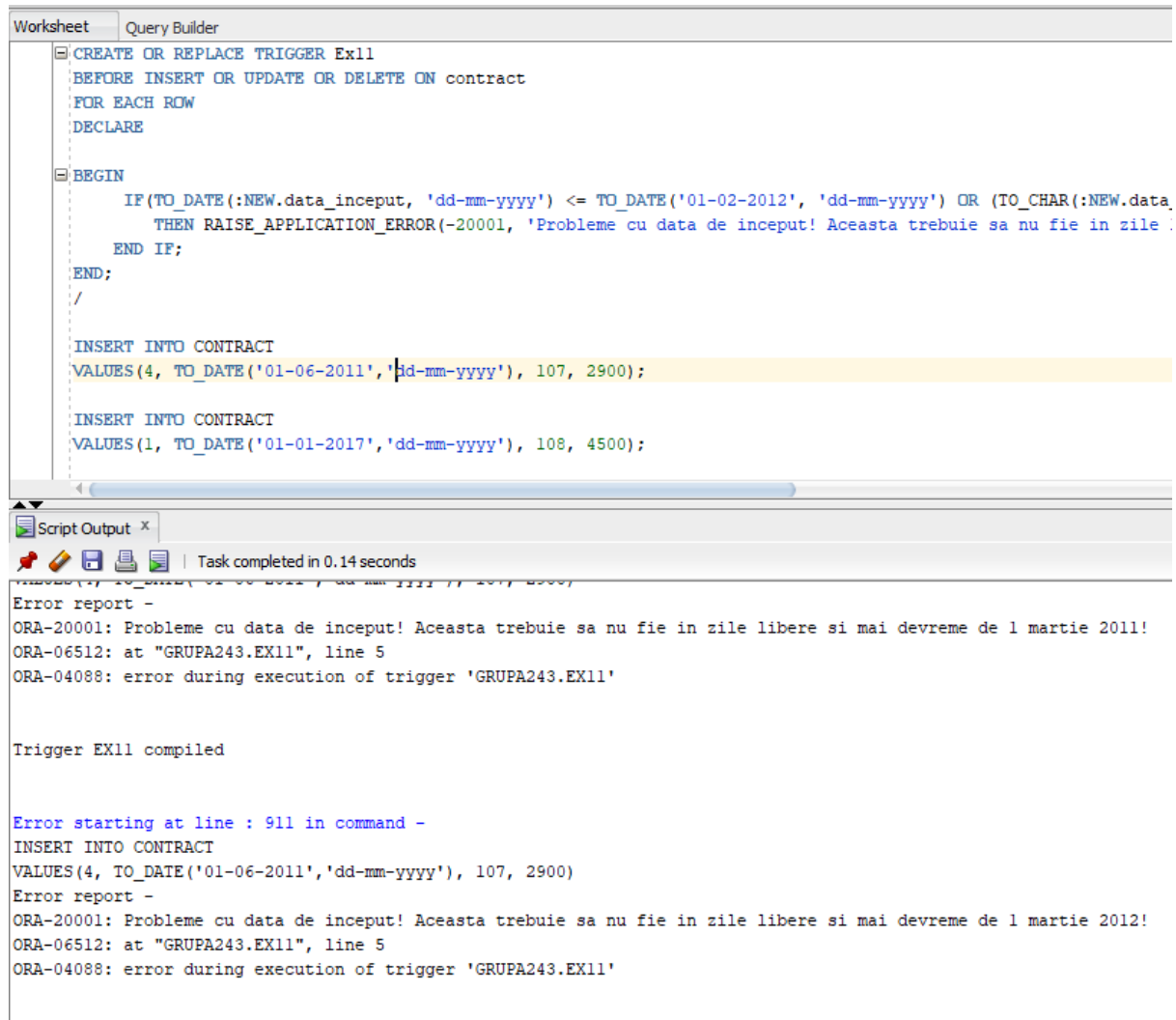
11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

```
-- ex 11
-- voi realiza un trigger care se va declansa atunci cand vreau sa modific tabela
contract cu conditiile
-- nu pot introduce date de inceput ale contractului mai vechi de 1 martie 2012
-- nu pot incepe contractul cu un profesor intr-o zi libera

CREATE OR REPLACE TRIGGER Ex11
BEFORE INSERT OR UPDATE OR DELETE ON contract
FOR EACH ROW
DECLARE

BEGIN
    IF(TO_DATE(:NEW.data_inceput, 'dd-mm-yyyy') <= TO_DATE('01-02-2012', 'dd-mm-yyyy')
OR (TO_CHAR(:NEW.data_inceput, 'DD-MM') = '30-11' OR TO_CHAR(:NEW.data_inceput, 'DD-MM')
= '01-12' OR TO_CHAR(:NEW.data_inceput, 'DD-MM') = '25-12' OR TO_CHAR(:NEW.data_inceput,
'DD-MM') = '01-01'))
        THEN RAISE_APPLICATION_ERROR(-20001, 'Probleme cu data de inceput! Aceasta
trebuie sa nu fie in zile libere si mai devreme de 1 martie 2012!');
    END IF;
END;
/
```

-- cand incerc sa inserez cu data_inceput in anul 2011



The screenshot shows a SQL Query Builder window with a worksheet titled 'Query Builder'. The SQL code defines a trigger named 'EX11' that fires before insert, update, or delete on the 'contract' table. The trigger logic checks if the 'data_inceput' is less than or equal to '01-02-2012'. If so, it raises an application error with message 'Probleme cu data de inceput! Aceasta trebuie sa nu fie in zile libere si mai devreme de 1 martie 2011!'. The trigger is then tested with two insert statements: one for '01-06-2011' (which fails) and one for '01-01-2017' (which succeeds).

```
CREATE OR REPLACE TRIGGER Ex11
BEFORE INSERT OR UPDATE OR DELETE ON contract
FOR EACH ROW
DECLARE

BEGIN
    IF (TO_DATE(:NEW.data_inceput, 'dd-mm-yyyy') <= TO_DATE('01-02-2012', 'dd-mm-yyyy') OR (TO_CHAR(:NEW.data_inceput, 'dd-mm-yyyy') <= '01-02-2012')) THEN
        RAISE_APPLICATION_ERROR(-20001, 'Probleme cu data de inceput! Aceasta trebuie sa nu fie in zile libere si mai devreme de 1 martie 2011!');
    END IF;
END;
/

INSERT INTO CONTRACT
VALUES(4, TO_DATE('01-06-2011', 'dd-mm-yyyy'), 107, 2900);

INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-01-2017', 'dd-mm-yyyy'), 108, 4500);
```

Script Output x

Task completed in 0.14 seconds

Error report -

ORA-20001: Probleme cu data de inceput! Aceasta trebuie sa nu fie in zile libere si mai devreme de 1 martie 2011!

