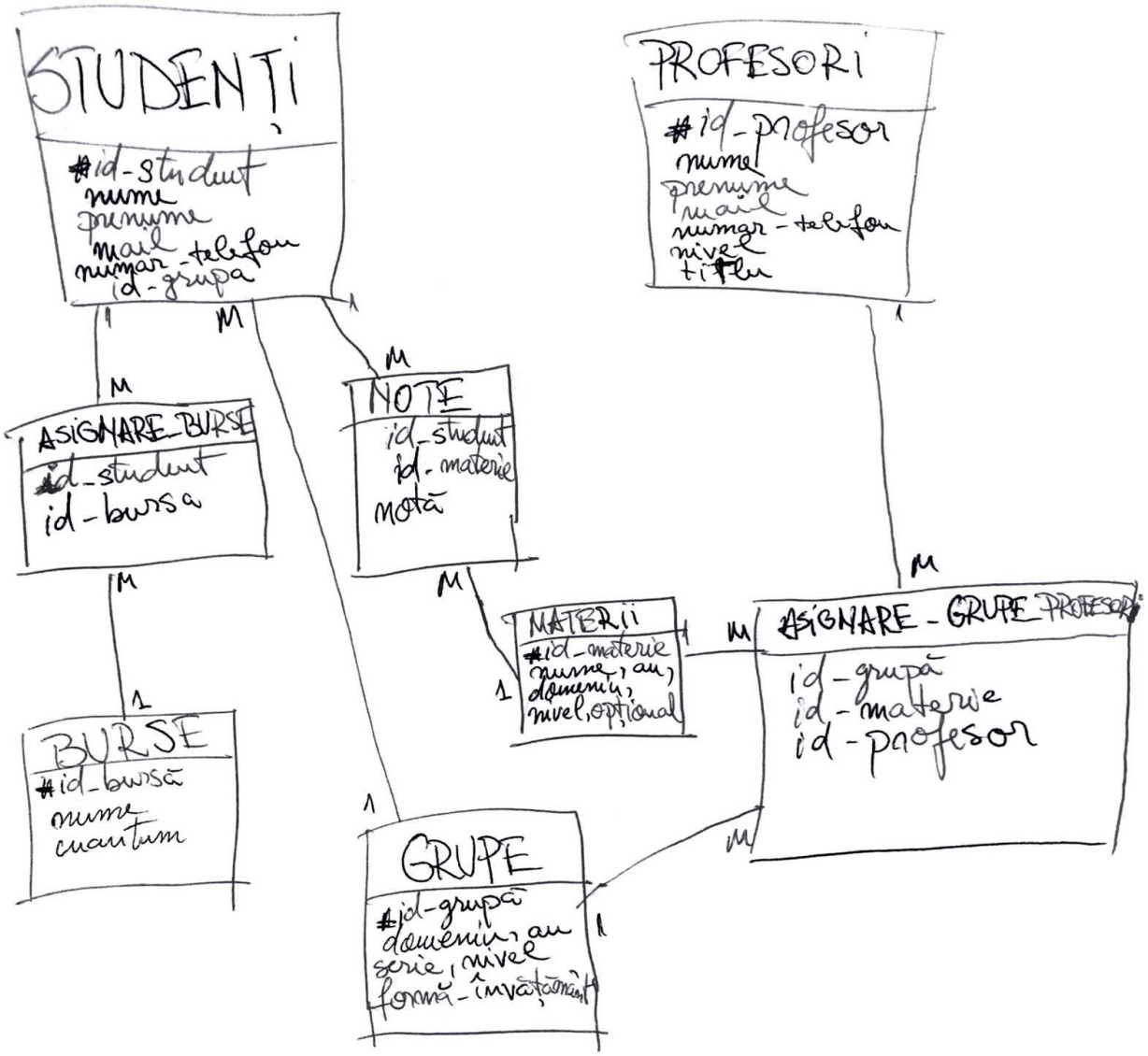
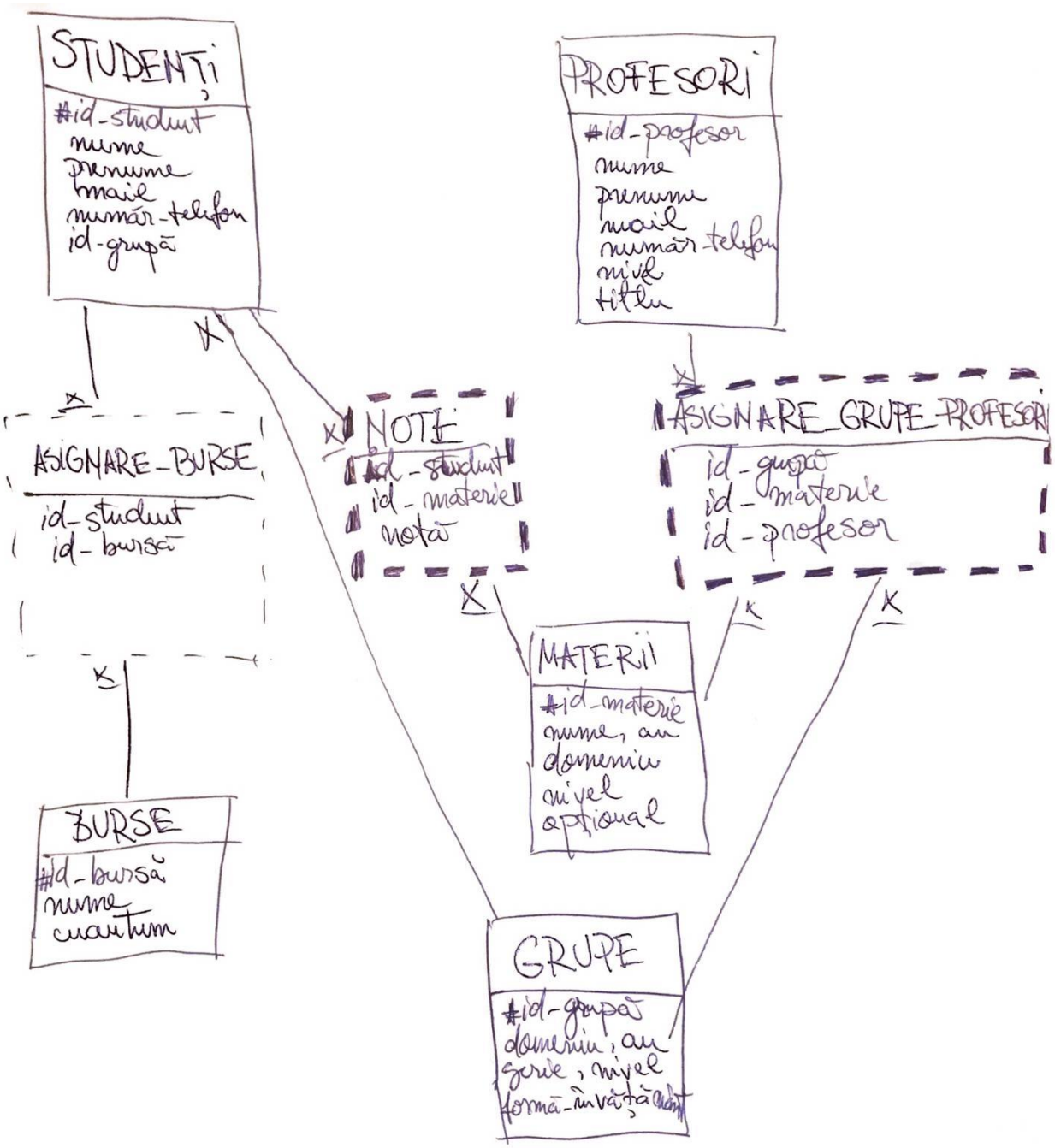
1. Prezentare baza de date

