Proiect SGBD

- Atelier Arhitectura -

CUPRINS

1. Prezentarea pe scurt a bazei de date (utilitatea ei)

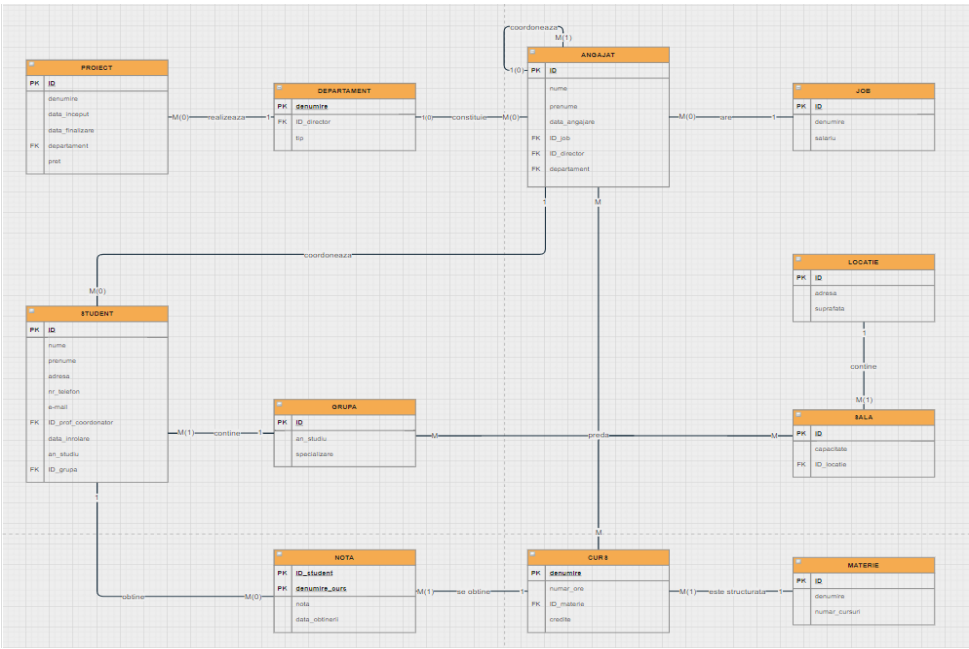
In prezentarea aceasta va voi introduce in cadrul **"*Atelierului de Arhitectura*"**, unde putem gasi specialisti in domeniul Arhitecturii, al Design-ului Interior si al Urbanismului, angajati in diverse Proiecte inovative, impreuna cu Profesorii care se ocupa de formarea viitorilor Arhitecti, prin predarea secretelor meseriei si prin indrumarea acestora catre succes in aceasta cariera esentiala in societate, bazata pe creativitate, stapanirea conceptelor teoretice si multa practica.

Scopul gestiunii acestei bazei de date este de a facilita operarea cu informatii si date legate de toate aspectele ce implica un Atelier de Arhitectura: realizarea proiectelor de catre angajati, activitatile de predare ale profesorilor si cele de invatare ale studentilor etc.

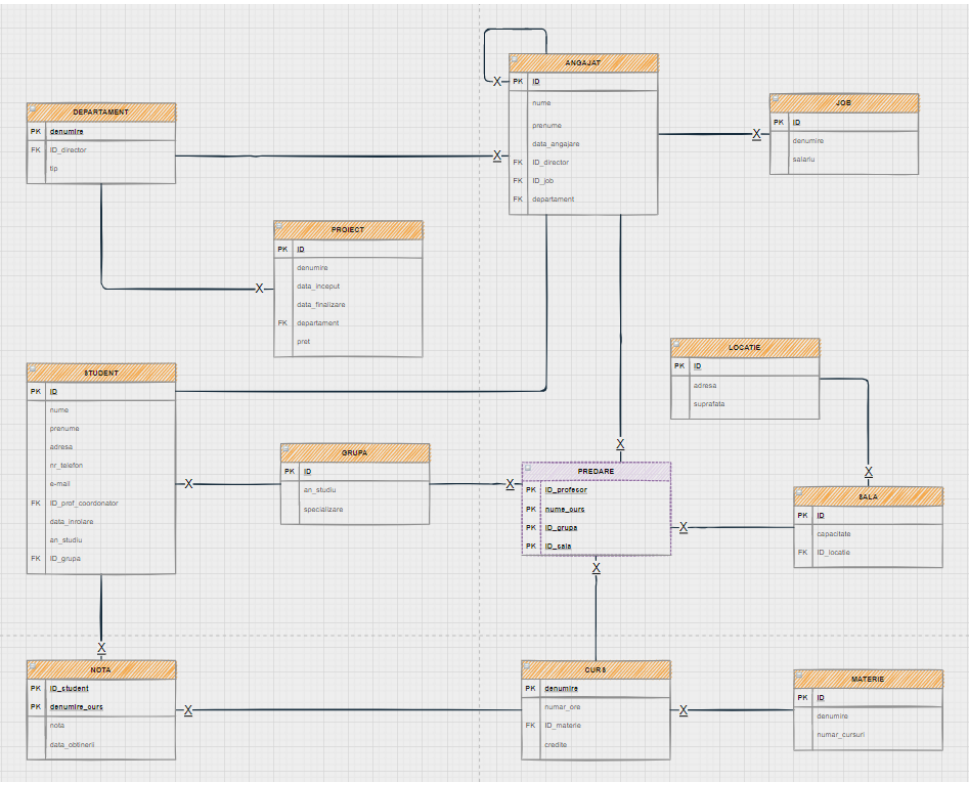
Prezentarea, pe scurt, a entitatilor si constrangerilor ce definesc baza de date:

- 12 tabele, dintre care 2 asociative (NOTA si PREDARE)
- fiecare angajat are un manager (sau null, in cazul in care acesta este managerul atelierului)
- fiecare angajat face parte dintr-un singur departament
- fiecare angajat primeste salariul in functie de job-ul in care activeaza
- un proiect poate fi realizat de un singur departament

2. Diagrama entitate-relatie (ERD) – in FN3



3. Diagrama conceptuala



4. Implementarea diagramei conceptuale in Oracle

1) JOB

Worksheet | Query Builder

```
-- crearea si popularea tabelei JOB
create table JOB(
ID number(4), primary key(ID),
denumire VARCHAR2(15),
salariu NUMBER(6));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into JOB values(IDs.nextval, 'Proiectant', 5000);
insert into JOB values(IDs.nextval, 'Profesor', 4100);
insert into JOB values(IDs.nextval, 'Manager', 7300);
insert into JOB values(IDs.nextval, 'Asistent', 2200);
insert into JOB values(IDs.nextval, 'Furnizor', 3500);

drop sequence IDs;
```

Script Output | Task completed in 0.026 seconds

```
Table JOB created.

Sequence IDS created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

2) *DEPARTAMENT*

Worksheet | Query Builder

```
-- crearea si popularea tabelei DEPARTAMENT
create table DEPARTAMENT(
denumire varchar2(15), primary key(denumire),
ID_director number(4),
tip varchar2(10));

insert into DEPARTAMENT values('Confectii', 5, 'Productie');
insert into DEPARTAMENT values('Machete', 4, 'Productie');
insert into DEPARTAMENT values('Planse', 7, 'Cercetare');
insert into DEPARTAMENT values('Software', 1, 'Cercetare');
insert into DEPARTAMENT values('Asezari', 2, 'Cercetare');
```

Script Output

Task completed in 0.027 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

Sequence IDS dropped.

Table DEPARTAMENT created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

3) PROIECT

Worksheet | Query Builder

```
-- crearea si popularea tabelei PROIECT
create table PROIECT(
ID number(6), primary key(ID),
denumire varchar2(30),
data_inceput date,
data_final date,
departament varchar2(15),
pret float(2),
foreign key(departament) references DEPARTAMENT(denumire));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 999999
nocycle;

insert into PROIECT values(IDs.nextval, 'Restaurare Castel', '10-03-2019', '18-03-2019', 'Planse', 450.00);
insert into PROIECT values(IDs.nextval, 'Macheta Residence', '21-05-2021', null, 'Machete', 1800.00);
insert into PROIECT values(IDs.nextval, 'Colaj Ateneul Roman', '07-07-2020', '02-03-2021', 'Asezari', 300.00);
insert into PROIECT values(IDs.nextval, 'Casa lui Mos-Martin', '01-04-2021', null, 'Machete', 2500.00);
insert into PROIECT values(IDs.nextval, 'Eficientizare AutoCad', '13-04-2021', '14-04-2021', 'Software', 135.00);

drop sequence IDs;
```

Script Output | Task completed in 0.047 seconds

```
1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Sequence IDS dropped.
```

4) ANGAJAT

Worksheet | Query Builder

```
-- crearea si popularea tabelei ANGAJAT
create table ANGAJAT(
ID number(4), primary key(ID),
nume varchar2(15),
prenume varchar2(15),
data_angajare date,
ID_director number(4),
departament varchar2(15),
ID_job number(4),
foreign key(ID_director) references ANGAJAT(ID),
foreign key(departament) references DEPARTAMENT(denumire),
foreign key(ID_job) references JOB(ID));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into ANGAJAT values(IDs.nextval, 'Neagu', 'Alexandra-Ioana', '22-12-2015', null, 'Asezari', 3);
insert into ANGAJAT values(IDs.nextval, 'Anton', 'Mihai-Cosmin', '17-11-2014', 1, 'Software', 1);
insert into ANGAJAT values(IDs.nextval, 'Dima', 'Carol-Valentin', '13-01-2016', 1, null, 2);
```

Script Output | Task completed in 0.044 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Sequence IDS dropped.
```

5) GRUPA

Worksheet | Query Builder

```
-- crearea si popularea tabelei GRUPA
create table GRUPA(
ID number(4), primary key(ID),
an_studiu number(1),
specializare varchar2(15));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into GRUPA values(IDs.nextval, 3, 'Urbanism');
insert into GRUPA values(IDs.nextval, 1, 'Arhitectura');
insert into GRUPA values(IDs.nextval, 2, 'Arhitectura');
insert into GRUPA values(IDs.nextval, 2, 'Interior');
insert into GRUPA values(IDs.nextval, 1, 'Urbanism');
insert into GRUPA values(IDs.nextval, 3, 'Arhitectura');

drop sequence IDs;
```

Script Output | Task completed in 0.06 seconds

```
Table GRUPA created.


Sequence IDS created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

6) STUDENT

Worksheet | Query Builder