ORA-06512: at "GRUPA243.EX11", line 5

ORA-04088: error during execution of trigger 'GRUPA243.EX11'

Trigger EX11 compiled

Error starting at line : 911 in command -

INSERT INTO CONTRACT

VALUES(4, TO_DATE('01-06-2011', 'dd-mm-yyyy'), 107, 2900)

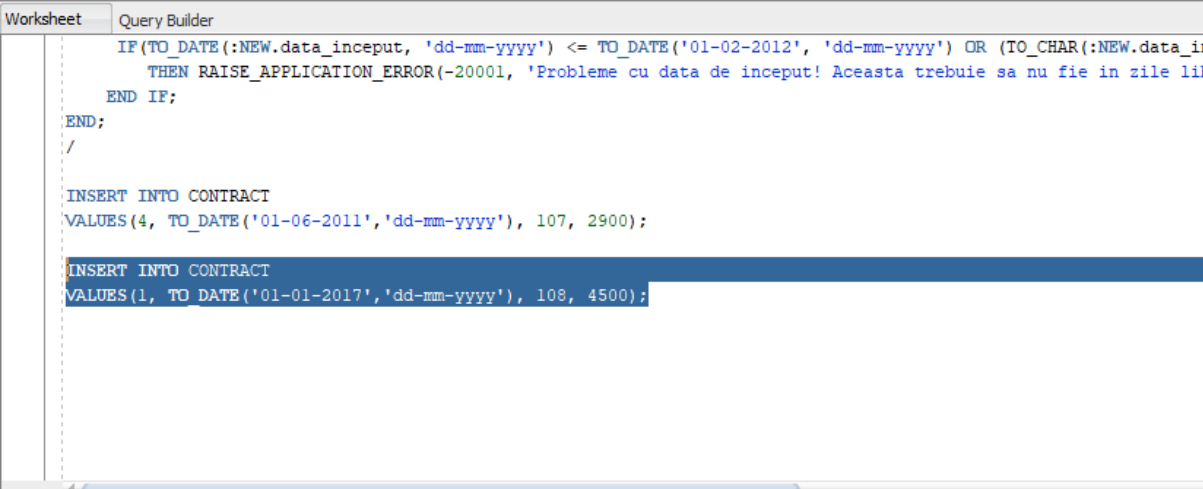
Error report -

ORA-20001: Probleme cu data de inceput! Aceasta trebuie sa nu fie in zile libere si mai devreme de 1 martie 2012!

ORA-06512: at "GRUPA243.EX11", line 5

ORA-04088: error during execution of trigger 'GRUPA243.EX11'

-- cand incerc sa inserez intr-o zi libera



The screenshot shows a SQL Query Builder window with a script containing an IF statement and two INSERT INTO CONTRACT statements. The first INSERT statement is highlighted in blue. Below the script, the Script Output window shows the execution results, including error messages for the first INSERT statement.

```
IF (TO_DATE(:NEW.data_inceput, 'dd-mm-yyyy') <= TO_DATE('01-02-2012', 'dd-mm-yyyy') OR (TO_CHAR(:NEW.data_inceput, 'dd-mm-yyyy') <= TO_CHAR('01-02-2012', 'dd-mm-yyyy')) THEN RAISE_APPLICATION_ERROR(-20001, 'Probleme cu data de inceput! Aceasta trebuie sa nu fie in zile libere si mai devreme de 1 martie 2012!') END IF;

END;

/

INSERT INTO CONTRACT
VALUES(4, TO_DATE('01-06-2011', 'dd-mm-yyyy'), 107, 2900);

INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-01-2017', 'dd-mm-yyyy'), 108, 4500);
```

Script Output x

Task completed in 0.133 seconds

Error starting at line : 911 in command -
INSERT INTO CONTRACT
VALUES(4, TO_DATE('01-06-2011', 'dd-mm-yyyy'), 107, 2900)
Error report -
ORA-20001: Probleme cu data de inceput! Aceasta trebuie sa nu fie in zile libere si mai devreme de 1 martie 2012!
ORA-06512: at "GRUPA243.EX11", line 5
ORA-04088: error during execution of trigger 'GRUPA243.EX11'

Error starting at line : 914 in command -
INSERT INTO CONTRACT
VALUES(1, TO_DATE('01-01-2017', 'dd-mm-yyyy'), 108, 4500)
Error report -
ORA-20001: Probleme cu data de inceput! Aceasta trebuie sa nu fie in zile libere si mai devreme de 1 martie 2012!
ORA-06512: at "GRUPA243.EX11", line 5
ORA-04088: error during execution of trigger 'GRUPA243.EX11'

```
-- ex 11 trigger pentru mutating error
-- voi realiza un trigger care se va declasa in momentul in care voi incerca sa atribui
```

```
cuiva un salariu de 3 ori mai mare decat cel mai mic salariu
desc contract;
CREATE OR REPLACE TRIGGER salariu_contract
  FOR UPDATE OR INSERT ON contract
  COMPOUND TRIGGER
  TYPE r_contract IS RECORD (
    cod_facultate contract.cod_facultate%type,
    cod_profesor contract.cod_profesor%type,
    salariu contract.salariu%type,
    data_inceput contract.data_inceput%type
  );
  TYPE t_contract IS TABLE OF r_contract INDEX BY PLS_INTEGER;

  tabel t_contract;
  counter NUMBER := 0;
  AFTER EACH ROW IS
  BEGIN
    tabel(counter).cod_profesor := :OLD.cod_profesor;
    tabel(counter).cod_facultate := :OLD.cod_facultate;
    tabel(counter).salariu := :NEW.salariu;
    tabel(counter).data_inceput := :OLD.data_inceput;
  END AFTER EACH ROW;

  AFTER STATEMENT IS
    salariu_maxim NUMBER;
  BEGIN
    SELECT MIN(salariu) * 3
    INTO salariu_maxim
    FROM contract;

    FOR ind IN tabel.first..tabel.last LOOP
      IF salariu_maxim < tabel(ind).salariu
      THEN
        UPDATE contract
        SET salariu = salariu_maxim
        WHERE cod_facultate = tabel(ind).cod_facultate AND cod_profesor =
tabel(ind).cod_profesor;
      END IF;
    END LOOP;
  END AFTER STATEMENT;
END;
/
COMMIT;
SELECT * from contract;

UPDATE contract
SET salariu = 12000
WHERE cod_profesor = 107 AND TO_DATE(data_inceput, 'dd-mm-yyyy') LIKE
TO_DATE('01-06-2011', 'dd-mm-yyyy');
```