Am creat o baza de date care retine date despre studentii, notele obtinute de acestia, materiile predate, organizarea si profesorii unei facultati. Baza de date are rolul de a usura calcularea anumitor aspecte legate de note, cum ar fi medii sau burse. Avem entitatea asignare_burse care retine ce bursa are fiecare student(dintre cei care au burse). Totodata baza de date are rol si de a tine o evidenta a studentilor, profesorilor, a datelor acestora si a incadrarii pe grupe. Astfel, in entitatea asignare_grupe_profesori retinem la ce grupa este asignat un profesor si ce materie preda el acolo. Profesorii pot preda mai multe materii la diverse grupe.

2. Diagrama ER



3. Diagrama conceptuala



4. Implementarea diagramei in oracle

```
CREATE TABLE studenti(id_student NUMBER(10) PRIMARY KEY, nume VARCHAR(50) NOT NULL,
prenume VARCHAR(50) NOT NULL, mail VARCHAR(200),
        numar_telefon VARCHAR(20),id_grupa NUMBER(10));
CREATE TABLE profesori(id_profesor NUMBER(10) PRIMARY KEY, nume VARCHAR(50) NOT
NULL, prenume VARCHAR(50) NOT NULL, mail VARCHAR(200),
        numar_telefon VARCHAR(20), nivel VARCHAR(50), titlu VARCHAR(50));
CREATE TABLE materii(id_materie NUMBER(10) PRIMARY KEY, nume_materie VARCHAR(50)
NOT NULL, an NUMBER(10), domeniu VARCHAR(50),
        nivel VARCHAR(50), optional VARCHAR(20));
CREATE TABLE burse(id_bursa NUMBER(10) PRIMARY KEY, nume_bursa VARCHAR(50) NOT
NULL, cuantum NUMBER(10));
CREATE TABLE grupe(id_grupa NUMBER(10) PRIMARY KEY, serie NUMBER(10), an NUMBER(10),
domeniu VARCHAR(50), forma_invatamant VARCHAR(20), nivel VARCHAR(50));

CREATE TABLE note(id_student NUMBER(10), id_materie NUMBER(10), nota NUMBER(10),
        CONSTRAINT fk_studenti FOREIGN KEY(id_student) REFERENCES
studenti(id_student),
        CONSTRAINT fk_materii FOREIGN KEY(id_materie) REFERENCES
materii(id_materie));
CREATE TABLE asignare_grupe_profesori(id_grupa NUMBER(10), id_materie NUMBER(10),
id_profesor NUMBER(10),
                CONSTRAINT fk_grupa FOREIGN KEY(id_grupa) REFERENCES
grupe(id_grupa),
                CONSTRAINT fk_materii2 FOREIGN KEY(id_materie) REFERENCES
materii(id_materie),
                CONSTRAINT fk_profesor FOREIGN KEY(id_profesor) REFERENCES
profesori(id_profesor));
CREATE TABLE asignare_burse(id_student NUMBER(10), id_bursa NUMBER(10),
            CONSTRAINT fk_studenti2 FOREIGN KEY(id_student) REFERENCES
studenti(id_student),
            CONSTRAINT fk_bursa FOREIGN KEY(id_bursa) REFERENCES burse(id_bursa));
```

Script Output x | Query Result x

Task completed in 0.041 seconds

```
Table STUDENTI created.

Table PROFESORI created.

Table MATERII created.

Table BURSE created.

Table GRUPE created.

Table NOTE created.

Table ASIGNARE_GRUPE_PROFESORI created.

Table ASIGNARE_BURSE created.
```