```
-- crearea si popularea tabelei STUDENT
create table STUDENT(
ID number(6), primary key(ID),
nume varchar2(15),
prenume varchar2(15),
nr_telefon varchar2(10),
email varchar2(30),
ID_grupa number(4),
ID_prof_coordonator number(4),
data_inrolare date,
an_studiu number(1),
foreign key(ID_grupa) references GRUPA(ID),
foreign key(ID_prof_coordonator) references ANGAJAT(ID));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 999999
nocycle;

insert into STUDENT values(IDs.nextval, 'Giurgiuleanu', 'Matei', '0727835824', 'gmatei@yahoo.ro', 5, 3, '10-04-2019', 1);
insert into STUDENT values(IDs.nextval, 'Sotau', 'Marin', '0256207670', 'smarin@yahoo.ro', 1, 3, '27-04-2019', 3);
insert into STUDENT values(IDs.nextval, 'Robert', 'Ioan', '0235423932', 'rioan@yahoo.ro', 3, 6, '31-07-2020', 2);
insert into STUDENT values(IDs.nextval, 'Barbu', 'Matei', '0727333342', 'bmatei@yahoo.ro', 2, 3, '06-12-2020', 1);
```

Script Output | Task completed in 0.055 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

Sequence IDS dropped.
```

7) MATERIE

Worksheet | Query Builder

```
-- crearea si popularea tabelei MATERIE
create table MATERIE(
ID number(4), primary key(ID),
denumire varchar2(30),
numar_cursuri number(2));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into MATERIE values(IDs.nextval, 'Studiul Formei', 8);
insert into MATERIE values(IDs.nextval, 'Istoria Asezarilor in Europa', 8);
insert into MATERIE values(IDs.nextval, 'Geometrie Descriptiva', 16);
insert into MATERIE values(IDs.nextval, 'Introducere in Arh. Cont.', 12);
insert into MATERIE values(IDs.nextval, 'Perspectiva', 16);
insert into MATERIE values(IDs.nextval, 'Limba Engleza', 4);

drop sequence IDs;
```

Script Output | Task completed in 0.052 seconds

```
Table MATERIE created.


Sequence IDS created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

8) CURS

Worksheet | Query Builder

```
-- crearea si popularea tabelei CURS
create table CURS(
denumire varchar2(30), primary key(denumire),
numar_ore number(1),
ID_materie number(4),
credite number(1),
foreign key(ID_materie) references MATERIE(ID));

insert into CURS values('Cunoasterea Spatiului', 2, 1, 2);
insert into CURS values('Diagrame', 2, 1, 3);
insert into CURS values('Eseu Grafic', 4, 1, 2);
insert into CURS values('Substractie si Aditie', 2, 1, 3);
insert into CURS values('Elemente Teoretice', 1, 2, 2);
insert into CURS values('Orasul Antic', 1, 2, 2);
insert into CURS values('Orasul Medieval', 1, 2, 2);
insert into CURS values('Orasul Industrial', 1, 2, 2);
insert into CURS values('Umbre 2D', 2, 3, 4);
insert into CURS values('Axonometrie', 4, 3, 6);
insert into CURS values('Poliedre', 2, 3, 4);
insert into CURS values('Scari', 6, 3, 5);
insert into CURS values('Vernacular', 2, 4, 3);
```

Script Output | Task completed in 0.077 seconds

```
Table CURS created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

9) LOCATIE

Worksheet | Query Builder

```
-- crearea si popularea tabelei LOCATIE
create table LOCATIE(
ID number(2), primary key(ID),
adresa varchar2(60),
suprafata number(6));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 99
nocycle;

insert into LOCATIE values(IDs.nextval, 'str. Belindo, nr. 6', 12500);
insert into LOCATIE values(IDs.nextval, 'str. Aurie, nr. 7A', 1500);
insert into LOCATIE values(IDs.nextval, 'str. Teilor, nr. 4', 20300);
insert into LOCATIE values(IDs.nextval, 'str. Suspendata, nr. 2B', 780);
insert into LOCATIE values(IDs.nextval, 'str. Nikolas Vinz, nr. 11', 321400);

drop sequence IDs;
```

Script Output | Task completed in 0.044 seconds

```
Table LOCATIE created.


Sequence IDS created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


Sequence IDS dropped.
```

10) SALA

Worksheet | Query Builder

```
-- crearea si popularea tabelei SALA
create table SALA(
ID number(4), primary key(ID),
capacitate number(2),
ID_locatie number(2),
foreign key(ID_locatie) references LOCATIE(ID));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into SALA values(IDs.nextval, 18, 1);
insert into SALA values(IDs.nextval, 10, 2);
insert into SALA values(IDs.nextval, 28, 4);
insert into SALA values(IDs.nextval, 70, 3);
insert into SALA values(IDs.nextval, 20, 5);
insert into SALA values(IDs.nextval, 80, 1);
insert into SALA values(IDs.nextval, 60, 4);
```

Script Output | Task completed in 0.065 seconds

```
Table SALA created.


Sequence IDS created.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.


1 row inserted.
```

11) NOTA

Worksheet | Query Builder

```
-- crearea si popularea tabelei NOTA
create table NOTA(
ID_student number(6),
denumire_curs varchar2(30),
primary key(ID_student, denumire_curs),
nota float(8),
data_obtinerii date,
foreign key(ID_student) references STUDENT(ID),
foreign key(denumire_curs) references CURS(denumire));

insert into NOTA values(1, 'Scari', 7.50, '12-06-2020');
insert into NOTA values(1, 'Axonometrie', 8.50, '13-07-2020');
insert into NOTA values(2, 'Poliedre', 9.50, '11-05-2019');
insert into NOTA values(2, 'Axonometrie', 9, '13-07-2019');
insert into NOTA values(3, 'Poliedre', 7, '13-07-2020');
insert into NOTA values(4, 'Poliedre', 6, '12-06-2020');
insert into NOTA values(4, 'Umbre 2D', 8.50, '11-05-2019');
insert into NOTA values(4, 'Orasul Traditional', 4.30, '11-05-2019');
insert into NOTA values(4, 'Arhitectul', 8.10, '13-07-2020');
insert into NOTA values(5, 'Poliedre', 7.20, '11-05-2019');
insert into NOTA values(5, 'Umbre 2D', 8.90, '12-06-2020');
```

Script Output | Task completed in 0.059 seconds

```
Table NOTA created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

12) PREDARE

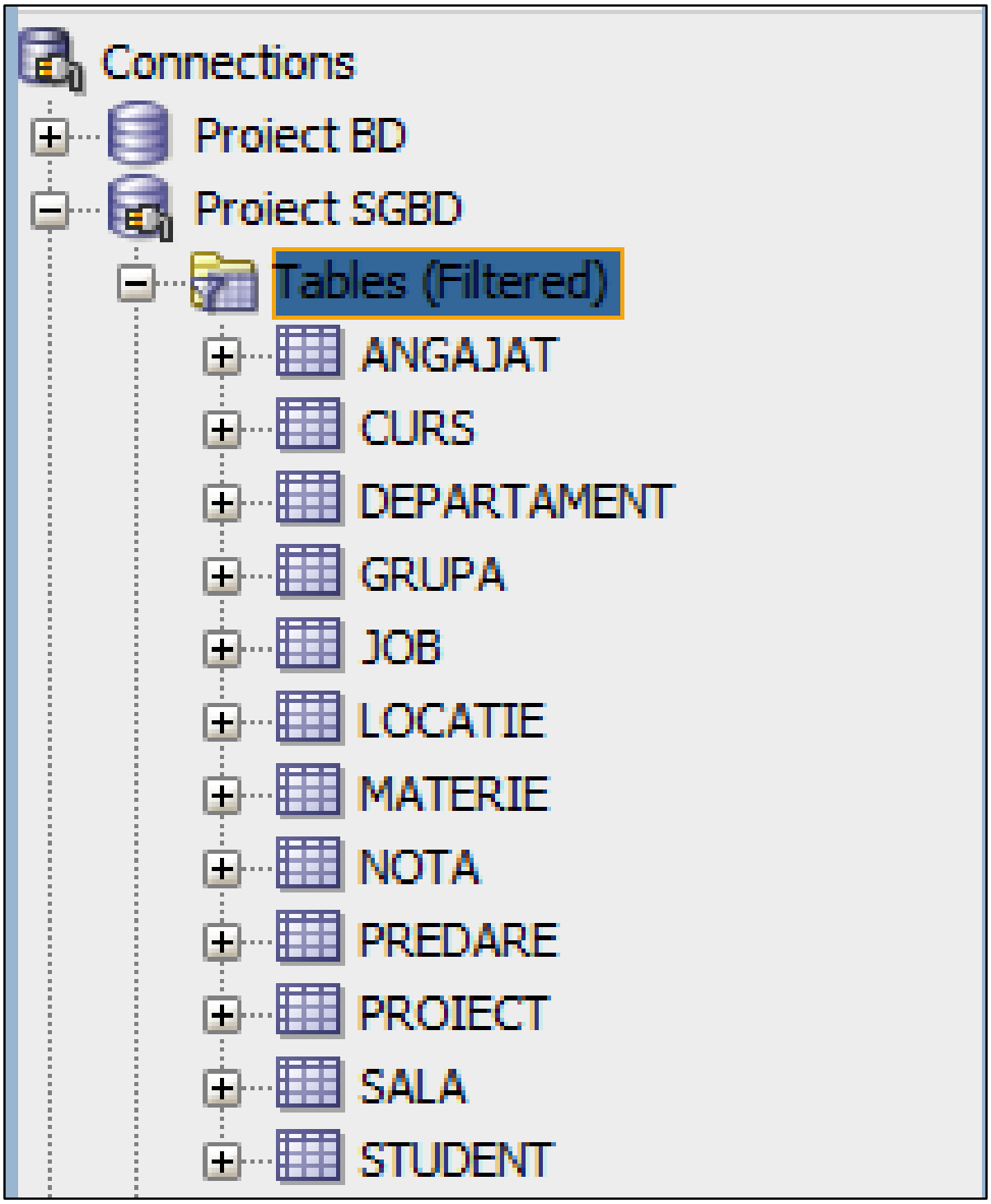
Worksheet | Query Builder

```
-- crearea si popularea tabelei PREDARE
create table PREDARE(
ID_profesor number(4),
nume_curs varchar2(30),
ID_grupa number(4),
ID_sala number(4),
primary key(ID_profesor, nume_curs, ID_grupa, ID_sala),
foreign key(ID_profesor) references ANGAJAT(ID),
foreign key(nume_curs) references CURS(denumire),
foreign key(ID_grupa) references GRUPA(ID),
foreign key(ID_sala) references SALA(ID));

insert into PREDARE values(3, 'Poliedre', 1, 1);
insert into PREDARE values(3, 'Axonometrie', 1, 1);
insert into PREDARE values(3, 'Poliedre', 2, 2);
insert into PREDARE values(3, 'Scari', 5, 3);
insert into PREDARE values(3, 'Umbre 2D', 2, 3);
insert into PREDARE values(3, 'Poliedre', 3, 4);
insert into PREDARE values(3, 'Axonometrie', 5, 5);
insert into PREDARE values(6, 'Arhitectul', 4, 1);
insert into PREDARE values(6, 'Arhitectul', 6, 6);
```

Script Output | Task completed in 0.056 seconds