– inainte de update

Sisteme de Gestiune a Bazelor de Date
Anul II - Seria 24

Worksheet		Query Builder		
		<pre>END AFTER STATEMENT; END; / COMMIT; SELECT * from contract; UPDATE contract</pre>		
Script Output x		Query Result x		
		All Rows Fetched: 25 in 0.005 seconds		
	COD_FACULTATE	DATA_INCEPUT	COD_PROFESOR	SALARIU
1	4	01-JUN-16	104	5400
2	5	02-AUG-15	102	5700
3	5	13-JUL-16	104	3600
4	2	19-JUN-17	102	3800
5	1	01-JAN-17	108	4500
6	1	01-AUG-18	111	5400
7	2	01-JUL-18	108	5200
8	3	01-JUN-17	107	5400
9	1	01-JUN-17	109	4500
10	1	01-JUN-17	110	4200
11	1	01-JUN-17	106	4700
12	1	01-JUN-19	107	4700
13	4	01-JUN-16	108	5400
14	5	02-AUG-15	111	5700
15	2	02-AUG-15	111	5700
16	5	19-JUN-17	108	3800
17	1	01-AUG-18	113	5400
18	1	01-AUG-18	100	5400
19	2	01-JUL-18	101	5200
20	3	01-JUN-17	100	5400
21	1	01-JUN-17	102	4500
22	1	01-JUN-17	103	4200
23	1	01-JUN-17	104	4700
24	1	01-JUN-19	105	4700
25	4	01-JUN-11	107	2900

END AFTER STATEMENT;

END;

/

COMMIT;

SELECT * from contract;

UPDATE contract

SET salariu = 10000

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

Sisteme de Gestiune a Bazelor de Date
Anul II - Seria 24

```
-- ex 12
-- am creat un trigger care se declanseaza de fiecare data cand se executa operatii ldd
-- salvez modificarile in tabela modificari_daria

CREATE TABLE modificari_daria (
    utilizator VARCHAR2(50),
    nume_bd VARCHAR2(50),
    eveniment VARCHAR2(50),
    nume_obiect VARCHAR2(50),
    tip_obiect VARCHAR2(50),
    data_realizarii DATE
);

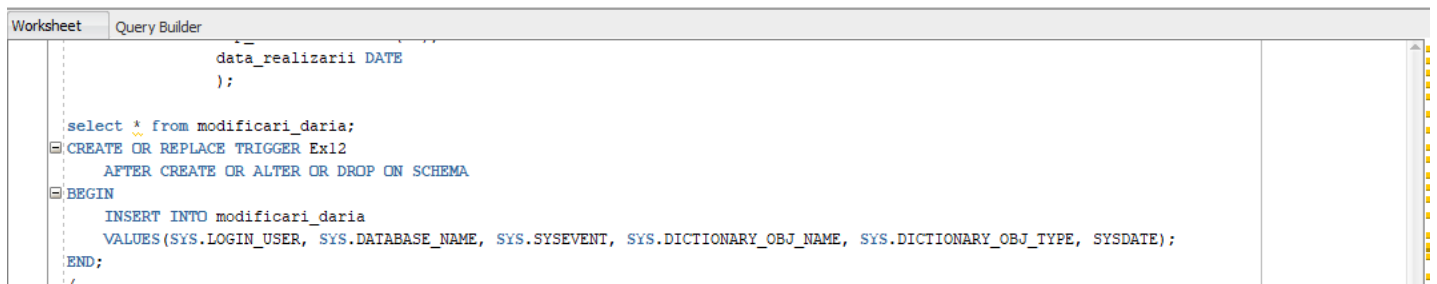
select * from modificari_daria;
CREATE OR REPLACE TRIGGER Ex12
    AFTER CREATE OR ALTER OR DROP ON SCHEMA
BEGIN
    INSERT INTO modificari_daria
    VALUES(SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT, SYS.DICTIONARY_OBJ_NAME,
    SYS.DICTIONARY_OBJ_TYPE, SYSDATE);
END;
/

CREATE TABLE FRUCTE ( cod_fruct NUMBER(6,2) CONSTRAINT codfruct_pk PRIMARY KEY,
    denumire VARCHAR2(60)
);

ALTER TABLE FRUCTE
ADD ( sezon VARCHAR2(60));

ALTER TABLE FRUCTE
DROP COLUMN denumire;

DROP TABLE FRUCTE;
SELECT *
FROM modificari_daria;
```