5. Populare tabele

```
INSERT INTO studenti
VALUES(1, 'Aciu', 'Lia', NULL, '0729027022', 234);

INSERT INTO studenti
VALUES(2, 'Baltag', 'Octavian', NULL, '0795002156', 106);

INSERT INTO studenti
VALUES(3, 'Constantinescu', 'Mircea', NULL, '0782491235', 334);

INSERT INTO studenti
VALUES(4, 'Darie', 'Eleonora', NULL, '0759825500', 441);

INSERT INTO studenti
VALUES(5, 'Manea', 'Lucian', NULL, '0765395342', 236);

INSERT INTO studenti
VALUES(21, 'Manea', 'Lucian', NULL, '0725395345', 334);

INSERT INTO studenti
VALUES(22, 'Popa', 'Daniel', NULL, '0757695345', 334);

INSERT INTO studenti
VALUES(8, 'Cobzaru', 'Alina', NULL, '0765395342', 334);
INSERT INTO profesori
VALUES (1, 'Morariu', 'Nicolae', NULL, '0710049511', 'dr.', 'lector');

INSERT INTO profesori
VALUES (2, 'Nicoara', 'Tania', NULL, '0702658959', 'drd.', 'asistent');

INSERT INTO profesori
VALUES (3, 'Ofrim', 'Dragos', NULL, '0787013325', 'dr.', 'profesor');

INSERT INTO profesori
VALUES (4, 'Soimu', 'Andreea', NULL, '0726681537', 'dr.', 'conferentiar');

INSERT INTO profesori
VALUES (5, 'Ursea', 'Angela', NULL, '0716023455', 'drd.', 'asistent');

INSERT INTO profesori
VALUES (6, 'Voicu', 'Gabriela', NULL, '0716023455', 'dr.', 'conferentiar');
INSERT INTO materii VALUES(1, 'Analiza I', 1, 'Matematica', 'Licenta', 'Nu');
```

```
INSERT INTO materii VALUES(2, 'SGBD', 2, 'Informatica', 'Licenta', 'Nu');
INSERT INTO materii VALUES(3, 'Artificial Vision', 3, 'Informatica', 'Licenta', 'Da');
INSERT INTO materii VALUES(4, 'Sisteme de operare', 1, 'Securitate', 'Master', 'Nu');
INSERT INTO materii VALUES(5, 'DAW', 2, 'Informatica', 'Licenta', 'Da');
INSERT INTO materii VALUES(6, 'Programare Functionala', 2, 'Informatica', 'Licenta', 'Nu');
INSERT INTO materii VALUES(7, 'Inteligenta Artificiala', 2, 'Informatica', 'Licenta', 'Nu');
INSERT INTO materii VALUES(8, 'Algoritmi fundamentali', 2, 'Informatica', 'Licenta', 'Nu');

INSERT INTO burse VALUES(1, 'Merit I', 750);
INSERT INTO burse VALUES(2, 'Merit II', 850);
INSERT INTO burse VALUES(3, 'Sociala', 650);
INSERT INTO burse VALUES(4, 'Cercetare', 1000);

INSERT INTO grupe VALUES(106, 10, 1, 'Matematica', 'IF', 'Licenta');
INSERT INTO grupe VALUES(236, 23, 2, 'Informatica', 'ID', 'Licenta');
INSERT INTO grupe VALUES(243, 24, 2, 'Informatica', 'IF', 'Licenta');
INSERT INTO grupe VALUES(334, 34, 3, 'Informatica', 'IF', 'Licenta');
INSERT INTO grupe VALUES(441, 44, 1, 'Securitate', 'IF', 'Master');

INSERT INTO note VALUES(1, 2, 10);
INSERT INTO note VALUES(5, 2, 9);
INSERT INTO note VALUES(1, 5, 10);
INSERT INTO note VALUES(5, 5, 8);
INSERT INTO note VALUES(5, 6, 6);
INSERT INTO note VALUES(1, 6, 9);
INSERT INTO note VALUES(1, 7, 10);
INSERT INTO note VALUES(5, 7, 10);
INSERT INTO note VALUES(1, 8, 10);
INSERT INTO note VALUES(5, 8, 8);
INSERT INTO note VALUES (2,1,10);
INSERT INTO note VALUES (3,3,9);
INSERT INTO note VALUES (4,4,8);
INSERT INTO note VALUES (8,3,10);
INSERT INTO note VALUES (21,3,9);
INSERT INTO note VALUES (22,3,10);

INSERT INTO asignare_grupe_profesori VALUES (243, 2, 1);
INSERT INTO asignare_grupe_profesori VALUES (236, 2, 1);
INSERT INTO asignare_grupe_profesori VALUES (243, 8, 2);
INSERT INTO asignare_grupe_profesori VALUES (236, 8, 3);
INSERT INTO asignare_grupe_profesori VALUES (243, 5, 4);
INSERT INTO asignare_grupe_profesori VALUES (236, 6, 4);
INSERT INTO asignare_grupe_profesori VALUES (243, 6, 5);
INSERT INTO asignare_grupe_profesori VALUES (236, 5, 6);
```

```
INSERT INTO asignare_grupe_profesori VALUES (441, 4, 6);
INSERT INTO asignare_grupe_profesori VALUES (334, 3, 5);


INSERT INTO asignare_burse VALUES (1,1);
INSERT INTO asignare_burse VALUES (1,3);
INSERT INTO asignare_burse VALUES (4,4);
INSERT INTO asignare_burse VALUES (4,2);
INSERT INTO asignare_burse VALUES (2,3);
INSERT INTO asignare_burse VALUES (8,3);
INSERT INTO asignare_burse VALUES (8,2);
INSERT INTO asignare_burse VALUES (3,1);
INSERT INTO asignare_burse VALUES (5,3);
INSERT INTO asignare_burse VALUES (1,4);
INSERT INTO asignare_burse VALUES (1,2);
INSERT INTO asignare_burse VALUES (2,2);
INSERT INTO asignare_burse VALUES (4,1);
INSERT INTO asignare_burse VALUES (5,1);
```

Worksheet | Query Builder

```
VALUES (1, 'Morariu', 'Nicolae', NULL, '0710049511', 'dr.', 'lector');

INSERT INTO profesori
VALUES (2, 'Nicoara', 'Tania', NULL, '0702658959', 'drd.', 'asistent');

INSERT INTO profesori
VALUES (3, 'Ofrim', 'Dragos', NULL, '0787013325', 'dr.', 'profesor');

INSERT INTO profesori
VALUES (4, 'Soimu', 'Andreea', NULL, '0726681537', 'dr.', 'conferentiar');

INSERT INTO profesori
VALUES (5, 'Ursea', 'Angela', NULL, '0716023455', 'drd.', 'asistent');

INSERT INTO profesori
```

Script Output | Query Result

Task completed in 0.034 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

seq 1 of 12

Worksheet Query Builder

```
VALUES(4, 'Darie', 'Eleonora', NULL, '0759825500', 441);

INSERT INTO studenti
VALUES(5, 'Manea', 'Lucian', NULL, '0765395342', 236);

INSERT INTO studenti
VALUES(21, 'Manea', 'Lucian', NULL, '0725395345', 334);

INSERT INTO studenti
VALUES(22, 'Popa', 'Daniel', NULL, '0757695345', 334);

INSERT INTO studenti
VALUES(8, 'Cobzaru', 'Alina', NULL, '0765395342', 334);

CREATE SEQUENCE prof_seq
```

Script Output | Query Result

Task completed in 0.034 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

```
77 INSERT INTO materii VALUES(1, 'Analiza I', 1, 'Matematica', 'Licenta', 'Nu');
78 INSERT INTO materii VALUES(2, 'SGBD', 2, 'Informatica', 'Licenta', 'Nu');
79 INSERT INTO materii VALUES(3, 'Artificial Vision', 3, 'Informatica', 'Licenta', 'Da');
80 INSERT INTO materii VALUES(4, 'Sisteme de operare', 1, 'Securitate', 'Master', 'Nu');
81 INSERT INTO materii VALUES(5, 'DAW', 2, 'Informatica', 'Licenta', 'Da');
82 INSERT INTO materii VALUES(6, 'Programare Functionala', 2, 'Informatica', 'Licenta', 'Nu');
83 INSERT INTO materii VALUES(7, 'Inteligenta Artificiala', 2, 'Informatica', 'Licenta', 'Nu');
84 INSERT INTO materii VALUES(8, 'Algoritmi fundamentali', 2, 'Informatica', 'Licenta', 'Nu');
85
86 CREATE SEQUENCE burse_seq
87     START WITH 1
88     INCREMENT BY 1;
89
90 INSERT INTO burse VALUES(1, 'Merit I', 750);
```

Script Output | Query Result

Task completed in 0.084 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

```
89
90 INSERT INTO burse VALUES(1, 'Merit I', 750);
91 INSERT INTO burse VALUES(2, 'Merit II', 850);
92 INSERT INTO burse VALUES(3, 'Sociala', 650);
93 INSERT INTO burse VALUES(4, 'Cercetare', 1000);
```

Script Output | Query Result

Task completed in 0.056 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

```
INSERT INTO grupe VALUES(106, 10, 1, 'Matematica', 'IF', 'Licenta');
INSERT INTO grupe VALUES(236, 23, 2, 'Informatica', 'ID', 'Licenta');
INSERT INTO grupe VALUES(243, 24, 2, 'Informatica', 'IF', 'Licenta');
INSERT INTO grupe VALUES(334, 34, 3, 'Informatica', 'IF', 'Licenta');
INSERT INTO grupe VALUES(441, 44, 1, 'Securitate', 'IF', 'Master');

INSERT INTO note VALUES(1, 2, 10);
INSERT INTO note VALUES(5, 2, 9);
INSERT INTO note VALUES(1, 5, 10);
```