```
Table PREDARE created.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

13) VEDERE de ANSAMBLU



5. Popularea tabelelor cu inregistrari in Oracle:

```
1. drop table predare;
2. drop table nota;
3. drop table sala;
4. drop table locatie;
5. drop table curs;
6. drop table materie;
7. drop table student;
8. drop table grupa;
9. drop table angajat;
10. drop table proiect;
11. drop table departament;
12. drop table job;
13.
14. ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MM-YYYY';
15.
16.
17. -- crearea si popularea tabelei JOB
18. create table JOB(
19. ID number(4), primary key(ID),
20. denumire varchar2(15),
21. salariu number(6));
22.
23. create sequence IDs
24. start with 1
25. increment by 1
26. minvalue 0
27. maxvalue 9999
28. nocycle;
29.
30. insert into JOB values(IDs.nextval, 'Proiectant', 5000);
31. insert into JOB values(IDs.nextval, 'Profesor', 4100);
32. insert into JOB values(IDs.nextval, 'Manager', 7300);
33. insert into JOB values(IDs.nextval, 'Asistent', 2200);
34. insert into JOB values(IDs.nextval, 'Furnizor', 3500);
35.
36. drop sequence IDs;
37. commit;
38.
39.
40. -- crearea si popularea tabelei DEPARTAMENT
41. create table DEPARTAMENT(
42. denumire varchar2(15), primary key(denumire),
43. ID_director number(4),
44. tip varchar2(10));
45.
46. insert into DEPARTAMENT values('Confectii', 5, 'Productie');
47. insert into DEPARTAMENT values('Machete', 4, 'Productie');
48. insert into DEPARTAMENT values('Planse', 7, 'Cercetare');
49. insert into DEPARTAMENT values('Software', 1, 'Cercetare');
50. insert into DEPARTAMENT values('Asezari', 2, 'Cercetare');
51.
52. -- crearea si popularea tabelei PROIECT
53. create table PROIECT(
54. ID number(6), primary key(ID),
```

```
denumire varchar2(30),
data_inceput date,
data_final date,
departament varchar2(15),
pret float(2),
foreign key(departament) references DEPARTAMENT(denumire));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 999999
nocycle;

insert into PROIECT values(IDs.nextval, 'Restaurare Castel', '10-03-2019', '18-03-2019', 'Planse', 450.00);
insert into PROIECT values(IDs.nextval, 'Macheta Residence', '21-05-2021', null, 'Machete', 1800.00);
insert into PROIECT values(IDs.nextval, 'Colaj Ateneul Roman', '07-07-2020', '02-03-2021', 'Asezari', 300.00);
insert into PROIECT values(IDs.nextval, 'Casa lui Mos-Martin', '01-04-2021', null, 'Machete', 2500.00);
insert into PROIECT values(IDs.nextval, 'Eficientizare AutoCad', '13-04-2021', '14-04-2021', 'Software', 135.00);

drop sequence IDs;
commit;


-- crearea si popularea tabelei ANGAJAT
create table ANGAJAT(
ID number(4), primary key(ID),
nume varchar2(15),
prenume varchar2(15),
data_angajare date,
ID_director number(4),
departament varchar2(15),
ID_job number(4),
foreign key(ID_director) references ANGAJAT(ID),
foreign key(departament) references DEPARTAMENT(denumire),
foreign key(ID_job) references JOB(ID));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into ANGAJAT values(IDs.nextval, 'Neagu', 'Alexandra-Ioana', '22-12-2015', null, 'Asezari', 3);
insert into ANGAJAT values(IDs.nextval, 'Anton', 'Mihai-Cosmin', '17-11-2014', 1, 'Software', 1);
insert into ANGAJAT values(IDs.nextval, 'Dima', 'Carol-Valentin', '13-01-2016', 1, null, 2);
insert into ANGAJAT values(IDs.nextval, 'Florea', 'Irina', '26-12-2019', 1, 'Confectii', 4);
insert into ANGAJAT values(IDs.nextval, 'Benescu', 'Ioan', '02-02-2020', 1, 'Planse', 4);
```

```
insert into ANGAJAT values(IDs.nextval, 'Gruia', 'Gabriel', '27-06-2019', 1, null, 2);
insert into ANGAJAT values(IDs.nextval, 'Dragulescu', 'Raluca', '02-08-2016', 4, 'Machete', 5);

drop sequence IDs;
commit;


-- crearea si popularea tabelei GRUPA
create table GRUPA(
ID number(4), primary key(ID),
an_studiu number(1),
specializare varchar2(15));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into GRUPA values(IDs.nextval, 3, 'Urbanism');
insert into GRUPA values(IDs.nextval, 1, 'Arhitectura');
insert into GRUPA values(IDs.nextval, 2, 'Arhitectura');
insert into GRUPA values(IDs.nextval, 2, 'Interior');
insert into GRUPA values(IDs.nextval, 1, 'Urbanism');
insert into GRUPA values(IDs.nextval, 3, 'Arhitectura');
drop sequence IDs;
commit;

-- crearea si popularea tabelei STUDENT
create table STUDENT(
ID number(6), primary key(ID),
nume varchar2(15),
prenume varchar2(15),
nr_telefon varchar2(10),
email varchar2(30),
ID_grupa number(4),
ID_prof_coordonator number(4),
data_inrolare date,
an_studiu number(1),
foreign key(ID_grupa) references GRUPA(ID),
foreign key(ID_prof_coordonator) references ANGAJAT(ID));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 999999
nocycle;

insert into STUDENT values(IDs.nextval, 'Giurgiuleanu', 'Matei', '0727835824', 'gmatei@yahoo.ro', 5, 3, '10-04-2019', 1);
insert into STUDENT values(IDs.nextval, 'Sotau', 'Marin', '0256207670', 'smarin@yahoo.ro', 1, 3, '27-04-2019', 3);
insert into STUDENT values(IDs.nextval, 'Robert', 'Ioan', '0235423932', 'rioan@yahoo.ro', 3, 6, '31-07-2020', 2);
```

```
insert into STUDENT values(IDs.nextval, 'Barbu', 'Matei', '0727333342', 'bmatei@yahoo.ro', 2, 3, '06-12-2020', 1);
insert into STUDENT values(IDs.nextval, 'Stefanescu', 'Bob', '0269559228', 'sbob@yahoo.ro', 2, 6, '28-09-2021', 1);
insert into STUDENT values(IDs.nextval, 'Paun', 'Cristinel', '0212520355', 'pcristinel@yahoo.ro', 6, 6, '06-11-2021', 3);
insert into STUDENT values(IDs.nextval, 'Bitulescu', 'Carol', '0214082820', 'bcarol@yahoo.ro', 1, 3, '20-08-2021', 3);
insert into STUDENT values(IDs.nextval, 'Popos', 'Florentin', '0740142399', 'pflorentin@yahoo.ro', 4, 3, '13-02-2021', 1);
insert into STUDENT values(IDs.nextval, 'Dobre', 'Marcel', '0721328241', 'dmarcel@yahoo.ro', 4, 3, '24-03-2020', 2);
insert into STUDENT values(IDs.nextval, 'Rosevilici', 'Teodor', '0744555788', 'rteodor@yahoo.ro', 2, 6, '19-07-2019', 1);

drop sequence IDs;
commit;


-- crearea si popularea tabelei MATERIE
create table MATERIE(
ID number(4), primary key(ID),
denumire varchar2(30),
numar_cursuri number(2));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into MATERIE values(IDs.nextval, 'Studiul Formei', 8);
insert into MATERIE values(IDs.nextval, 'Istoria Asezarilor in Europa', 8);
insert into MATERIE values(IDs.nextval, 'Geometrie Descriptiva', 16);
insert into MATERIE values(IDs.nextval, 'Introducere in Arh. Cont.', 12);
insert into MATERIE values(IDs.nextval, 'Perspectiva', 16);
insert into MATERIE values(IDs.nextval, 'Limba Engleza', 4);

drop sequence IDs;
commit;


-- crearea si popularea tabelei CURS
create table CURS(
denumire varchar2(30), primary key(denumire),
numar_ore number(1),
ID_materie number(4),
credite number(1),
foreign key(ID_materie) references MATERIE(ID));

insert into CURS values('Cunoasterea Spatiului', 2, 1, 2);
insert into CURS values('Diagrame', 2, 1, 3);
insert into CURS values('Eseu Grafic', 4, 1, 2);
insert into CURS values('Substractie si Aditie', 2, 1, 3);
insert into CURS values('Elemente Teoretice', 1, 2, 2);
insert into CURS values('Orasul Antic', 1, 2, 2);
insert into CURS values('Orasul Medieval', 1, 2, 2);
insert into CURS values('Orasul Industrial', 1, 2, 2);
```

```
insert into CURS values('Umbre 2D', 2, 3, 4);
insert into CURS values('Axonometrie', 4, 3, 6);
insert into CURS values('Poliedre', 2, 3, 4);
insert into CURS values('Scari', 6, 3, 5);
insert into CURS values('Vernacular', 2, 4, 3);
insert into CURS values('Recuperarea Orasului', 1, 4, 2);
insert into CURS values('Arhitectul', 1, 4, 2);
insert into CURS values('Orasul Traditional', 1, 4, 2);
insert into CURS values('Sisteme de Proiectie', 4, 5, 5);
insert into CURS values('Perspectiva la Fuga', 2, 5, 4);
insert into CURS values('Trasarea Umbrelor', 4, 5, 6);
insert into CURS values('Perspectiva la Calculator', 2, 5, 4);
insert into CURS values('Past Tenses', 1, 6, 2);
insert into CURS values('Architectural Vocabulary', 2, 6, 3);
insert into CURS values('Useful Phrases', 2, 6, 2);

commit;


-- crearea si popularea tabelei LOCATIE
create table LOCATIE(
ID number(2), primary key(ID),
adresa varchar2(60),
suprafata number(6));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 99
nocycle;

insert into LOCATIE values(IDs.nextval, 'str. Belindo, nr. 6', 12500);
insert into LOCATIE values(IDs.nextval, 'str. Aurie, nr. 7A', 1500);
insert into LOCATIE values(IDs.nextval, 'str. Teilor, nr. 4', 20300);
insert into LOCATIE values(IDs.nextval, 'str. Suspendata, nr. 2B', 780);
insert into LOCATIE values(IDs.nextval, 'str. Nikolas Vinz, nr. 11',
321400);

drop sequence IDs;
commit;


-- crearea si popularea tabelei SALA
create table SALA(
ID number(4), primary key(ID),
capacitate number(2),
ID_locatie number(2),
foreign key(ID_locatie) references LOCATIE(ID));

create sequence IDs
start with 1
increment by 1
minvalue 0
maxvalue 9999
nocycle;

insert into SALA values(IDs.nextval, 18, 1);
insert into SALA values(IDs.nextval, 10, 2);
```