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

```
-- 13 pachet
CREATE OR REPLACE PACKAGE proiect_daria AS
    PROCEDURE Ex6(denumirefac FACULTATE.denumire%TYPE);
    PROCEDURE Ex7(tip_examen examen.forma%TYPE, nota_examen promoveaza.nota%TYPE);
    FUNCTION Ex8(cod profesor.cod_profesor%TYPE) RETURN NUMBER;
    PROCEDURE Ex9(ume_student student.ume%type);
END proiect_daria;
/

CREATE OR REPLACE PACKAGE BODY proiect_daria
AS
    -- pentru o facultate al carei nume este dat, pentru fiecare amfiteatru, afisati
    -- studentii ai caror grupe desfasoara ore in amfiteatrul curent sau afisati 'nu exista'
    -- daca pentru grupa x nu exista studenti care se aiba ore in amfiteatrul y

    PROCEDURE Ex6 (denumirefac FACULTATE.denumire%TYPE)
AS
    TYPE tabl_idx IS TABLE OF sala%rowtype INDEX BY PLS_INTEGER;
    amfiteatre tabl_idx;

    TYPE tip_lista_nested IS TABLE OF grupa%rowtype;
    grupe tip_lista_nested := tip_lista_nested();

    TYPE tabl_index IS TABLE OF VARCHAR(200) INDEX BY PLS_INTEGER;
    v_ume tabl_index;

    numar NUMBER(6);
BEGIN
    SELECT *
    BULK COLLECT INTO amfiteatre
    FROM sala;

    SELECT COUNT(*)
    INTO numar
    FROM grupa g, sectie sect, serie ser, facultate f
    WHERE g.cod_serie = ser.cod_serie and ser.cod_sectie = sect.cod_sectie and
    sect.cod_facultate = f.cod_facultate
    AND UPPER(f.denumire) LIKE UPPER(denumirefac) AND ROWNUM <= 1000;

    grupe.extend(numar + 1);

    SELECT g.cod_grupa, g.denumire, g.cod_serie
    BULK COLLECT INTO grupe
    FROM grupa g, sectie sect, serie ser, facultate f
    WHERE g.cod_serie = ser.cod_serie and ser.cod_sectie = sect.cod_sectie and
    sect.cod_facultate = f.cod_facultate
    AND UPPER(f.denumire) LIKE UPPER('Facultatea de Matematica%');

    FOR i IN amfiteatre.first..amfiteatre.last LOOP
        DBMS_OUTPUT.PUT_LINE('Amfiteatrul ' || amfiteatre(i).denumire);
        DBMS_OUTPUT.PUT_LINE('-----');

        FOR j IN grupe.first..grupe.last LOOP
            DBMS_OUTPUT.PUT_LINE('Grupa ' || grupe(j).denumire);
```

```
        DBMS_OUTPUT.PUT_LINE('-----');
        SELECT s.num || ' ' || s.prenume
        BULK COLLECT INTO v_num
        FROM student s, orar o, inscriere ins
        WHERE s.cod_student = ins.cod_student and ins.cod_grupa = grupe(j).cod_grupa
and o.cod_grupa = grupe(j).cod_grupa and amfiteatre(i).cod_sala = o.cod_sala;
        IF v_num.count > 0 THEN
            numar := 0;
            FOR k in v_num.first..v_num.last LOOP
                numar := numar + 1;
                DBMS_OUTPUT.PUT_LINE(numar || '. ' || v_num(k));
            END LOOP;
        END IF;
        IF v_num.count = 0
            THEN DBMS_OUTPUT.PUT_LINE('Nu exista');
        END IF;
        DBMS_OUTPUT.NEW_LINE;
        DBMS_OUTPUT.NEW_LINE;
    END LOOP;
END LOOP;

END Ex6;

-- pentru examenele care au ca forma 'Examen Scris'
-- afisati studentii care le-au promovat cu nota mai mare de 7
PROCEDURE Ex7 (tip_examen examen.forma%TYPE, nota_examen promoveaza.nota%TYPE)
AS
    CURSOR cursuri (cod curs.cod_curs%TYPE) IS
        SELECT denumire
        FROM curs
        WHERE cod_curs = cod;

    CURSOR examene (tip_examen examen.forma%TYPE) IS
        SELECT cod_examen, cod_curs
        FROM examen
        WHERE UPPER(forma) LIKE UPPER(typ_examen);

    CURSOR studenti (cod examen.cod_examen%TYPE ) IS
        SELECT s.num || ' ' || s.prenume || ' a obtinut nota ' || p.nota as
result
        FROM promoveaza p, student s
        WHERE p.cod_student = s.cod_student AND p.cod_examen = cod AND p.nota >=
nota_examen;

    den curs.denumire%TYPE;
    cod curs.cod_curs%TYPE;
BEGIN
    FOR examen in examene(typ_examen) LOOP
        OPEN cursuri(examen.cod_curs);
        FETCH cursuri INTO den;
        DBMS_OUTPUT.PUT_LINE('Cursul ' || den);
        DBMS_OUTPUT.PUT_LINE('-----');
    CLOSE cursuri;
```

```
        FOR student IN studenti(examen.cod_examen) LOOP
            DBMS_OUTPUT.PUT_LINE(student.result);
        END LOOP;