Script Output | Query Result

Task completed in 0.071 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

```
INSERT INTO note VALUES(1, 6, 9);
INSERT INTO note VALUES(1, 7, 10);
INSERT INTO note VALUES(5, 7, 10);
INSERT INTO note VALUES(1, 8, 10);
INSERT INTO note VALUES(5, 8, 8);
INSERT INTO note VALUES (2,1,10);
INSERT INTO note VALUES (3,3,9);
INSERT INTO note VALUES (4,4,8);
INSERT INTO note VALUES (8,3,10);
INSERT INTO note VALUES (21,3,9);
INSERT INTO note VALUES (22,3,10);
```

Script Output | Query Result

Task completed in 0.104 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Worksheet | Query Builder

```
118 INSERT INTO asignare_grupe_profesori VALUES (243, 2, 1);
119 INSERT INTO asignare_grupe_profesori VALUES (236, 2, 1);
120 INSERT INTO asignare_grupe_profesori VALUES (243, 8, 2);
121 INSERT INTO asignare_grupe_profesori VALUES (236, 8, 3);
122 INSERT INTO asignare_grupe_profesori VALUES (243, 5, 4);
123 INSERT INTO asignare_grupe_profesori VALUES (236, 6, 4);
124 INSERT INTO asignare_grupe_profesori VALUES (243, 6, 5);
125 INSERT INTO asignare_grupe_profesori VALUES (236, 5, 6);
126 INSERT INTO asignare_grupe_profesori VALUES (441, 4, 6);
127 INSERT INTO asignare_grupe_profesori VALUES (334, 3, 5);
128
129
```

Script Output | Query Result

Task completed in 0.085 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

Worksheet | Query Builder

```
INSERT INTO asignare_burse VALUES (4,4);
INSERT INTO asignare_burse VALUES (4,2);
INSERT INTO asignare_burse VALUES (2,3);
INSERT INTO asignare_burse VALUES (8,3);
INSERT INTO asignare_burse VALUES (8,2);
INSERT INTO asignare_burse VALUES (3,1);
INSERT INTO asignare_burse VALUES (5,3);
INSERT INTO asignare_burse VALUES (1,4);
INSERT INTO asignare_burse VALUES (1,2);
INSERT INTO asignare_burse VALUES (2,2);
INSERT INTO asignare_burse VALUES (4,1);
```

Script Output | Query Result

Task completed in 0.089 seconds

```
1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.

1 row inserted.
```

6. procedura care genereaza mail-ul institutional al studentilor

```
CREATE OR REPLACE PROCEDURE generare_mail_studenti
AS
   TYPE ti IS TABLE OF studenti.id_student%TYPE;
   TYPE tn IS TABLE OF studenti.nume%TYPE;
   TYPE tp IS TABLE OF studenti.prenume%TYPE;
   t_ids ti;
   t_nume tn;
   t_prenume tp;
   checker VARCHAR(50);
BEGIN
   SELECT id_student, nume, prenume BULK COLLECT INTO t_ids, t_nume, t_prenume
   FROM studenti;
   FOR i IN t_ids.first..t_ids.last LOOP
      SELECT NVL(mail, 'vid') INTO checker
      FROM studenti
      WHERE id_student = t_ids(i);

      IF checker LIKE 'vid' THEN
         UPDATE studenti
         SET mail = t_prenume(i) || '.' || t_nume(i) || '@s.unibuc.ro'
         WHERE id_student LIKE t_ids(i);
      END IF;
   END LOOP;
END generare_mail_studenti;
/
```

```
/
SELECT * FROM STUDENTI;
DECLARE
BEGIN
    generare_mail_studenti;
END;
/
```

Script Output | Query Result

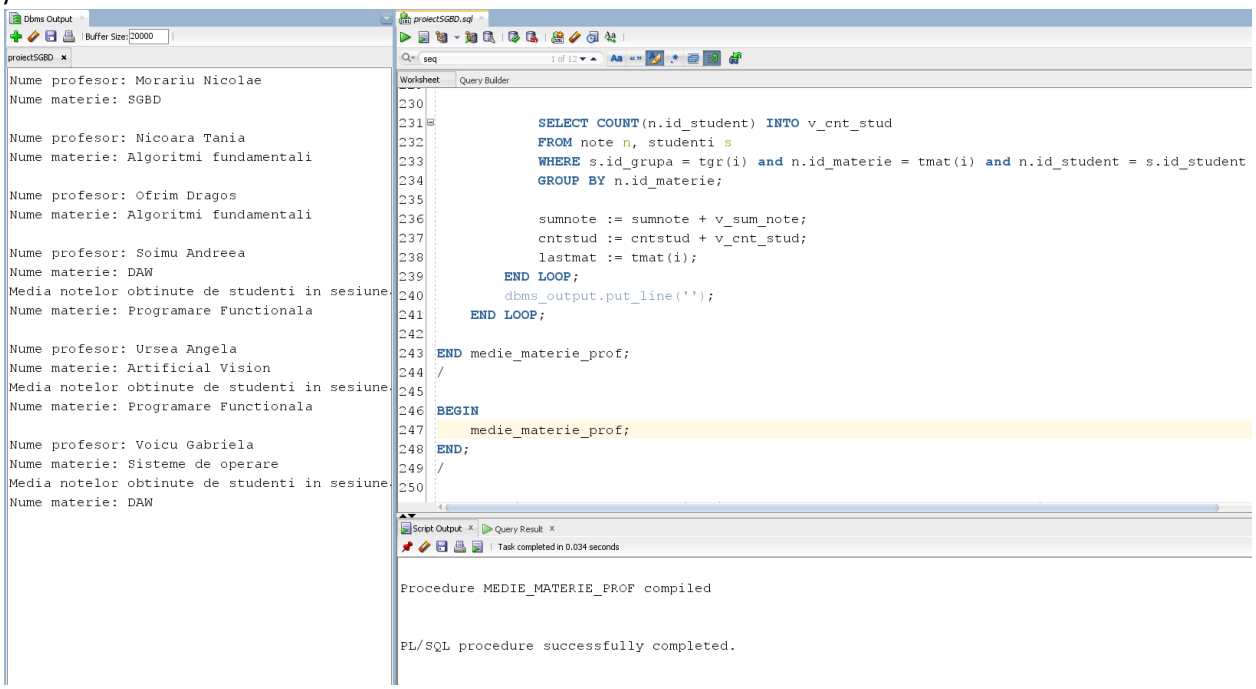
SQL | All Rows Fetched: 8 in 0.004 seconds

	ID_STUDENT	NUME	PRENUME	MAIL	NUMAR_TELEFON	ID_GRUPA
1	1	Aciu	Lia	Lia.Aciu@s.unibuc.ro	0729027022	234
2	2	Baltag	Octavian	Octavian.Baltag@s.unibuc.ro	0795002156	106
3	3	Constantinescu	Mircea	Mircea.Constantinescu@s.unibuc.ro	0782491235	334
4	4	Darie	Eleonora	Eleonora.Darie@s.unibuc.ro	0759825500	441
5	5	Manea	Lucian	Lucian.Manea@s.unibuc.ro	0765395342	236
6	21	Manea	Lucian	Lucian.Manea@s.unibuc.ro	0725395345	334
7	22	Popa	Daniel	Daniel.Popa@s.unibuc.ro	0757695345	334
8	8	Cobzaru	Alina	Alina.Cobzaru@s.unibuc.ro	0765395342	334