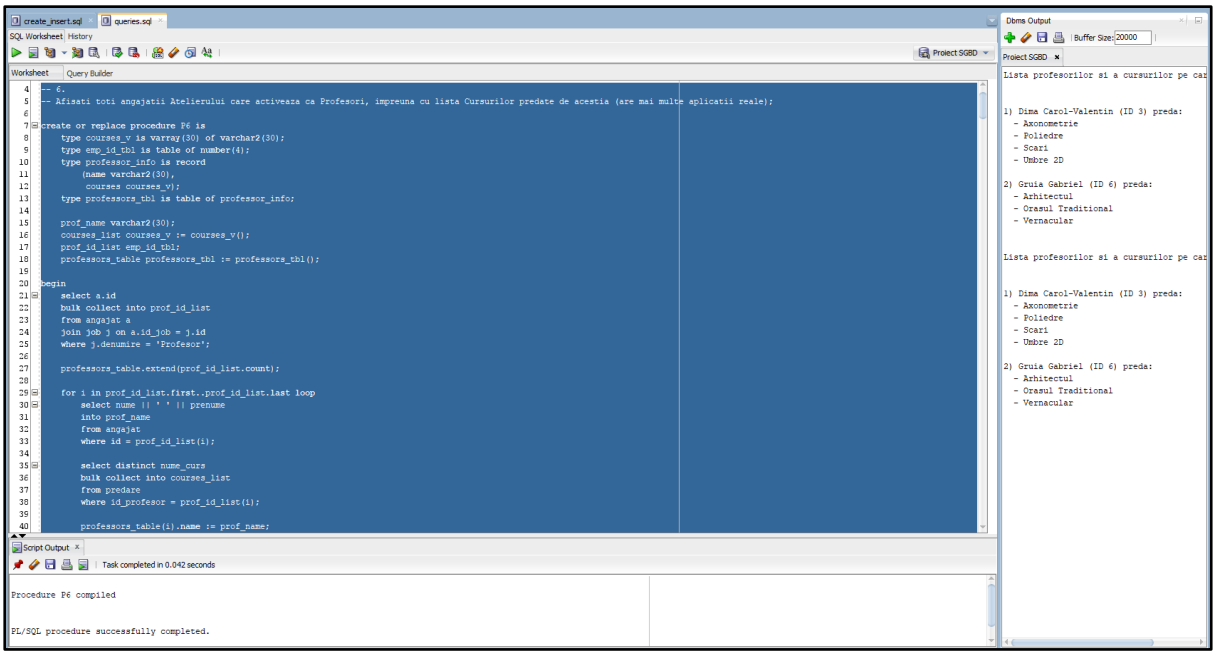
```
insert into SALA values(IDs.nextval, 28, 4);
insert into SALA values(IDs.nextval, 70, 3);
insert into SALA values(IDs.nextval, 20, 5);
insert into SALA values(IDs.nextval, 80, 1);
insert into SALA values(IDs.nextval, 60, 4);

drop sequence IDs;
commit;


-- crearea si popularea tabelei NOTA
create table NOTA(
ID_student number(6),
denumire_curs varchar2(30),
primary key(ID_student, denumire_curs),
nota float(8),
data_obtinerii date,
foreign key(ID_student) references STUDENT(ID),
foreign key(denumire_curs) references CURS(denumire));

insert into NOTA values(1, 'Scari', 7.50, '12-06-2020');
insert into NOTA values(1, 'Axonometrie', 8.50, '13-07-2020');
insert into NOTA values(2, 'Poliedre', 9.50, '11-05-2019');
insert into NOTA values(2, 'Axonometrie', 9, '13-07-2019');
insert into NOTA values(3, 'Poliedre', 7, '13-07-2020');
insert into NOTA values(4, 'Poliedre', 6, '12-06-2020');
insert into NOTA values(4, 'Umbre 2D', 8.50, '11-05-2019');
insert into NOTA values(4, 'Orasul Traditional', 4.30, '11-05-2019');
insert into NOTA values(4, 'Arhitectul', 8.10, '13-07-2020');
insert into NOTA values(5, 'Poliedre', 7.20, '11-05-2019');
insert into NOTA values(5, 'Umbre 2D', 8.90, '12-06-2020');
insert into NOTA values(5, 'Orasul Traditional', 8.50, '11-05-2019');
insert into NOTA values(5, 'Arhitectul', 10, '13-07-2020');
insert into NOTA values(6, 'Arhitectul', 9, '11-05-2019');
insert into NOTA values(6, 'Vernacular', 6.50, '13-07-2020');
insert into NOTA values(7, 'Poliedre', 8.50, '11-05-2019');
insert into NOTA values(7, 'Axonometrie', 7.10, '12-06-2020');
insert into NOTA values(8, 'Arhitectul', 3.20, '12-06-2020');
insert into NOTA values(9, 'Arhitectul', 9.10, '11-05-2019');
insert into NOTA values(10, 'Poliedre', 5, '13-07-2020');
insert into NOTA values(10, 'Umbre 2D', 5, '11-05-2019');
insert into NOTA values(10, 'Orasul Traditional', 4, '12-06-2020');
insert into NOTA values(10, 'Arhitectul', 6, '13-07-2020');

commit;


-- crearea si popularea tabelei PREDARE
create table PREDARE(
ID_profesor number(4),
nume_curs varchar2(30),
ID_grupa number(4),
ID_sala number(4),
primary key(ID_profesor, nume_curs, ID_grupa, ID_sala),
foreign key(ID_profesor) references ANGAJAT(ID),
foreign key(nume_curs) references CURS(denumire),
foreign key(ID_grupa) references GRUPA(ID),
foreign key(ID_sala) references SALA(ID));

```

```
insert into PREDARE values(3, 'Poliedre', 1, 1);
insert into PREDARE values(3, 'Axonometrie', 1, 1);
insert into PREDARE values(3, 'Poliedre', 2, 2);
insert into PREDARE values(3, 'Scari', 5, 3);
insert into PREDARE values(3, 'Umbre 2D', 2, 3);
insert into PREDARE values(3, 'Poliedre', 3, 4);
insert into PREDARE values(3, 'Axonometrie', 5, 5);
insert into PREDARE values(6, 'Arhitectul', 4, 1);
insert into PREDARE values(6, 'Arhitectul', 6, 6);
insert into PREDARE values(6, 'Vernacular', 6, 3);
insert into PREDARE values(6, 'Orasul Traditional', 2, 6);
insert into PREDARE values(6, 'Arhitectul', 2, 2);

commit;
```

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.

“Afisati toti angajatii Atelierului care activeaza ca Profesori, impreuna cu lista Cursurilor predate de acestia (are mai multe aplicatii reale)”.

Retinem intr-un nested table, ID-urile angajatilor care activaeza ca profesori in cadrul atelierului apoi, pentru fiecare dintre acestia, obtinem si afisam lista cursurilor (din tabela “PREDARE”) pe care acestia le-au sustinut.



```oracle_plsql
-- 6.
-- Afisati toti angajatii Atelierului care activeaza ca Profesori, impreuna cu lista Cursurilor predate de acestia (are mai multe aplicatii reale);

create or replace procedure P6 is
    type courses_v is varray(30) of varchar2(30);
    type emp_id_tbl is table of number(4);
    type professor_info is record
        (name varchar2(30),
         courses courses_v);
    type professors_tbl is table of professor_info;

    prof_name varchar2(30);
    courses_list courses_v := courses_v();
    prof_id_list emp_id_tbl;
    professors_table professors_tbl := professors_tbl();

begin
    select a.id
    bulk collect into prof_id_list
    from angajat a
    join job j on a.id_job = j.id
    where j.denumire = 'Profesor';

    professors_table.extend(prof_id_list.count);

    for i in prof_id_list.first..prof_id_list.last loop
        select nume || ' ' || prenume
        into prof_name
        from angajat
        where id = prof_id_list(i);

        select distinct nume_curs
        bulk collect into courses_list
        from predare
        where id_profesor = prof_id_list(i);

        professors_table(i).name := prof_name;
        professors_table(i).courses := courses_list;
    end loop;

    dbms_output.put_line('Lista profesorilor si a cursurilor pe care le sustin:');
    dbms_output.new_line;
    dbms_output.new_line;

    for i in professors_table.first..professors_table.last loop
        dbms_output.put_line(i || ') ' || professors_table(i).name || ' (ID ' || prof_id_list(i) || ') preda: ');
        for j in professors_table(i).courses.first..professors_table(i).courses.last loop
            dbms_output.put_line('  - ' || professors_table(i).courses(j));
        end loop;
        dbms_output.new_line;
    end loop;
end P6;
/
execute P6;
/
```

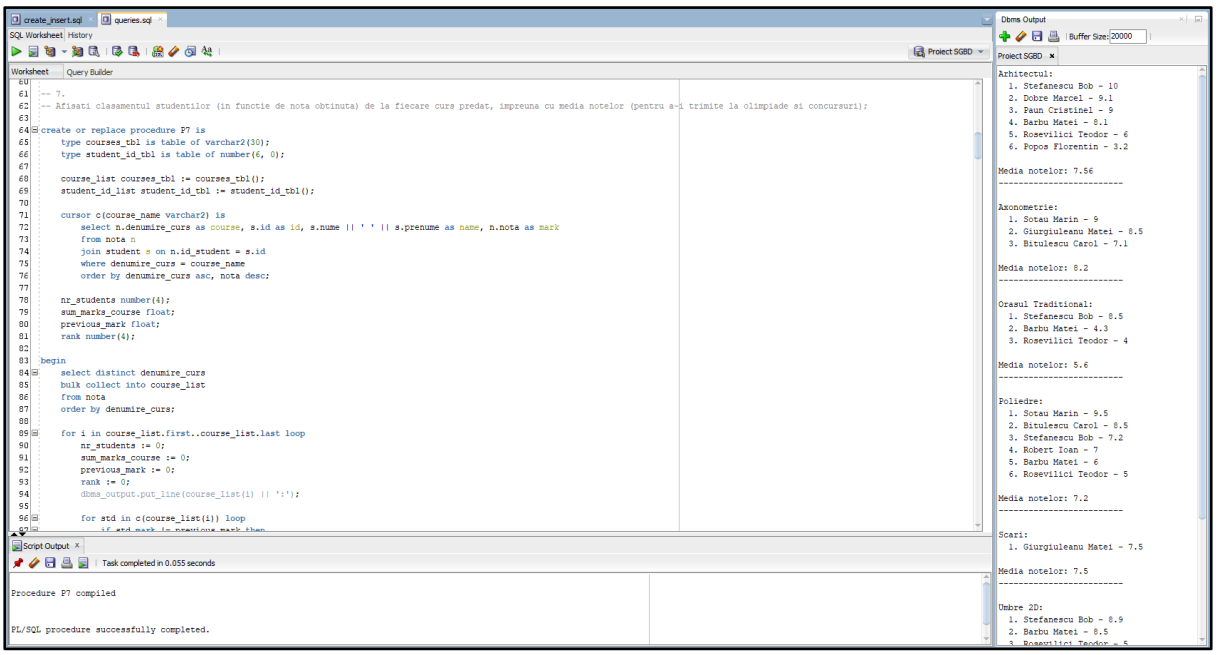
7. Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

"Afisati clasamentul studentilor (in functie de nota obtinuta) de la fiecare curs predat, impreuna cu media notelor (pentru a-i trimite la olimpiade si concursuri)".