        DBMS_OUTPUT.NEW_LINE;
        DBMS_OUTPUT.NEW_LINE;
    END LOOP;

END Ex7;

FUNCTION Ex8(cod profesor.cod_profesor%TYPE) RETURN NUMBER
IS
    nr_facultati NUMBER;
    TYPE tip_tabel IS TABLE OF contract%rowtype INDEX BY PLS_INTEGER;
    tabel tip_tabel;
    pren profesor.prenume%type;
    --exceptii
    NEGATIVE_NUMBER EXCEPTION;
    NO_DATA_FOUND1 EXCEPTION;
    NO_DATA_FOUND2 EXCEPTION;

BEGIN
    IF cod < 0 THEN -- codul profesorului nu e valid
        RAISE NEGATIVE_NUMBER;
    END IF;

    SELECT *
    BULK COLLECT INTO tabel
    FROM contract
    WHERE cod_profesor = cod;

    IF SQL%NOTFOUND THEN
        RAISE NO_DATA_FOUND1;
    END IF;

    SELECT prenume
    INTO pren
    FROM profesor
    WHERE cod_profesor = cod;

    SELECT COUNT(f.cod_facultate)
    INTO nr_facultati
    FROM facultate f
    JOIN contract c ON (c.cod_facultate = f.cod_facultate)
    WHERE c.cod_profesor = cod
    AND (SELECT COUNT(prenume) FROM profesor WHERE UPPER(prenume) LIKE UPPER(pren) ) >= 2
    AND cod_postal LIKE '%1%';

    IF nr_facultati = 0 THEN
        RAISE NO_DATA_FOUND2;
    ELSE RETURN nr_facultati;
    END IF;
```

```
EXCEPTION
    WHEN NEGATIVE_NUMBER THEN
        DBMS_OUTPUT.PUT_LINE('Codul profesorului nu poate sa fie negativ!');
        RETURN -1;
    WHEN NO_DATA_FOUND1 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista profesorul cu codul ' || cod || ' in tabela
contract!');
        RETURN -1;
    WHEN NO_DATA_FOUND2 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista facultati care sa indeplineasca acele
conditiile!');
        RETURN -1;
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' || SQLCODE);
        DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' || SQLERRM);
        RETURN -1;
END Ex8;

PROCEDURE Ex9(ume_student student.ume%type)
AS
    TYPE tabel_index IS TABLE OF curs.denumire%type INDEX BY PLS_INTEGER;
    den_curs tabel_index;
    TYPE tabel_index_studenti IS TABLE OF student%rowtype INDEX BY PLS_INTEGER;
    studenti tabel_index_studenti;

    NO_DATA_FOUND1 EXCEPTION;
    NO_DATA_FOUND2 EXCEPTION;
    TOO_MANY_ROWS1 EXCEPTION;

BEGIN
    SELECT *
    BULK COLLECT INTO studenti
    FROM student
    WHERE UPPER(ume) = UPPER(ume_student);
    IF SQL%NOTFOUND THEN
        RAISE NO_DATA_FOUND1;
    END IF;

    IF studenti.count >= 2 THEN
        RAISE TOO_MANY_ROWS1;
    END IF;

    SELECT c.denumire
    BULK COLLECT INTO den_curs
    FROM curs c JOIN orar o ON (o.cod_curs = c.cod_curs)
    JOIN sala s ON (o.cod_sala = s.cod_sala)
    JOIN grupa g ON (o.cod_grupa = g.cod_grupa)
    JOIN inscriere i ON (i.cod_grupa = g.cod_grupa)
    JOIN student stud ON (stud.cod_student = i.cod_student)
    WHERE UPPER(stud.ume) = UPPER(ume_student)
    AND s.denumire LIKE '%i%';

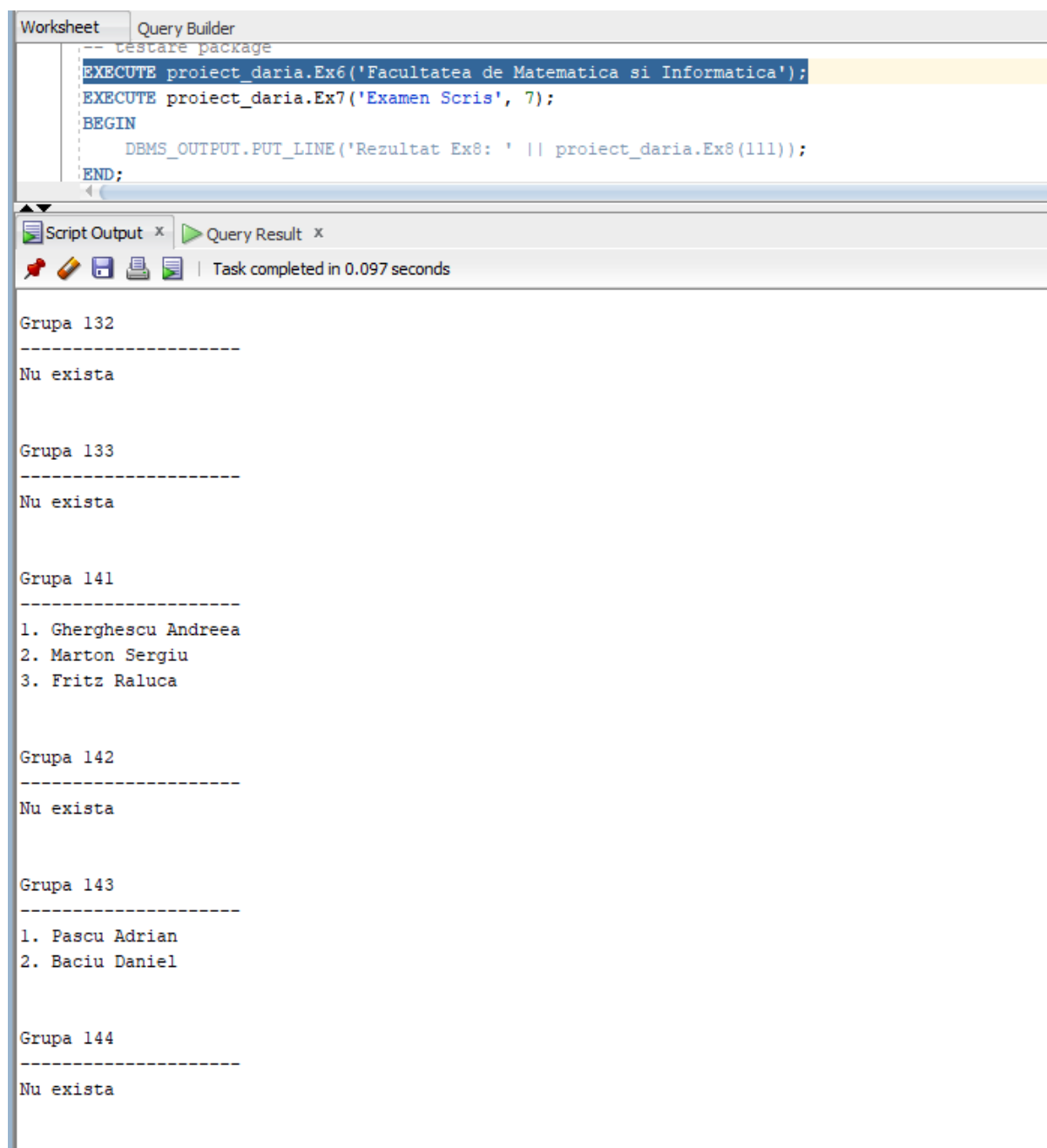
    IF den_curs.count = 0 THEN
        RAISE NO_DATA_FOUND2;
    END IF;
```

```
        DBMS_OUTPUT.PUT_LINE('Cursurile la care participa studentul cu numele ' ||
nume_student);
    FOR i IN den_curs.FIRST..den_curs.LAST LOOP
        DBMS_OUTPUT.PUT_LINE(den_curs(i));
    END LOOP;

    EXCEPTION
        WHEN NO_DATA_FOUND1 THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista studenti care sa aiba acest nume de
familie!');
        WHEN NO_DATA_FOUND2 THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista cursuri care sa indeplineasca acele
conditii!');
        WHEN TOO_MANY_ROWS1 THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai mult de un student cu acel nume!');
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Codul de eroare: ' || SQLCODE);
            DBMS_OUTPUT.PUT_LINE('Mesajul erorii: ' || SQLERRM);
    END Ex9;

END proiect_daria;
/

-- testare package
EXECUTE proiect_daria.Ex6('Facultatea de Matematica si Informatica');
EXECUTE proiect_daria.Ex7('Examen Scris', 7);
BEGIN
    DBMS_OUTPUT.PUT_LINE('Rezultat Ex8: ' || proiect_daria.Ex8(111));
END;
/
EXECUTE proiect_daria.Ex9('Dima');
```