7. **procedura care calculeaza media artimetica a rezultatelor studentilor pt fiecare profesor la fiecare materie pe care acesta o preda**

```
CREATE OR REPLACE PROCEDURE medie_materie_prof AS
   CURSOR c IS SELECT id_profesor, nume, prenume FROM profesori;
   TYPE tg IS TABLE OF studenti.id_grupa%TYPE;
   TYPE tm IS TABLE OF materii.id_materie%TYPE;
   tgr tg;
   tmat tm;
   lastmat materii.id_materie%TYPE;
   v_nume_materie materii.nume_materie%TYPE;
   sumnote NUMBER(20) := 0;
   cntstud NUMBER(20) := 0;
   v_sum_note NUMBER(20);
   v_cnt_stud NUMBER(20);
BEGIN
   FOR prof in c LOOP
      dbms_output.put_linE('Nume profesor: ' || prof.nume || ' ' || prof.prenume);
      SELECT DISTINCT id_grupa, id_materie BULK COLLECT INTO tgr, tmat
      FROM asignare_grupe_profesori
      WHERE id_profesor = prof.id_profesor
      ORDER BY id_materie;

      lastmat := tmat(tmat.first);
      SELECT nume_materie INTO v_nume_materie
      FROM materii
      WHERE id_materie = lastmat;
      dbms_output.put_linE('Nume materie: ' || v_nume_materie);
      FOR i in tmat.first..tmat.last LOOP
         IF lastmat <> tmat(i) THEN
            dbms_output.put_line('Media notelor obtinute de studenti in sesiunea curenta: '
|| sumnote/cntstud);

            SELECT nume_materie INTO v_nume_materie
            FROM materii
            WHERE id_materie = tmat(i);
            dbms_output.put_line('Nume materie: ' || v_nume_materie);
            sumnote := 0;
            cntstud := 0;
         END IF;
```

```
        SELECT SUM(nota) INTO v_sum_note
        FROM note n, studenti s
        WHERE s.id_grupa = tgr(i) and n.id_materie = tmat(i) and n.id_student = s.id_student;

        SELECT COUNT(n.id_student) INTO v_cnt_stud
        FROM note n, studenti s
        WHERE s.id_grupa = tgr(i) and n.id_materie = tmat(i) and n.id_student = s.id_student
        GROUP BY n.id_materie;

        sumnote := sumnote + v_sum_note;
        cntstud := cntstud + v_cnt_stud;
        lastmat := tmat(i);
      END LOOP;
      dbms_output.put_line('');
   END LOOP;
END medie_materie_prof;
/
```

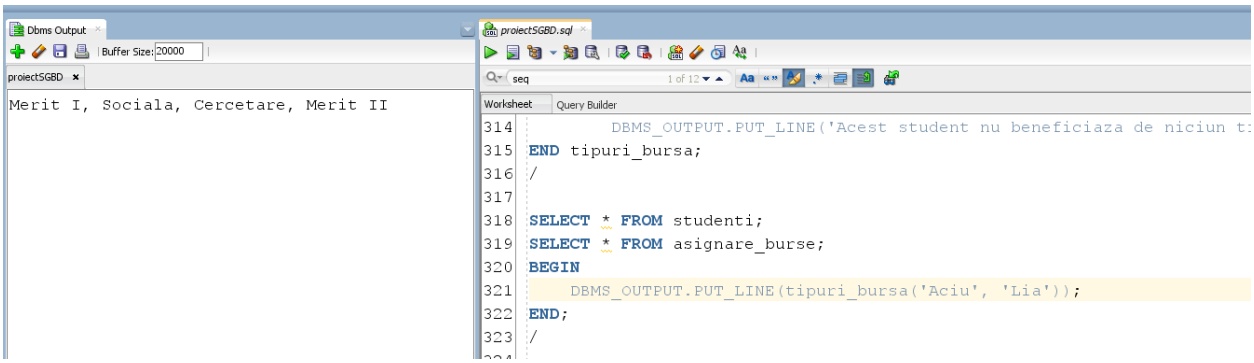


8. **functie care returneaza tipurile de bursa pentru un student al carui nume este dat ca parametru**

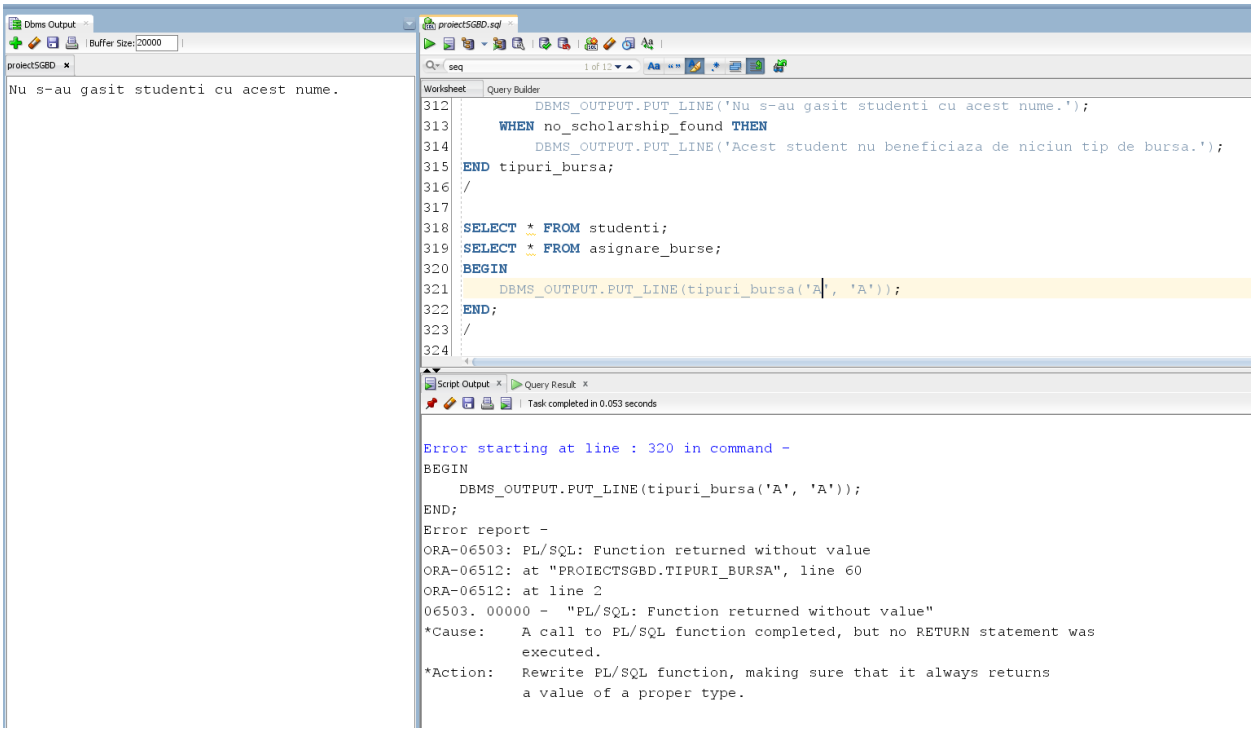
```
CREATE OR REPLACE FUNCTION tipuri_bursa
   (v_nume studenti.nume%TYPE, v_prenume studenti.prenume%TYPE) RETURN VARCHAR
AS
   TYPE t IS TABLE OF burse.id_bursa%TYPE;
   t_id_bursa t;
   too_many_students EXCEPTION;
   no_students_found EXCEPTION;
   no_scholarship_found EXCEPTION;
   v_id_student studenti.id_student%TYPE;
   v_nume_bursa burse.nume_bursa%TYPE;
   err_checker NUMBER(10);
   str_burse VARCHAR(100) := '';
BEGIN
   SELECT COUNT(id_student) INTO err_checker
   FROM studenti
   WHERE nume = v_nume AND prenume = v_prenume;

   IF err_checker > 1 THEN
      RAISE too_many_students;
   END IF;

   IF err_checker = 0 THEN
      RAISE no_students_found;
   END IF;

   SELECT id_student INTO v_id_student
   FROM studenti
   WHERE nume = v_nume AND prenume = v_prenume;

   SELECT COUNT(id_bursa) INTO err_checker
   FROM asignare_burse
   WHERE id_student = v_id_student;

   IF err_checker = 0 THEN
      RAISE no_scholarship_found;
   END IF;

   SELECT id_bursa BULK COLLECT INTO t_id_bursa
   FROM asignare_burse
   WHERE id_student = v_id_student;
```

```
    FOR i in t_id_bursa.first..t_id_bursa.last LOOP
        SELECT nume_bursa INTO v_nume_bursa
        FROM burse
        WHERE id_bursa = t_id_bursa(i);
        IF i != t_id_bursa.first THEN
            str_burse := str_burse || ', ' || v_nume_bursa;
        ELSE
            str_burse := v_nume_bursa;
        END IF;
    END LOOP;
    RETURN str_burse;
EXCEPTION
    WHEN too_many_students THEN
        DBMS_OUTPUT.PUT_LINE('S-au gasit mai multi studenti cu acest nume.');
    WHEN no_students_found THEN
        DBMS_OUTPUT.PUT_LINE('Nu s-au gasit studenti cu acest nume.');
    WHEN no_scholarship_found THEN
        DBMS_OUTPUT.PUT_LINE('Acest student nu beneficiaza de niciun tip de bursa.');
END tipuri_bursa;
/
```