Retinem intr-un cursor parametrizat, pentru fiecare curs: id-ul, numele si nota fiecarui student ce a parcurs acest curs.

Pentru fiecare curs din cursor, realizam clasamentul studentilor in functie de nota, iar daca mai multi studenti au aceeasi nota, ii vom clasa pe aceeasi pozitie.

In final, afisam media notelor de la curs.



```oracle
-- 7.
-- Afisati clasamentul studentilor (in functie de nota obtinuta) de la fiecare curs predat, impreuna cu media notelor (pentru a-i trimite la olimpiade si concursuri);

create or replace procedure P7 is
    type courses_tbl is table of varchar2(30);
    type student_id_tbl is table of number(6, 0);

    course_list courses_tbl := courses_tbl();
    student_id_list student_id_tbl := student_id_tbl();

    cursor c(course_name varchar2) is
        select n.denumire_curs as course, s.id as id, s.nume || ' ' || s.prenume as name, n.nota as mark
        from nota n
        join student s on n.id_student = s.id
        where denumire_curs = course_name
        order by denumire_curs asc, nota desc;

    nr_students number(4);
    sum_marks_course float;
    previous_mark float;
    rank number(4);

begin
    select distinct denumire_curs
    bulk collect into course_list
    from nota
    order by denumire_curs;

    for i in course_list.first..course_list.last loop
        nr_students := 0;
        sum_marks_course := 0;
        previous_mark := 0;
        rank := 0;
        dbms_output.put_line(course_list(i) || ':');

        for std in c(course_list(i)) loop
            if std.mark != previous_mark then
                rank := rank + 1;
                previous_mark := std.mark;
            end if;
            dbms_output.put_line('  ' || rank || '. ' || std.name || ' - ' || std.mark);
            nr_students := nr_students + 1;
            sum_marks_course := sum_marks_course + std.mark;
        end loop;
        dbms_output.new_line;
        dbms_output.put_line('Media notelor: ' || trunc(sum_marks_course / nr_students, 2));
        dbms_output.put_line('-------------------------');
        dbms_output.new_line;
    end loop;
end P7;
/
execute P7;
/
```

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.

"Creati o Functie care sa intoarca lista Angajatilor unui Departament, care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul a generat o suma mare de bani)".

Retinem intr-o colectie de tip nested table, numele angajatilor care au fost angajati inainte de data de inceput a proiectului si care au terminat acel proiect aferent unui departament.

Functia primeste ca parametru numele proiectului.

EXCEPTIILE tratate:

- NO_PROJECT_FOUND cand nu exista niciun proiect cu numele dat in evidenta atelierului;
- NO_EMPLOYEES_FOUND cand proiectul a fost realizat de angajati care nu mai lucreaza in prezent in acel departament.

```
-- 8.
-- Creati o Functie care sa intoarca lista Angajatilor unui Departament, care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul
create or replace type emp_tbl as table of varchar2(30);
/

create or replace function P8 (project_name proiect.denumire%type) return varchar2 is
    employees_list emp_tbl := emp_tbl();
    employees_string varchar2(500) := '';
    nr_projects number(4) := 0;

    NO_PROJECT_FOUND exception;
    pragma exception_init (NO_PROJECT_FOUND, -20001);
    NO_EMPLOYEES_FOUND exception;
    pragma exception_init (NO_EMPLOYEES_FOUND, -20002);

begin
    select count(*)
    into nr_projects
    from proiect
    where denumire = project_name;

    if nr_projects = 0 then
        raise NO_PROJECT_FOUND;
    end if;

    select a.nume || ' ' || a.prenume as nume
    bulk collect into employees_list
    from angajat a
    join departament d on a.departament = d.denumire
    join proiect p on d.denumire = p.departament
    where p.denumire = project_name and nvl(p.data_final, SYSDATE) > a.data_angajare
    order by a.id;

    if employees_list.count = 0 then
        raise NO_EMPLOYEES_FOUND;
    end if;
```

```
Function P8 compiled

PL/SQL procedure successfully completed.
```

```
Lista Angajatilor care au lucrat la Proiectul "Macheta Residence":   Dragulescu Raluca

Nu exista niciun proiect cu acest nume!

Proiectul a fost realizat de angajati care nu mai lucreaza in cadrul firmei!
```

```oracle_plsql
-- 8.
-- Creati o Functie care sa intoarca lista Angajatilor unui Departament, care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul a generat o suma mare de bani);
create or replace type emp_tbl as table of varchar2(30);
/

create or replace function P8 (project_name proiect.denumire%type) return varchar2 is
    employees_list emp_tbl := emp_tbl();
    employees_string varchar2(500) := '';
    nr_projects number(4) := 0;

    NO_PROJECT_FOUND exception;
    pragma exception_init (NO_PROJECT_FOUND, -20001);
    NO_EMPLOYEES_FOUND exception;
    pragma exception_init (NO_EMPLOYEES_FOUND, -20002);

begin
    select count(*)
    into nr_projects
    from proiect
    where denumire = project_name;

    if nr_projects = 0 then
        raise NO_PROJECT_FOUND;
    end if;

    select a.nume || ' ' || a.prenume as nume
    bulk collect into employees_list
    from angajat a
    join departament d on a.departament = d.denumire
    join proiect p on d.denumire = p.departament
    where p.denumire = project_name and nvl(p.data_final, SYSDATE) > a.data_angajare
    order by a.id;

    if employees_list.count = 0 then
        raise NO_EMPLOYEES_FOUND;
    end if;

    for i in employees_list.first..employees_list.last loop
        employees_string := employees_string || employees_list(i) || ', ';
    end loop;

    return rtrim(employees_string, ', ');

    exception
        when NO_PROJECT_FOUND then
            dbms_output.put_line('Nu exista niciun proiect cu acest nume!');
            raise_application_error(-20001, 'No Project found that matched the string passed as parameter to P8!');
        when NO_EMPLOYEES_FOUND then
            dbms_output.put_line('Proiectul a fost realizat de angajati care nu mai lucreaza in cadrul firmei!');
            raise_application_error(-20002, 'No Employees found that were employed before the Project started!');
end P8;
/
```

```
53. begin   -- OK;
54.   dbms_output.put_line('Lista Angajatilor care au lucrat la Proiectul "Macheta
    Residence":   ' || P8('Macheta Residence'));
55. end;
56. /
57. begin   -- NO_PROJECT_FOUND;
58.   dbms_output.put_line('Nu exista niciun proiect cu acest nume!' ||
    P8('denumire'));
59. end;
60. /
61. begin   -- NO_EMPLOYEES_FOUND;
62.   dbms_output.put_line('Nu exista niciun angajat in prezent in firma care a
    lucrat la acest proiect!' || P8('Restaurare Castel'));
63. end;
64. /
```

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.

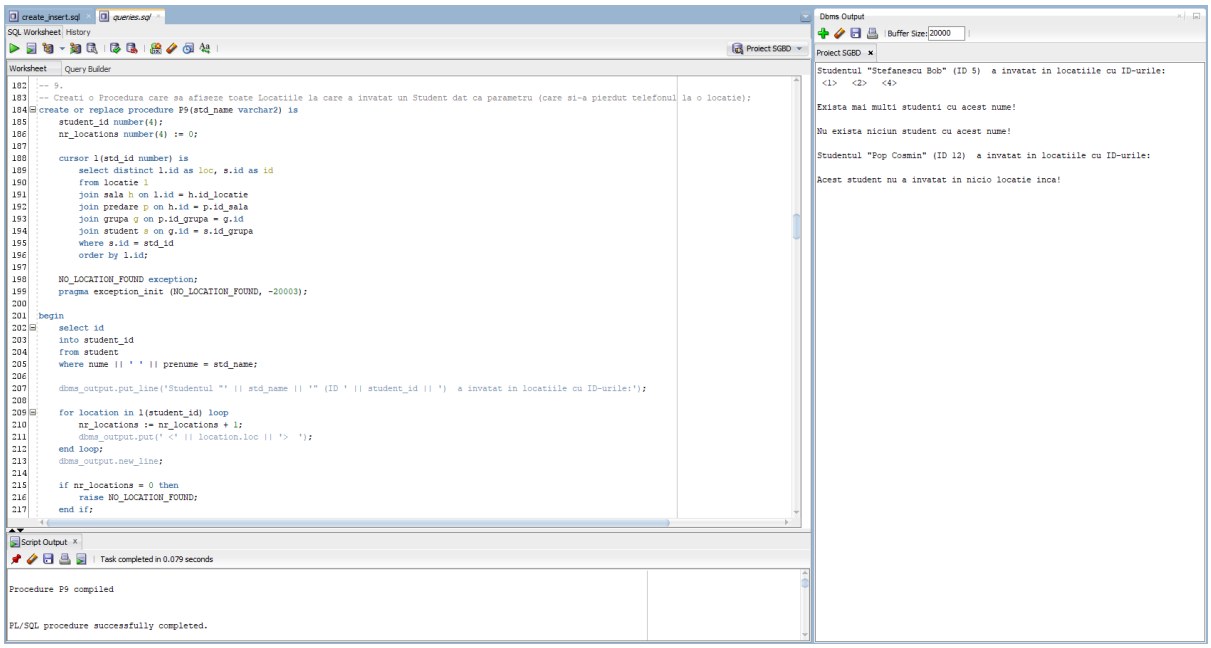
"Creati o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat ca parametru (care si-a pierdut telefonul la o locatie)."