The screenshot displays a database query tool interface. At the top, there are two tabs: "Worksheet" and "Query Builder". The "Query Builder" tab is active, showing a SQL script. The script is as follows:

```
-- testare package
EXECUTE proiect_daria.Ex6('Facultatea de Matematica si Informatica');
EXECUTE proiect_daria.Ex7('Examen Scris', 7);
BEGIN
    DBMS_OUTPUT.PUT_LINE('Rezultat Ex8: ' || proiect_daria.Ex8(111));
END;
```

Below the script, there is a status bar indicating "Task completed in 0.097 seconds". The main area of the tool displays the output of the script, which is a list of student names grouped by their group number. The output is as follows:

```
Grupa 132
-----
Nu exista

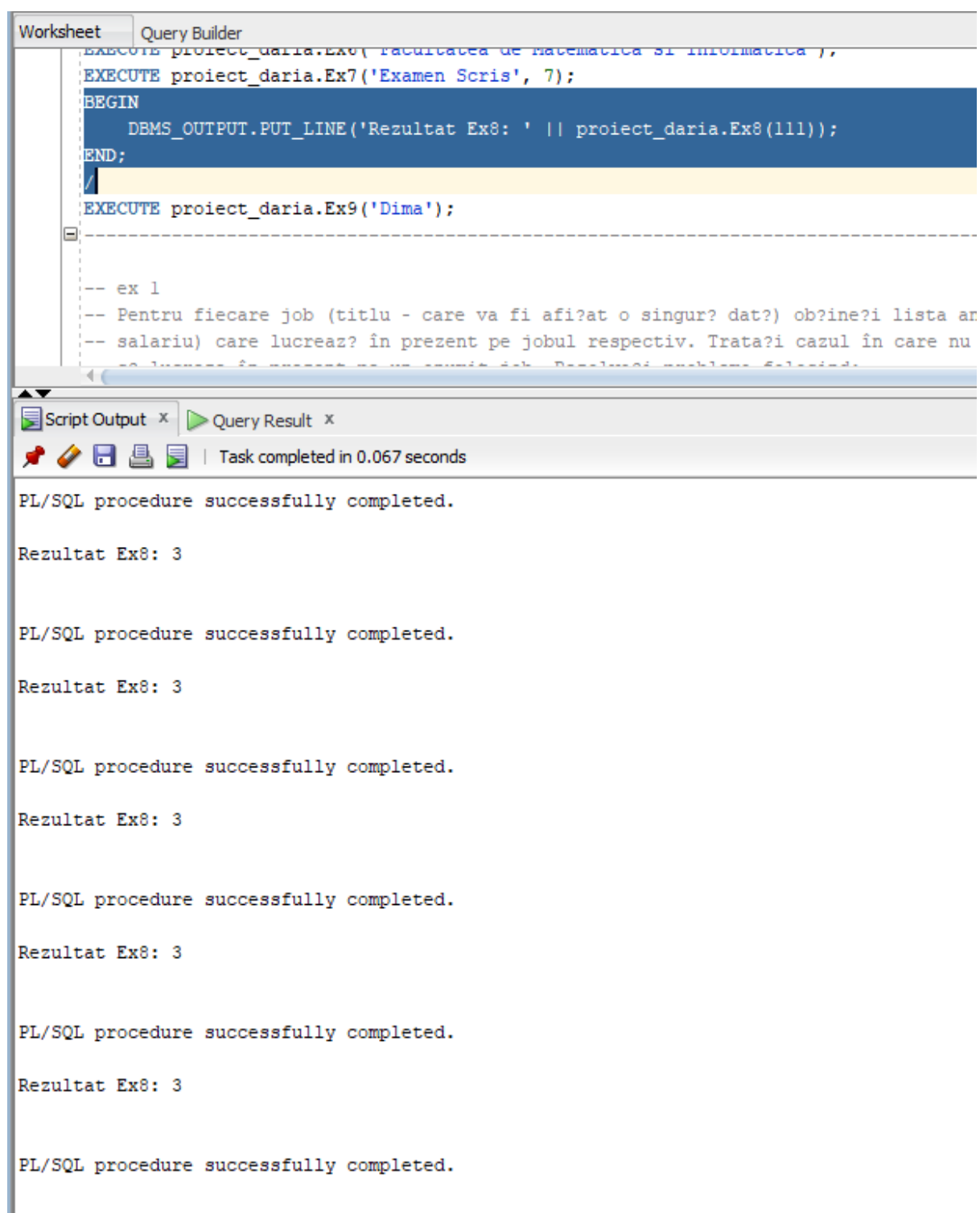
Grupa 133
-----
Nu exista

Grupa 141
-----
1. Gherghescu Andreea
2. Marton Sergiu
3. Fritz Raluca

Grupa 142
-----
Nu exista

Grupa 143
-----
1. Pascu Adrian
2. Baci Daniel

Grupa 144
-----
Nu exista
```



The screenshot shows a SQL IDE interface with two main panes. The top pane, titled 'Query Builder', contains a PL/SQL script. The script starts with a comment 'Facultatea de Matematica si Informatica', followed by 'EXECUTE proiect_daria.Ex7('Examen Scris', 7);'. Then, a block is defined with 'BEGIN', 'DBMS_OUTPUT.PUT_LINE('Rezultat Ex8: ' || proiect_daria.Ex8(111));', and 'END;'. This block is highlighted in blue. Below it, 'EXECUTE proiect_daria.Ex9('Dima');' is shown. The bottom pane, titled 'Script Output x' and 'Query Result x', shows the execution results. It displays 'PL/SQL procedure successfully completed.' followed by 'Rezultat Ex8: 3' five times, indicating the block was executed five times. The task completed in 0.067 seconds.

```
Worksheet Query Builder
EXECUTE proiect_daria.Ex7('Facultatea de Matematica si Informatica',
EXECUTE proiect_daria.Ex7('Examen Scris', 7);
BEGIN
    DBMS_OUTPUT.PUT_LINE('Rezultat Ex8: ' || proiect_daria.Ex8(111));
END;
/
EXECUTE proiect_daria.Ex9('Dima');

-- ex 1
-- Pentru fiecare job (titlu - care va fi afi?at o singur? dat?) ob?ine?i lista ar
-- salariu) care lucreaz? în prezent pe jobul respectiv. Trata?i cazul în care nu
-- lucreaz? în prezent pe un anumit job. Rezolva?i problema folosind:
```

Script Output x Query Result x

Task completed in 0.067 seconds

PL/SQL procedure successfully completed.

Rezultat Ex8: 3

PL/SQL procedure successfully completed.

Rezultat Ex8: 3

PL/SQL procedure successfully completed.

Rezultat Ex8: 3

PL/SQL procedure successfully completed.

Rezultat Ex8: 3

PL/SQL procedure successfully completed.

Rezultat Ex8: 3

PL/SQL procedure successfully completed.


```
-- testare package
EXECUTE proiect_daria.Ex6('Facultatea de Matematica si Informatica');
EXECUTE proiect_daria.Ex7('Examen Scris', 7);
BEGIN
    DBMS_OUTPUT.PUT_LINE('Rezultat Ex8: ' || proiect_daria.Ex8(111));
END;
```