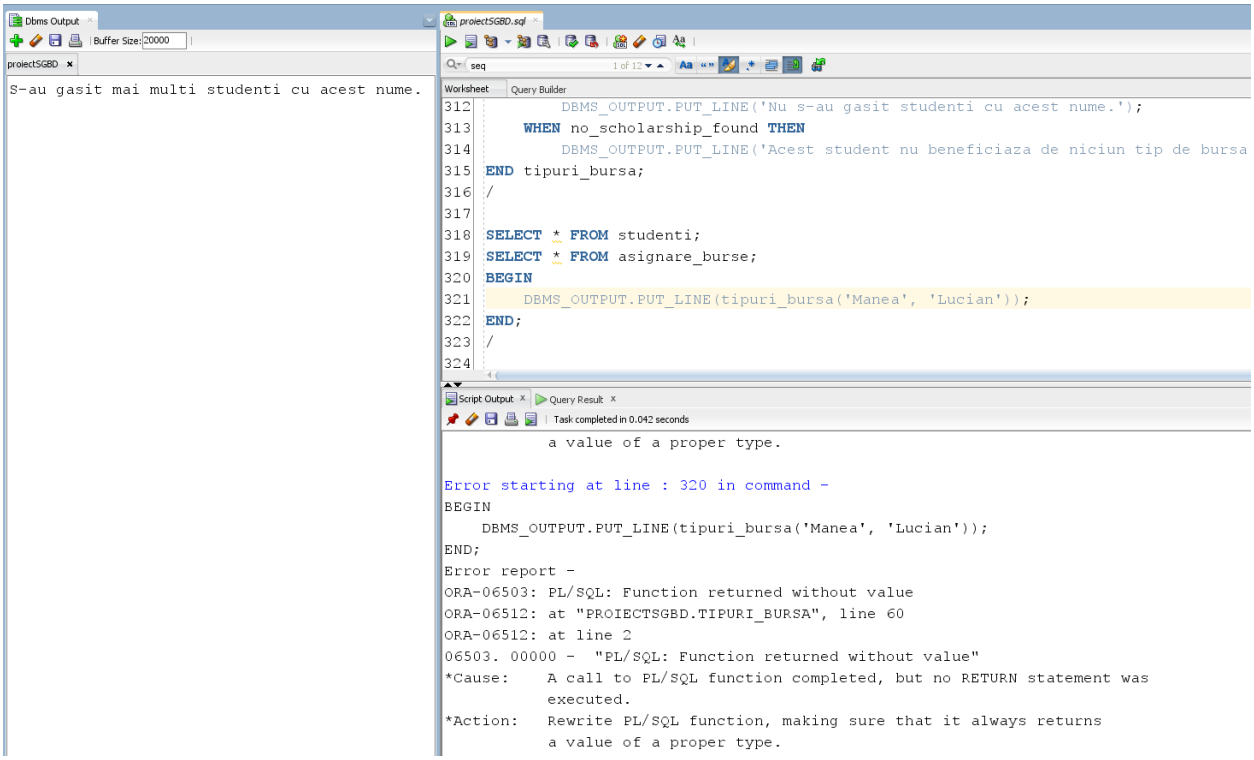
Cazul 1 : studentul este gasit si are bursa