Retinem intr-un cursor parametrizat, pentru fiecare student, locatiile la care a invatat acesta, prin join-uri pe 5 tabele.

Procedura primeste ca parametru numele unui student.

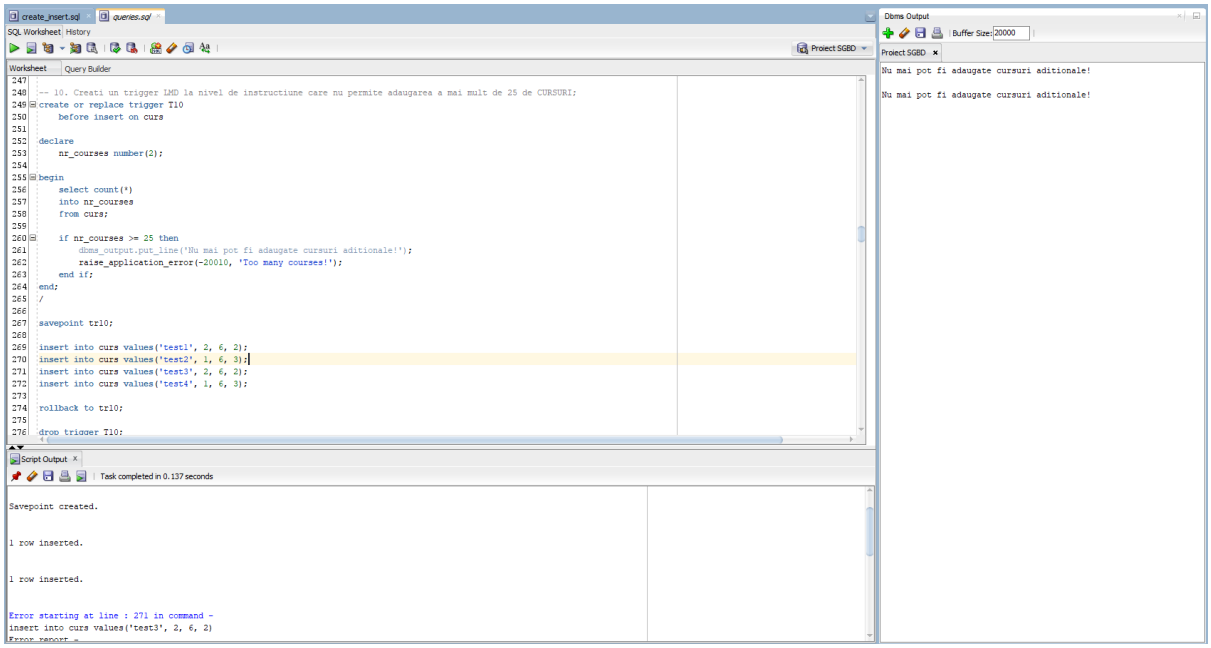
EXCEPTIILE tratate:

- NO_DATA_FOUND cand nu exista studenti cu numele introdus
- TOO_MANY_ROWS cand exista mai multi studenti cu numele introdus
- NO_LOCATION_FOUND cand studentul cu numele introdus nu a invatat la nicio locatie



```oracle_plsql
- 9.
-- Creati o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat ca parametru (care si-a pierdut telefonul la o locatie);
create or replace procedure P9(std_name varchar2) is
    student_id number(4);
    nr_locations number(4) := 0;

    cursor l(std_id number) is
        select distinct l.id as loc, s.id as id
        from locatie l
        join sala h on l.id = h.id_locatie
        join predare p on h.id = p.id_sala
        join grupa g on p.id_grupa = g.id
        join student s on g.id = s.id_grupa
        where s.id = std_id
        order by l.id;

    NO_LOCATION_FOUND exception;
    pragma exception_init (NO_LOCATION_FOUND, -20003);

begin
    select id
    into student_id
    from student
    where nume || ' ' || prenume = std_name;

    dbms_output.put_line('Studentul "' || std_name || '" (ID ' || student_id || ')  a invatat in locatiile cu ID-urile:');

    for location in l(student_id) loop
        nr_locations := nr_locations + 1;
        dbms_output.put(' <' || location.loc || '>  ');
    end loop;
    dbms_output.new_line;

    if nr_locations = 0 then
        raise NO_LOCATION_FOUND;
    end if;

    exception
        when NO_DATA_FOUND then
            dbms_output.put_line('Nu exista niciun student cu acest nume!');
            raise_application_error(-20001, 'Query returned NO ROWS when asked about the student whose name was passed as parameter!');
        when TOO_MANY_ROWS then
            dbms_output.put_line('Exista mai multi studenti cu acest nume!');
            raise_application_error(-20002, 'Query returned MORE THAN ONE ROW when asked about the student whose name was passed as parameter!');
        when NO_LOCATION_FOUND then
            dbms_output.put_line('Acest student nu a invatat in nicio locatie inca!');
            raise_application_error(-20003, 'Query returned NO ROWS when asked about the location a student studied at!');
end P9;
/

-- OK;
execute P9('Stefanescu Bob');
-- TOO_MANY_ROWS;
execute P9('Barbu Matei');
-- NO_DATA_FOUND;
execute P9('nume student');
-- NO_LOCATION_FOUND;
execute P9('Pop Cosmin');
/
```

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




```
1.  -- 10. Creati un trigger LMD la nivel de instructiune care nu permite adaugarea a mai
    mult de 25 de CURSURI;
2.  create or replace trigger T10
3.      before insert on curs
4.
5.  declare
6.      nr_courses number(2);
7.
8.  begin
9.      select count(*)
10.     into nr_courses
11.     from curs;
12.
13.     if nr_courses >= 25 then
14.         dbms_output.put_line('Nu mai pot fi adaugate cursuri aditionale!');
15.         raise_application_error(-20010, 'Too many courses!');
16.     end if;
17. end;
18. /
19.
20. savepoint tr10;
21.
22. insert into curs values('test1', 2, 6, 2);
23. insert into curs values('test2', 1, 6, 3);
24. insert into curs values('test3', 2, 6, 2);
25. insert into curs values('test4', 1, 6, 3);
26.
27. rollback to tr10;
28.
29. drop trigger T10;
30. /
```

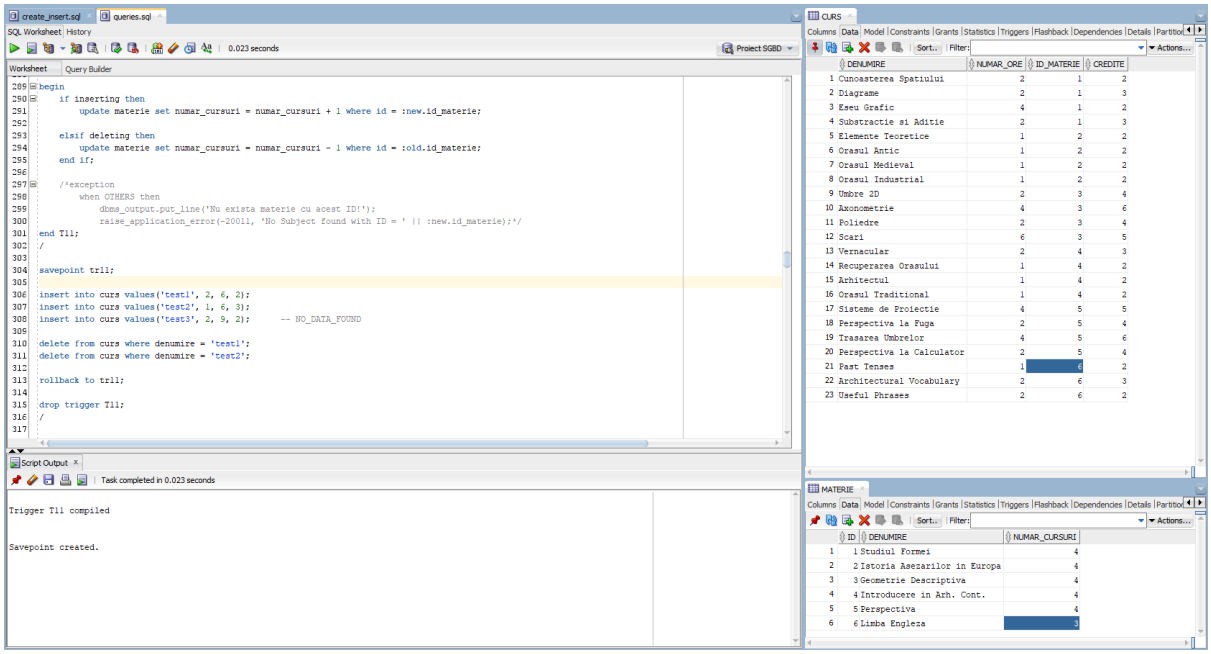
11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

```
-- 11. Creati un trigger LMD la nivel de linie, care actualizeaza tabela MATERIE in functie de operatiile efectuate pe tabela CURS;
create or replace trigger T11
    after insert or delete on curs
    for each row

declare
    subject_id number(2);
    nr_courses number(2);

begin
    if inserting then
        update materie set numar_cursuri = numar_cursuri + 1 where id = :new.id_materie;

    elsif deleting then
        update materie set numar_cursuri = numar_cursuri - 1 where id = :new.id_materie;
    end if;

    /*exception
        when OTHERS then
            dbms_output.put_line('Nu exista materie cu acest ID!');
            raise_application_error(-20011, 'No Subject found with ID = ' || :new.id_materie);*/
end T11;
/

savepoint tr11;

insert into curs values('test1', 2, 6, 2);
insert into curs values('test2', 1, 6, 3);
insert into curs values('test3', 2, 9, 2);      -- NO_DATA_FOUND
```