Script Output x Query Result x

Task completed in 0.075 seconds

```
-----
Cursul Chimie organica
-----

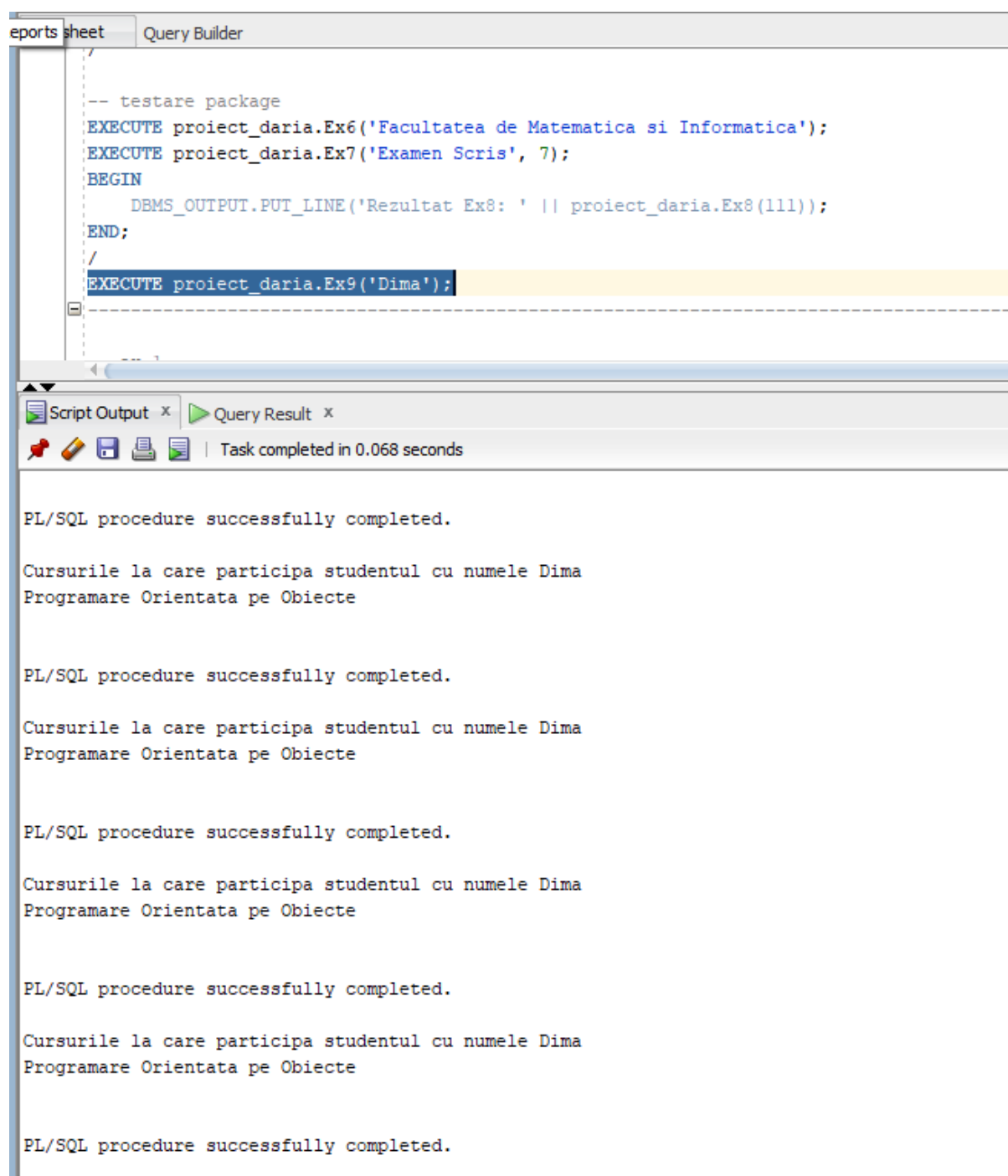
Cursul Arhitectura sistemelor de calcul
-----

Cursul Gandire Critica si Etica Academica
-----
Nimara Dan a obtinut nota 10

Cursul Limba si Literatura Engleza
-----
Fritz Raluca a obtinut nota 10
Vultur Sofia a obtinut nota 8.7

Cursul Chimie organica
-----
Dima Oana a obtinut nota 10
Baciu Daniel a obtinut nota 8
Fritz Raluca a obtinut nota 10
Guleama Dan a obtinut nota 7.7

PL/SQL procedure successfully completed.
```



The screenshot displays the Oracle SQL Developer environment. The top pane, titled 'Query Builder', contains a PL/SQL script. The script starts with a comment '-- testare package', followed by two EXECUTE statements: 'EXECUTE proiect_daria.Ex6('Facultatea de Matematica si Informatica');' and 'EXECUTE proiect_daria.Ex7('Examen Scris', 7);'. These are followed by a BEGIN block containing a DBMS_OUTPUT.PUT_LINE statement: 'DBMS_OUTPUT.PUT_LINE('Rezultat Ex8: ' || proiect_daria.Ex8(111));'. The block ends with END; and a forward slash /. The final line of the script is 'EXECUTE proiect_daria.Ex9('Dima');', which is highlighted in yellow. The bottom pane, titled 'Script Output', shows the results of the script execution. It includes a status bar indicating 'Task completed in 0.068 seconds'. The output consists of five identical blocks, each starting with 'PL/SQL procedure successfully completed.' followed by the text 'Cursurile la care participa studentul cu numele Dima' and 'Programare Orientata pe Obiecte' on separate lines.

```
-- testare package
EXECUTE proiect_daria.Ex6('Facultatea de Matematica si Informatica');
EXECUTE proiect_daria.Ex7('Examen Scris', 7);
BEGIN
    DBMS_OUTPUT.PUT_LINE('Rezultat Ex8: ' || proiect_daria.Ex8(111));
END;
/
EXECUTE proiect_daria.Ex9('Dima');
```

Script Output x Query Result x

Task completed in 0.068 seconds

PL/SQL procedure successfully completed.

Cursurile la care participa studentul cu numele Dima
Programare Orientata pe Obiecte

PL/SQL procedure successfully completed.

Cursurile la care participa studentul cu numele Dima
Programare Orientata pe Obiecte

PL/SQL procedure successfully completed.

Cursurile la care participa studentul cu numele Dima
Programare Orientata pe Obiecte

PL/SQL procedure successfully completed.

Cursurile la care participa studentul cu numele Dima
Programare Orientata pe Obiecte

PL/SQL procedure successfully completed.

14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare unui flux de acțiuni integrate, specifice bazei de date definite (minim 2 tipuri de date, minim 2 funcții, minim 2 proceduri).

Mi-am facut un pachet care afiseaza pentru rumaoarele date de input:

- nume si prenume student
- nume si prenume profesor
- denumire curs

=> Salile in care acestia desfasoara cursuri conform orarului.