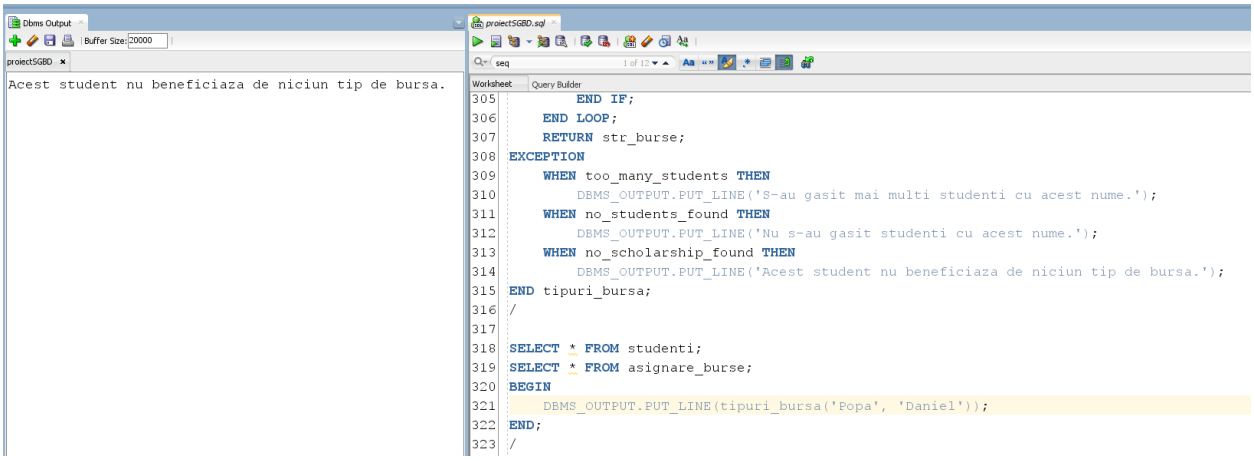
Cazul 2 : studentul nu este gasit



Cazul 3 : exista mai multi studenti cu numele dat



Cazul 4 : studentul dat nu beneficiaza de nicio bursa



9. **procedura prin care se calculeaza bursa(merit I si II) pentru fiecare student in functie de notele obtinute. Criterii: daca media este 10 se acorda merit II, daca media este peste 8 dar != 10 se acorda merit I.**

```
CREATE OR REPLACE PROCEDURE calculare_burse
AS
   suma_note NUMBER(10);
   nr_materii NUMBER(10);
   medie NUMBER(10);
   ok NUMBER(10);
   no_grades EXCEPTION;
   no_classes EXCEPTION;
   grupa_invalida EXCEPTION;
   v_an NUMBER(10);
   v_domeniu grupe.domeniu%TYPE;
   v_id_grupa grupe.id_grupa%TYPE;
   v_localizare NUMBER(10);
BEGIN
   --calculare burse
   FOR i IN (SELECT id_student FROM studenti) LOOP

      v_localizare := 1;
      SELECT SUM(nota) INTO suma_note
      FROM note
      WHERE id_student = i.id_student;

      SELECT id_grupa INTO v_id_grupa
      FROM studenti
```

```
WHERE id_student = i.id_student;

ok := 0;
FOR j IN (SELECT id_grupa FROM grupe) LOOP
   IF j.id_grupa = v_id_grupa THEN
      ok := 1;
   END IF;
END LOOP;

IF ok = 0 THEN
   RAISE grupa_invalida;
END IF;

v_localizare := 2;
SELECT COUNT(id_materie) INTO nr_materii
FROM materii m, studenti s, grupe g
WHERE s.id_student = i.id_student AND s.id_grupa = g.id_grupa
      AND m.an = g.an AND m.domeniu = g.domeniu
GROUP BY m.an;

IF nr_materii = 0 THEN
   RAISE no_classes;
END IF;

medie := suma_note/nr_materii;
-- asignare merit II
IF medie = 10 THEN
   ok := 1;
   FOR j IN (SELECT id_bursa
             FROM asignare_burse
             WHERE id_student = i.id_student) LOOP
      --studentul are deja merit II trecut in tabel
      IF j.id_bursa = 2 THEN
         ok := 0;
      END IF;
   END LOOP;

   IF ok = 1 THEN
      INSERT INTO asignare_burse VALUES (i.id_student, 2);
   END IF;
END IF;

--asignare merit 1
IF medie >= 8 AND medie <> 10 THEN
```

```
        ok := 1;
        FOR j IN (SELECT id_bursa
                  FROM asignare_burse
                  WHERE id_student = i.id_student) LOOP
          --studentul are deja merit II trecut in tabel
          IF j.id_bursa = 1 THEN
            ok := 0;
          END IF;
        END LOOP;

        IF ok = 1 THEN
          INSERT INTO asignare_burse VALUES (i.id_student, 1);
        END IF;
      END IF;
    END LOOP;
EXCEPTION
-- 1. studentul nu are notele trecute
    WHEN no_data_found THEN
      IF v_localizare = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Exista studenti care nu au notele trecute');
      END IF;
-- 2. nu a putut fi calculat totalul materiilor la care a luat parte un student
      IF v_localizare = 2 THEN
        DBMS_OUTPUT.PUT_LINE('Nu a putut fi calculat totalul materiilor');
      END IF;
-- 3. grupa invalida
    WHEN grupa_invalida THEN
      dbms_output.put_line('Nu exista grupa ' || v_id_grupa);
END calculare_burse;
/
```

Cazul 1 : procedura functioneaza si calculeaza bursele

inainte de calculare:

Script Output | Query Result

SQL | All Rows Fetched: 14 in 0

	ID_STUDENT	ID_BURSA
1	1	1
2	1	3
3	4	4
4	4	2
5	2	3
6	8	3
7	8	2
8	3	1
9	5	3
10	1	4
11	1	2
12	2	2
13	4	1
14	5	1

dupa calculare:

```
BEGIN
    calculare_burse;
END;
/
ROLLBACK;
SELECT * FROM ASIGNARE_BURSE;
SELECT * FROM NOTE;
SELECT * FROM MATERII;
SELECT * FROM STUDENTI;
```

Script Output | Query Result

SQL | All Rows Fetched: 16 in 0.001 seconds

	ID_STUDENT	ID_BURSA
1	1	1
2	1	3
3	4	4
4	4	2
5	2	3
6	8	3
7	8	2
8	3	1
9	5	3
10	1	4
11	1	2
12	2	2
13	4	1
14	5	1
15	21	1
16	22	2

Cazul 2: a fost gasit un student cu grupa invalida

```
Nu exista grupa 555
```

```
BEGIN
    calculare_burse;
END;
/
ROLLBACK;
SELECT * FROM ASIGNARE_BURSE;
SELECT * FROM NOTE;
SELECT * FROM MATERII;
SELECT * FROM STUDENTI;

--verificare eroare pt grupa inexistenta
INSERT INTO studenti(id_student, nume, prenume, id_grupa) VALUES (9, 'Badescu', 'Georgiana', 555);
```

Cazul 3 : nu au fost introduse materii pentru un anumit an de studiu/domeniu licenta

```
Nu a putut fi calculat totalul materiilor
```

```
SELECT * FROM materii;
SELECT * FROM grupe;

BEGIN
    calculare_burse;
END;
/
ROLLBACK;
SELECT * FROM ASIGNARE_BURSE;

--verificare eroare pt grupa inexistenta
INSERT INTO studenti(id_student, nume, prenume, id_grupa) VALUES (9, 'Badescu', 'Georgiana', 555);

--verificare eroare no_classes
INSERT INTO grupe (id_grupa, an, domeniu, nivel) VALUES (555, 2, 'Baze de date', 'Master');
```

10. trigger care declanseaza generarea automata a adresei institutionale pentru studentii inserati

```
CREATE OR REPLACE TRIGGER trig_mail_gen_stud
AFTER INSERT ON studenti
BEGIN
   generare_mail_studenti;
END;
/
```

```
--10 trigger care declanseaza generarea automata a adresei institutionale pentru studentii inserati
CREATE OR REPLACE TRIGGER trig_mail_gen_stud
AFTER INSERT ON studenti
BEGIN
    generare_mail_studenti;
END;
/
INSERT INTO studenti(id_student, nume, prenume, id_grupa) VALUES (9, 'Badescu', 'Georgiana', 555);
SELECT * FROM studenti;
DROP TRIGGER trig_mail_gen_stud;

--11
```