Script Output — Task completed in 0.078 seconds

```
1 row deleted.


1 row deleted.


Rollback complete.


Trigger T11 dropped.
```

```oracle_plsql
-- 11. Creati un trigger LMD la nivel de linie, care actualizeaza tabela MATERIE in functie de operatiile efectuate pe tabela CURS;
create or replace trigger T11
    after insert or delete on curs
    for each row

declare
    subject_id number(2);
    nr_courses number(2);

begin
    if inserting then
        update materie set numar_cursuri = numar_cursuri + 1 where id = :new.id_materie;

    elsif deleting then
        update materie set numar_cursuri = numar_cursuri - 1 where id = :old.id_materie;
    end if;
end T11;
/

savepoint tr11;

insert into curs values('test1', 2, 6, 2);
insert into curs values('test2', 1, 6, 3);
insert into curs values('test3', 2, 9, 2);      -- parent key not found

delete from curs where denumire = 'test1';
delete from curs where denumire = 'test2';

rollback to tr11;

drop trigger T11;
/
```



```
begin
    if inserting then
        update materie set numar_cursuri = numar_cursuri + 1 where id = :new.id_materie;

    elsif deleting then
        update materie set numar_cursuri = numar_cursuri - 1 where id = :old.id_materie;
    end if;
end T11;
/

savepoint tr11;

insert into curs values('test1', 2, 6, 2);
insert into curs values('test2', 1, 6, 3);
insert into curs values('test3', 2, 9, 2);      -- parent key not found

delete from curs where denumire = 'test1';
delete from curs where denumire = 'test2';

rollback to tr11;

drop trigger T11;
/

-- 12. Creati un trigger LDD, care pastreaza in tabela LOG evenimentele de la nivelul BAZEI DE DATE;
create table LOG(
    baza_date varchar2(20),
    utilizator varchar2(30),
```

Script Output — Task completed in 0.047 seconds

```
1 row inserted.

1 row inserted.

Error starting at line : 308 in command -
insert into curs values('test3', 2, 9, 2)
Error report -
ORA-02291: integrity constraint (GRUIA.SYS_C007037) violated - parent key not found
```

CURS

	DENUMIRE	NUMAR_ORE	ID_MATERIE	CREDITE
1	Cunoasterea Spatiului	2	1	2
2	Diagrame	2	1	3
3	Eseu Grafic	4	1	2
4	Substractie si Aditie	2	1	3
5	Elemente Teoretice	1	2	2
6	Orasul Antic	1	2	2
7	Orasul Medieval	1	2	2
8	Orasul Industrial	1	2	2
9	Umbre 2D	2	3	4
10	Axonometrie	4	3	6
11	Poliedre	2	3	4
12	Scari	6	3	5
13	Vernacular	2	4	3
14	Recuperarea Orasului	1	4	2
15	Arhitectul	1	4	2
16	Orasul Traditional	1	4	2
17	Sisteme de Proiectie	4	5	5
18	Perspectiva la Fuga	2	5	4
19	Trasarea Umbrelor	4	5	6
20	Perspectiva la Calculator	2	5	4
21	Past Tenses	1	6	2
22	Architectural Vocabulary	2	6	3
23	Useful Phrases	2	6	2
24	test2	1	6	3
25	test1	2	6	2

MATERIE

	ID	DENUMIRE	NUMAR_CURSURI
1	1	Studiul Formei	4
2	2	Istoria Asezarilor in Europa	4
3	3	Geometrie Descriptiva	4
4	4	Introducere in Arh. Cont.	4
5	5	Perspectiva	4
6	6	Limba Engleza	5

```
begin
    if inserting then
        update materie set numar_cursuri = numar_cursuri + 1 where id = :new.id_materie;

    elsif deleting then
        update materie set numar_cursuri = numar_cursuri - 1 where id = :old.id_materie;
    end if;
end T11;
/

savepoint tr11;

insert into curs values('test1', 2, 6, 2);
insert into curs values('test2', 1, 6, 3);
insert into curs values('test3', 2, 9, 2);      -- parent key not found

delete from curs where denumire = 'test1';
delete from curs where denumire = 'test2';

rollback to tr11;

drop trigger T11;
/

-- 12. Creati un trigger LDD, care pastreaza in tabela LOG evenimentele de la nivelul BAZEI DE DATE;
create table LOG(
    baza_date varchar2(20),
    utilizator varchar2(30),
```

Script Output — Task completed in 0.034 seconds

```
Error starting at line : 308 in command -
insert into curs values('test3', 2, 9, 2)
Error report -
ORA-02291: integrity constraint (GRUIA.SYS_C007037) violated - parent key not found

1 row deleted.

1 row deleted.
```

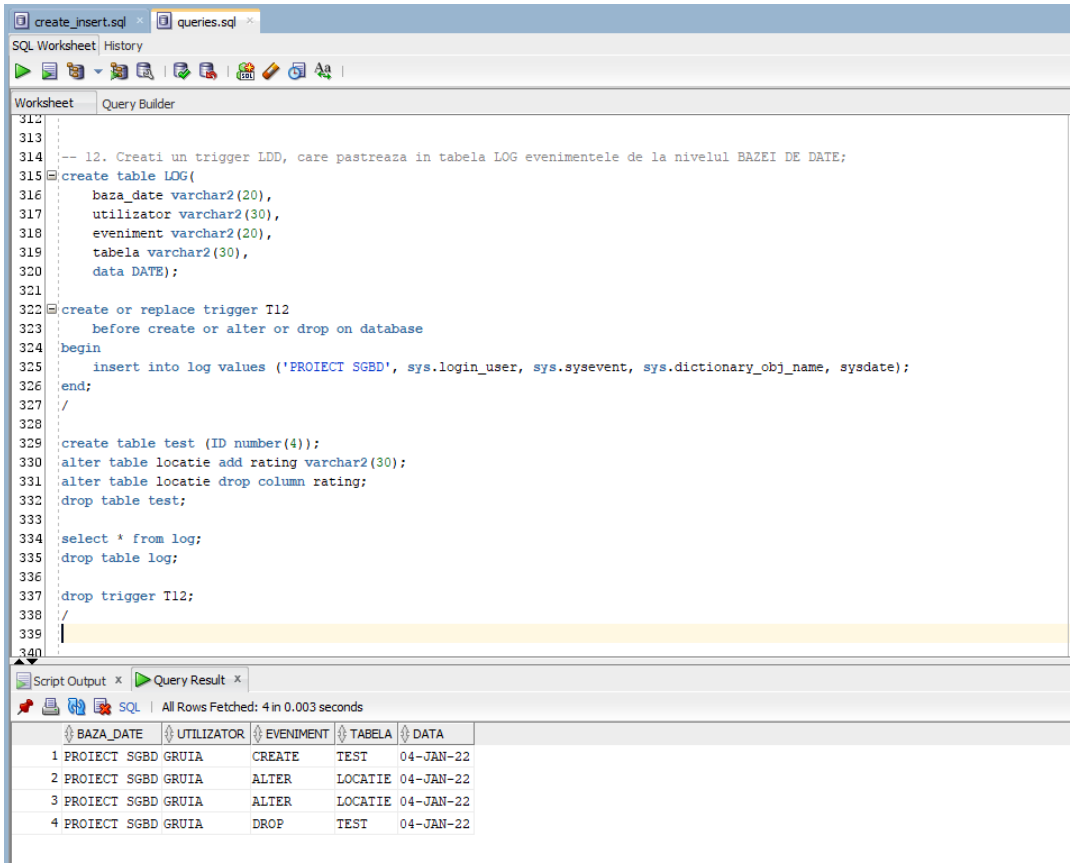
CURS

	DENUMIRE	NUMAR_ORE	ID_MATERIE	CREDITE
1	Cunoasterea Spatiului	2	1	2
2	Diagrame	2	1	3
3	Eseu Grafic	4	1	2
4	Substractie si Aditie	2	1	3
5	Elemente Teoretice	1	2	2
6	Orasul Antic	1	2	2
7	Orasul Medieval	1	2	2
8	Orasul Industrial	1	2	2
9	Umbre 2D	2	3	4
10	Axonometrie	4	3	6
11	Poliedre	2	3	4
12	Scari	6	3	5
13	Vernacular	2	4	3
14	Recuperarea Orasului	1	4	2
15	Arhitectul	1	4	2
16	Orasul Traditional	1	4	2
17	Sisteme de Proiectie	4	5	5
18	Perspectiva la Fuga	2	5	4
19	Trasarea Umbrelor	4	5	6
20	Perspectiva la Calculator	2	5	4
21	Past Tenses	1	6	2
22	Architectural Vocabulary	2	6	3
23	Useful Phrases	2	6	2

MATERIE

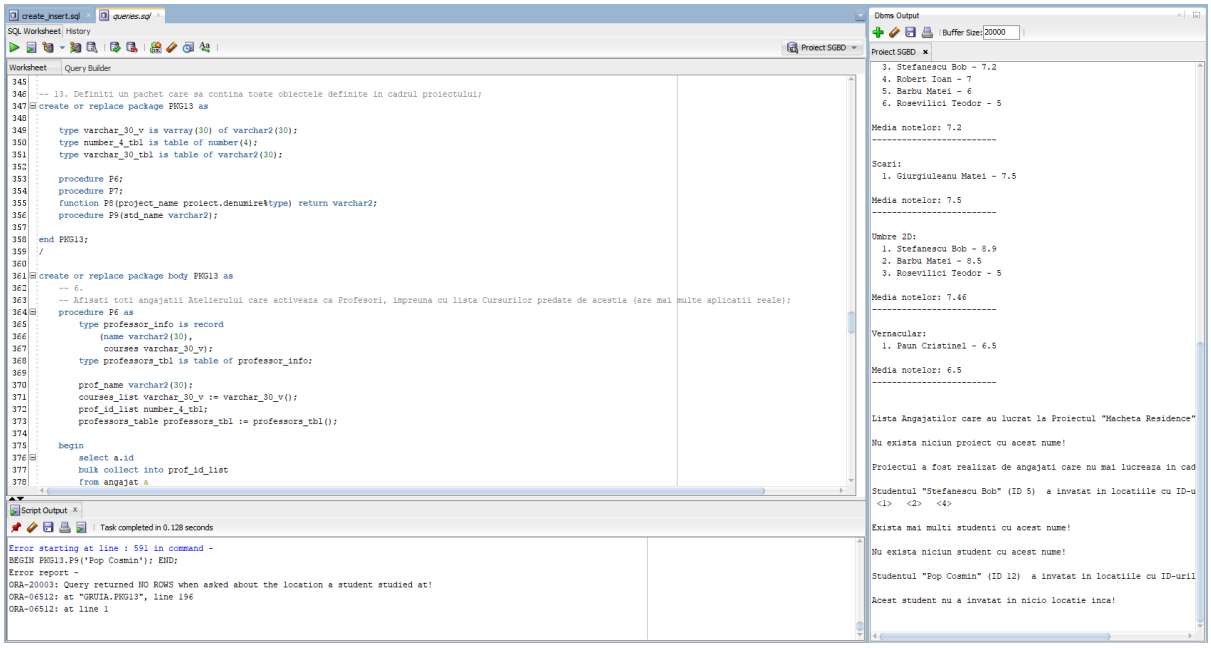
	ID	DENUMIRE	NUMAR_CURSURI
1	1	Studiul Formei	4
2	2	Istoria Asezarilor in Europa	4
3	3	Geometrie Descriptiva	4
4	4	Introducere in Arh. Cont.	4
5	5	Perspectiva	4
6	6	Limba Engleza	3