Mi-am facut 4 functii pentru optinerea unei chei primare pentru campuri din input, dupa care am utilizat 2 proceduri - una pentru afisare si alte pentru colectarea datelor in tabelul imbricat, respectiv in tabelul indexat.

```
CREATE OR REPLACE PACKAGE Ex14 AS
    TYPE tip_tabel_orar IS TABLE OF orar%rowtype INDEX BY PLS_INTEGER;
    tabel_orar tip_tabel_orar;

    TYPE tip_tabel_sali IS TABLE OF sala.cod_sala%type;
    tabel_sali tip_tabel_sali := tip_tabel_sali();

    FUNCTION GetStudentId(num Student.nume%type, prenume Student.prenume%type)
        RETURN Student.cod_student%type;
    FUNCTION GetGrupa(cod Stud Student.cod_student%type)
        RETURN grupa.cod_grupa%type;
    FUNCTION GetProfesor(num Prof profesor.nume%type, prenume Prof profesor.prenume%type)
        RETURN profesor.cod_profesor%type;
    FUNCTION GetCurs(denumire Curs curs.denumire%type)
        RETURN curs.cod_curs%type;

    PROCEDURE GetOrar(
        nume_student IN Student.nume%type,
        prenume_student IN Student.prenume%type,
        nume_profesor IN profesor.nume%type,
        prenume_profesor IN profesor.prenume%type,
        denumire_curs IN curs.denumire%type);

    PROCEDURE afisSala(cod IN sala.cod_sala%type);

END Ex14;
/

CREATE OR REPLACE PACKAGE BODY Ex14 AS
    FUNCTION GetStudentId(num Student.nume%type, prenume Student.prenume%type)
```

```
        RETURN student.cod_student%type AS
        v_student_id student.cod_student%type;
        nr NUMBER(6) := 0;
    BEGIN
        DBMS_OUTPUT.PUT_LINE('Caut studentul ' || prenume_student || ' ' ||
nume_student);
        SELECT COUNT(cod_student)
        INTO nr
        FROM student s
        WHERE UPPER(s.nume) = UPPER(nume_student) AND UPPER(s.prenume) =
UPPER(prenume_student);
        --DBMS_OUTPUT.PUT_LINE(nr);
        IF nr > 1 THEN
            DBMS_OUTPUT.PUT_LINE('Exista mai multi studenti');
            RETURN 0;
        ELSIF nr = 0 THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista studenti');
            RETURN 0;
        ELSE
            SELECT cod_student
            INTO v_student_id
            FROM student s
            WHERE UPPER(s.nume) = UPPER(nume_student) AND UPPER(s.prenume) =
UPPER(prenume_student);

            RETURN v_student_id;
        END IF;
    END GetStudentId;

FUNCTION GetGrupa(cod_stud student.cod_student%type)
RETURN grupa.cod_grupa%type AS
v_grupa_id grupa.cod_grupa%type;
nr NUMBER(6);
BEGIN
    DBMS_OUTPUT.PUT_LINE('Caut grupa studentului cu codul ' || cod_stud);
    SELECT COUNT(cod_grupa)
    INTO nr
    FROM inscriere s
    WHERE s.cod_student = cod_stud;

    IF nr > 1 THEN
        DBMS_OUTPUT.PUT_LINE('Exista mai multe grupe');
        RETURN 0;
    ELSIF nr = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Nu exista grupe');
        RETURN 0;
    ELSE
        SELECT cod_grupa
        INTO v_grupa_id
        FROM inscriere s
        WHERE s.cod_student = cod_stud;

        RETURN v_grupa_id;
    END IF;
END GetGrupa;
```

```
FUNCTION GetProfesor(ume_prof profesor.ume%type, prenume_prof
profesor.prenume%type)
RETURN profesor.cod_profesor%type AS
v_profesor_id grupa.cod_grupa%type;
nr NUMBER(6);
BEGIN
DBMS_OUTPUT.PUT_LINE('Caut profesorul ' || ume_prof || ' ' ||
prenume_prof);
SELECT COUNT(cod_profesor)
INTO nr
FROM profesor p
WHERE p.ume = ume_prof AND p.prenume = prenume_prof;

IF nr > 1 THEN
DBMS_OUTPUT.PUT_LINE('Exista mai multi profesori');
RETURN 0;
ELSIF nr = 0 THEN
DBMS_OUTPUT.PUT_LINE('Nu exista profesori');
RETURN 0;
ELSE
SELECT cod_profesor
INTO v_profesor_id
FROM profesor p
WHERE p.ume = ume_prof AND p.prenume = prenume_prof;

RETURN v_profesor_id;
END IF;
END GetProfesor;

FUNCTION GetCurs(denumire_curs curs.denumire%type)
RETURN curs.cod_curs%type AS
v_curs_id curs.cod_curs%type;
nr NUMBER(6);
BEGIN
DBMS_OUTPUT.PUT_LINE('Caut cursul ' || denumire_curs);
SELECT COUNT(cod_curs)
INTO nr
FROM curs c
WHERE c.denumire = denumire_curs;

IF nr > 1 THEN
DBMS_OUTPUT.PUT_LINE('Exista mai multe cursuri');
RETURN 0;
ELSIF nr = 0 THEN
DBMS_OUTPUT.PUT_LINE('Nu exista cursuri');
RETURN 0;
ELSE
SELECT cod_curs
INTO v_curs_id
FROM curs c
WHERE c.denumire = denumire_curs;

RETURN v_curs_id;
END IF;
```

```
END GetCurs;

PROCEDURE afisSala(cod IN sala.cod_sala%type) AS
    den sala.denumire%type;
    nr NUMBER(6);
    BEGIN
        SELECT count(*)
        INTO nr
        FROM sala
        WHERE cod_sala = cod;
        IF nr = 0 THEN
            DBMS_OUTPUT.PUT_LINE('Nu exista');
        ELSE
            SELECT denumire
            INTO den
            FROM sala
            WHERE cod_sala = cod;

            DBMS_OUTPUT.PUT_LINE(den);
        END IF;
    END afisSala;

PROCEDURE GetOrar(
    nume_student IN student.nume%type,
    prenume_student IN student.prenume%type,
    nume_profesor IN profesor.nume%type,
    prenume_profesor IN profesor.prenume%type,
    denumire_curs IN curs.denumire%type) AS
    cod_stud student.cod_student%type;
    cod_prof profesor.cod_profesor%type;
    cod_c curs.cod_curs%type;
    cod_gr grupa.cod_grupa%type;
    cod_sala sala.cod_sala%type;
    v_orar orar%rowtype;
    BEGIN
        cod_stud := GetStudentId(nume_student, prenume_student);
        IF cod_stud != 0 THEN
            cod_gr := GetGrupa(cod_stud);
            cod_prof := GetProfesor(nume_profesor, prenume_profesor);
            cod_c := GetCurs(denumire_curs);
            IF cod_stud != 0 AND cod_prof != 0 AND cod_c != 0 THEN
                --DBMS_OUTPUT.PUT_LINE('Am ajuns aici');
                DBMS_OUTPUT.PUT_LINE(cod_prof || ' ' || cod_gr || ' ' || cod_c);
                SELECT *
                BULK COLLECT INTO tabel_orar
                FROM orar o
                WHERE o.cod_profesor = cod_prof AND cod_gr = o.cod_grupa AND
cod_c = o.cod_curs;
                --DBMS_OUTPUT.PUT_LINE(tabel_orar.count);
                IF tabel_orar.count > 0 THEN
                    FOR i in tabel_orar.FIRST..tabel_orar.LAST LOOP
                        cod := tabel_orar(i).cod_sala;
```

```
        tabel_sali.extend;
        tabel_sali(i) := cod;
        --DBMS_OUTPUT.PUT_LINE(cod);
    END LOOP;
    IF tabel_sali.count > 0 THEN
        FOR i in tabel_sali.FIRST..tabel_sali.LAST LOOP
            afisSala(tabel_sali(i));
        END LOOP;
    END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Nu se poate');
    END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Nu exista');
    END IF;
    ELSE
        DBMS_OUTPUT.PUT_LINE('Student incorect');
    END IF;
END GetOrar;

END Ex14;
/
SELECT *
from orar;
select *
from curs
where cod_curs = 303;
select *
from inscriere
where cod_grupa = 41;
select *
from student
where cod_student = 52;
select *
from profesor
where cod_profesor = 102;

EXECUTE Ex14.GetOrar('Dima', 'Oana', 'Avram', 'Bianca', 'Programare Orientata pe
Obiecte');
BEGIN
    DBMS_OUTPUT.PUT_LINE(Ex14.GetStudentId('Dima', 'Oana'));
END;
/
```

```
Caut studentul Oana Dima
Caut grupa studentului cu codul 52
Caut profesorul Avram Bianca
Caut cursul Programare Orientata pe Obiecte
102 41 303
Amfiteatrul Haret
```