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




```
1.  -- 12. Creati un trigger LDD, care pastreaza in tabela LOG evenimentele de la
    nivelul BAZEI DE DATE;
2.  create table LOG(
3.      baza_date varchar2(20),
4.      utilizator varchar2(30),
5.      eveniment varchar2(20),
6.      tabela varchar2(30),
7.      data DATE);
8.
9.  create or replace trigger T12
10.     before create or alter or drop on database
11. begin
12.     insert into log values ('PROIECT SGBD', sys.login_user, sys.sysevent,
    sys.dictionary_obj_name, sysdate);
13. end;
14. /
15.
16. create table test (ID number(4));
17. alter table locatie add rating varchar2(30);
18. alter table locatie drop column rating;
19. drop table test;
20.
21. select * from log;
22. drop table log;
23. drop trigger T12;
24. /
```

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



```
1.  -- 13. Definiti un pachet care sa contina toate obiectele definite in cadrul
    proiectului;
2.  create or replace package PKG13 as
3.
4.      type varchar_30_v is varray(30) of varchar2(30);
5.      type number_4_tbl is table of number(4);
6.      type varchar_30_tbl is table of varchar2(30);
7.
8.      procedure P6;
9.      procedure P7;
10.     function P8(project_name proiect.denumire%type) return varchar2;
11.     procedure P9(std_name varchar2);
12.
13. end PKG13;
14. /
15.
16. create or replace package body PKG13 as
17.     -- 6.
18.     -- Afisati toti angajatii Atelierului care activeaza ca Profesori,
    impreuna cu lista Cursurilor predate de acestia (are mai multe aplicatii
    reale);
19.     procedure P6 as
20.         type professor_info is record
21.             (name varchar2(30),
22.              courses varchar_30_v);
23.         type professors_tbl is table of professor_info;
24.
25.         prof_name varchar2(30);
26.         courses_list varchar_30_v := varchar_30_v();
```

```
        prof_id_list number_4_tbl;
        professors_table professors_tbl := professors_tbl();

    begin
        select a.id
        bulk collect into prof_id_list
        from angajat a
        join job j on a.id_job = j.id
        where j.denumire = 'Profesor';

        professors_table.extend(prof_id_list.count);

        for i in prof_id_list.first..prof_id_list.last loop
            select nume || ' ' || prenume
            into prof_name
            from angajat
            where id = prof_id_list(i);

            select distinct nume_curs
            bulk collect into courses_list
            from predare
            where id_profesor = prof_id_list(i);

            professors_table(i).name := prof_name;
            professors_table(i).courses := courses_list;
        end loop;

        dbms_output.put_line('Lista profesorilor si a cursurilor pe care le sustin:');
        dbms_output.new_line;
        dbms_output.new_line;

        for i in professors_table.first..professors_table.last loop
            dbms_output.put_line(i || ') ' || professors_table(i).name || ' (ID ' || prof_id_list(i) || ') preda: ');
            for j in professors_table(i).courses.first..professors_table(i).courses.last loop
                dbms_output.put_line('  - ' || professors_table(i).courses(j));
            end loop;
            dbms_output.new_line;
        end loop;
    end P6;


    -- 7.
    -- Afisati clasamentul studentilor (in functie de nota obtinuta) de la fiecare curs predat, impreuna cu media notelor (pentru a-i trimite la olimpiade si concursuri);
    procedure P7 as
        course_list varchar_30_tbl := varchar_30_tbl();

        cursor c(course_name varchar2) is
            select n.denumire_curs as course, s.id as id, s.nume || ' ' || s.prenume as name, n.nota as mark
            from nota n
            join student s on n.id_student = s.id
            where denumire_curs = course_name
            order by denumire_curs asc, nota desc;
```

```
79.
80.         nr_students number(4);
81.         sum_marks_course float;
82.         previous_mark float;
83.         rank number(4);
84.
85.     begin
86.         select distinct denumire_curs
87.         bulk collect into course_list
88.         from nota
89.         order by denumire_curs;
90.
91.         for i in course_list.first..course_list.last loop
92.             nr_students := 0;
93.             sum_marks_course := 0;
94.             previous_mark := 0;
95.             rank := 0;
96.             dbms_output.put_line(course_list(i) || ':');
97.
98.             for std in c(course_list(i)) loop
99.                 if std.mark != previous_mark then
100.                    rank := rank + 1;
101.                    previous_mark := std.mark;
102.                end if;
103.                dbms_output.put_line('  ' || rank || '. ' || std.name || ' -
    ' || std.mark);
104.                nr_students := nr_students + 1;
105.                sum_marks_course := sum_marks_course + std.mark;
106.            end loop;
107.            dbms_output.new_line;
108.            dbms_output.put_line('Media notelor: ' || trunc(sum_marks_course
    / nr_students, 2));
109.            dbms_output.put_line('-------------------------');
110.            dbms_output.new_line;
111.        end loop;
112.    end P7;
113.
114.
115.    -- 8.
116.    -- Creati o Functie care sa intoarca lista Angajatilor unui Departament,
    care au lucrat la un Proiect trimis ca parametru (pentru ca proiectul a
    generat o suma mare de bani);
117.    function P8 (project_name proiect.denumire%type) return varchar2 as
118.        employees_list varchar_30_tbl := varchar_30_tbl();
119.        employees_string varchar2(500) := '';
120.        nr_projects number(4) := 0;
121.
122.        NO_PROJECT_FOUND exception;
123.        pragma exception_init (NO_PROJECT_FOUND, -20001);
124.        NO_EMPLOYEES_FOUND exception;
125.        pragma exception_init (NO_EMPLOYEES_FOUND, -20002);
126.
127.    begin
128.        select count(*)
129.        into nr_projects
130.        from proiect
131.        where denumire = project_name;
132.
133.        if nr_projects = 0 then
```

```
            raise NO_PROJECT_FOUND;
        end if;

        select a.nume || ' ' || a.prenume as nume
        bulk collect into employees_list
        from angajat a
        join departament d on a.departament = d.denumire
        join proiect p on d.denumire = p.departament
        where p.denumire = project_name and nvl(p.data_final, SYSDATE) > a.data_angajare
        order by a.id;

        if employees_list.count = 0 then
            raise NO_EMPLOYEES_FOUND;
        end if;

        for i in employees_list.first..employees_list.last loop
            employees_string := employees_string || employees_list(i) || ', ';
        end loop;

        return rtrim(employees_string, ', ');

        exception
            when NO_PROJECT_FOUND then
                dbms_output.put_line('Nu exista niciun proiect cu acest nume!');
                raise_application_error(-20001, 'No Project found that matched the string passed as parameter to P8!');
            when NO_EMPLOYEES_FOUND then
                dbms_output.put_line('Proiectul a fost realizat de angajati care nu mai lucreaza in cadrul firmei!');
                raise_application_error(-20002, 'No Employees found that were employed before the Project started!');
    end P8;


    -- 9.
    -- Creati o Procedura care sa afiseze toate Locatiile la care a invatat un Student dat ca parametru (care si-a pierdut telefonul la o locatie);
    procedure P9(std_name varchar2) as
        student_id number(4);
        nr_locations number(4) := 0;

        cursor l(std_id number) is
            select distinct l.id as loc, s.id as id
            from locatie l
            join sala h on l.id = h.id_locatie
            join predare p on h.id = p.id_sala
            join grupa g on p.id_grupa = g.id
            join student s on g.id = s.id_grupa
            where s.id = std_id
            order by l.id;

        NO_LOCATION_FOUND exception;
        pragma exception_init (NO_LOCATION_FOUND, -20003);

    begin
        select id
```

```
186.         into student_id
187.         from student
188.         where nume || ' ' || prenume = std_name;
189.
190.         dbms_output.put_line('Studentul "' || std_name || '" (ID ' ||
   student_id || ')  a invatat in locatiile cu ID-urile:');
191.
192.         for location in l(student_id) loop
193.             nr_locations := nr_locations + 1;
194.             dbms_output.put(' <' || location.loc || '>  ');
195.         end loop;
196.         dbms_output.new_line;
197.
198.         if nr_locations = 0 then
199.             raise NO_LOCATION_FOUND;
200.         end if;
201.
202.         exception
203.             when NO_DATA_FOUND then
204.                 dbms_output.put_line('Nu exista niciun student cu acest
   nume!');
205.                 raise_application_error(-20001, 'Query returned NO ROWS when
   asked about the student whose name was passed as parameter!');
206.             when TOO_MANY_ROWS then
207.                 dbms_output.put_line('Exista mai multi studenti cu acest
   nume!');
208.                 raise_application_error(-20002, 'Query returned MORE THAN
   ONE ROW when asked about the student whose name was passed as parameter!');
209.             when NO_LOCATION_FOUND then
210.                 dbms_output.put_line('Acest student nu a invatat in nicio
   locatie inca!');
211.                 raise_application_error(-20003, 'Query returned NO ROWS when
   asked about the location a student studied at!');
212.     end P9;
213.
214. end PKG13;
215. /
216.
217. -- TESTAM PACHETUL PKG13;
218. -- 6
219. execute PKG13.P6;
220.
221. -- 7
222. execute PKG13.P7;
223.
224. -- 8
225. begin   -- OK;
226.   dbms_output.put_line('Lista Angajatilor care au lucrat la Proiectul
   "Macheta Residence":   ' || PKG13.P8('Macheta Residence'));
227. end;
228. /
229. begin   -- NO_PROJECT_FOUND;
230.   dbms_output.put_line('Nu exista niciun proiect cu acest nume!' ||
   PKG13.P8('denumire'));
231. end;
232. /
233. begin   -- NO_EMPLOYEES_FOUND;
234.   dbms_output.put_line('Nu exista niciun angajat in prezent in firma care a
   lucrat la acest proiect!' || PKG13.P8('Restaurare Castel'));
```

```
end;
/

-- 9
-- OK;
execute PKG13.P9('Stefanescu Bob');
-- TOO_MANY_ROWS;
execute PKG13.P9('Barbu Matei');
-- NO_DATA_FOUND;
execute PKG13.P9('nume student');
-- NO_LOCATION_FOUND;
execute PKG13.P9('Pop Cosmin');
```

Va multumesc pentru atentia acordata!

Student: Gabriel GRUIA

Grupa: 231

Data: 05.01.2021