Script Output | Query Result

All Rows Fetched: 9 in 0.004 seconds

	ID_STUDENT	NUME	PRENUME	MAIL	NUMAR_TELEFON	ID_GRUPA
1	1	Aciu	Lia	Lia.Aciu@s.unibuc.ro	0729027022	243
2	2	Baltag	Octavian	Octavian.Baltag@s.unibuc.ro	0795002156	106
3	3	Constantinescu	Mircea	Mircea.Constantinescu@s.un...	0782491235	334
4	4	Darie	Eleonora	Eleonora.Darie@s.unibuc.ro	0759825500	441
5	5	Manea	Lucian	Lucian.Manea@s.unibuc.ro	0765395342	236
6	21	Manea	Lucian	Lucian.Manea@s.unibuc.ro	0725395345	334
7	22	Popa	Daniel	Daniel.Popa@s.unibuc.ro	0757695345	334
8	8	Cobzaru	Alina	Alina.Cobzaru@s.unibuc.ro	0765395342	334
9	9	Badescu	Georg...	Georgiana.Badescu@s.unibuc.ro	(null)	555

11.trigger care genereaza mail-ul institutional al unui profesor inserat

```
CREATE OR REPLACE TRIGGER trig_mail_gen_prof
BEFORE INSERT ON profesori
FOR EACH ROW
BEGIN
        :NEW.mail := :NEW.prenume || '.' || :NEW.nume || '@unibuc.ro';
END;
/
```

```
--11
CREATE OR REPLACE TRIGGER trig_mail_gen_prof
BEFORE INSERT ON profesori
FOR EACH ROW
BEGIN
    :NEW.mail := :NEW.prenume || '.' || :NEW.nume || '@unibuc.ro';
END;
/

INSERT INTO profesori(id_profesor, nume, prenume) VALUES (9, 'Badescu', 'Georgiana');
SELECT * FROM profesori;
DROP TRIGGER trig_mail_gen_prof;

--12
CREATE TABLE modif_log(utilizator VARCHAR2(30),
```

Script Output | Query Result

All Rows Fetched: 7 in 0.003 seconds

	ID_PROFESOR	NUME	PRENUME	MAIL	NUMAR_TELEFON	NIVEL	TITLU
1	1	Morariu	Nicolae	(null)	0710049511	dr.	lector
2	2	Nicoara	Tania	(null)	0702658959	drd.	asistent
3	3	Ofrim	Dragos	(null)	0787013325	dr.	profesor
4	4	Soimu	Andreea	(null)	0726681537	dr.	conferentiar
5	5	Ursea	Angela	(null)	0716023455	drd.	asistent
6	6	Voicu	Gabriela	(null)	0716023455	dr.	conferentiar
7	9	Badescu	Georg...	Geo...	(null)	(...	(null)

12. trigger care retine intr-un tabel modificarile realizate la nivelul bazei de date

```
CREATE TABLE modif_log(utilizator VARCHAR2(30),
            baza_de_date VARCHAR2(50),
            eveniment VARCHAR2(20),
            nume_obiect VARCHAR2(30),
            data DATE);
CREATE OR REPLACE TRIGGER trig_log
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
  INSERT INTO modif_log
  VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
      SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;
/
```

```
--12 trigger care retine intr-un tabel modificarile realizate la nivelul bazei de date
CREATE TABLE modif_log(utilizator VARCHAR2(30),
                       baza_de_date VARCHAR2(50),
                       eveniment VARCHAR2(20),
                       nume_obiect VARCHAR2(30),
                       data DATE);
CREATE OR REPLACE TRIGGER trig_log
AFTER CREATE OR DROP OR ALTER ON SCHEMA
BEGIN
    INSERT INTO modif_log
    VALUES (SYS.LOGIN_USER, SYS.DATABASE_NAME, SYS.SYSEVENT,
            SYS.DICTIONARY_OBJ_NAME, SYSDATE);
END;
/
CREATE TABLE test (id int);
SELECT * FROM modif_log;
DROP TRIGGER trig_log;
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.004 seconds

	UTILIZATOR	BAZA_DE_DATE	EVENIMENT	NUME_OBIECT	DATA
1	PROIECTSGBD	XE	CREATE	TEST	09-JAN-21

13.pachetul ce contine toate procedurile create

```
CREATE OR REPLACE PACKAGE pachetall AS
   PROCEDURE generare_mail_studenti;
   PROCEDURE medie_materie_prof;
   FUNCTION tipuri_bursa
      (v_nume studenti.nume%TYPE, v_prenume studenti.prenume%TYPE)
   RETURN VARCHAR;
   PROCEDURE calculare_burse;
END pachetall;
/
CREATE OR REPLACE PACKAGE BODY pachetall AS
   --6
   PROCEDURE generare_mail_studenti
   AS
      TYPE ti IS TABLE OF studenti.id_student%TYPE;
      TYPE tn IS TABLE OF studenti.nume%TYPE;
      TYPE tp IS TABLE OF studenti.prenume%TYPE;
      t_ids ti;
      t_nume tn;
      t_prenume tp;
      checker VARCHAR(50);
```

```
BEGIN
   SELECT id_student, nume, prenume BULK COLLECT INTO t_ids, t_nume, t_prenume
   FROM studenti;
   FOR i IN t_ids.first..t_ids.last LOOP
      SELECT NVL(mail, 'vid') INTO checker
      FROM studenti
      WHERE id_student = t_ids(i);

      IF checker LIKE 'vid' THEN
         UPDATE studenti
         SET mail = t_prenume(i) || '.' || t_nume(i) || '@s.unibuc.ro'
         WHERE id_student LIKE t_ids(i);
      END IF;
   END LOOP;
END generare_mail_studenti;


--7
PROCEDURE medie_materie_prof AS
   CURSOR c IS SELECT id_profesor, nume, prenume FROM profesori;
   TYPE tg IS TABLE OF studenti.id_grupa%TYPE;
   TYPE tm IS TABLE OF materii.id_materie%TYPE;
   tgr tg;
   tmat tm;
   lastmat materii.id_materie%TYPE;
   v_nume_materie materii.nume_materie%TYPE;
   sumnote NUMBER(20) := 0;
   cntstud NUMBER(20) := 0;
   v_sum_note NUMBER(20);
   v_cnt_stud NUMBER(20);
BEGIN
   FOR prof in c LOOP
      dbms_output.put_linE('Nume profesor: ' || prof.nume || ' ' || prof.prenume);
      SELECT DISTINCT id_grupa, id_materie BULK COLLECT INTO tgr, tmat
      FROM asignare_grupe_profesori
      WHERE id_profesor = prof.id_profesor
      ORDER BY id_materie;

      lastmat := tmat(tmat.first);
      SELECT nume_materie INTO v_nume_materie
      FROM materii
      WHERE id_materie = lastmat;
      dbms_output.put_linE('Nume materie: ' || v_nume_materie);
      FOR i in tmat.first..tmat.last LOOP
```

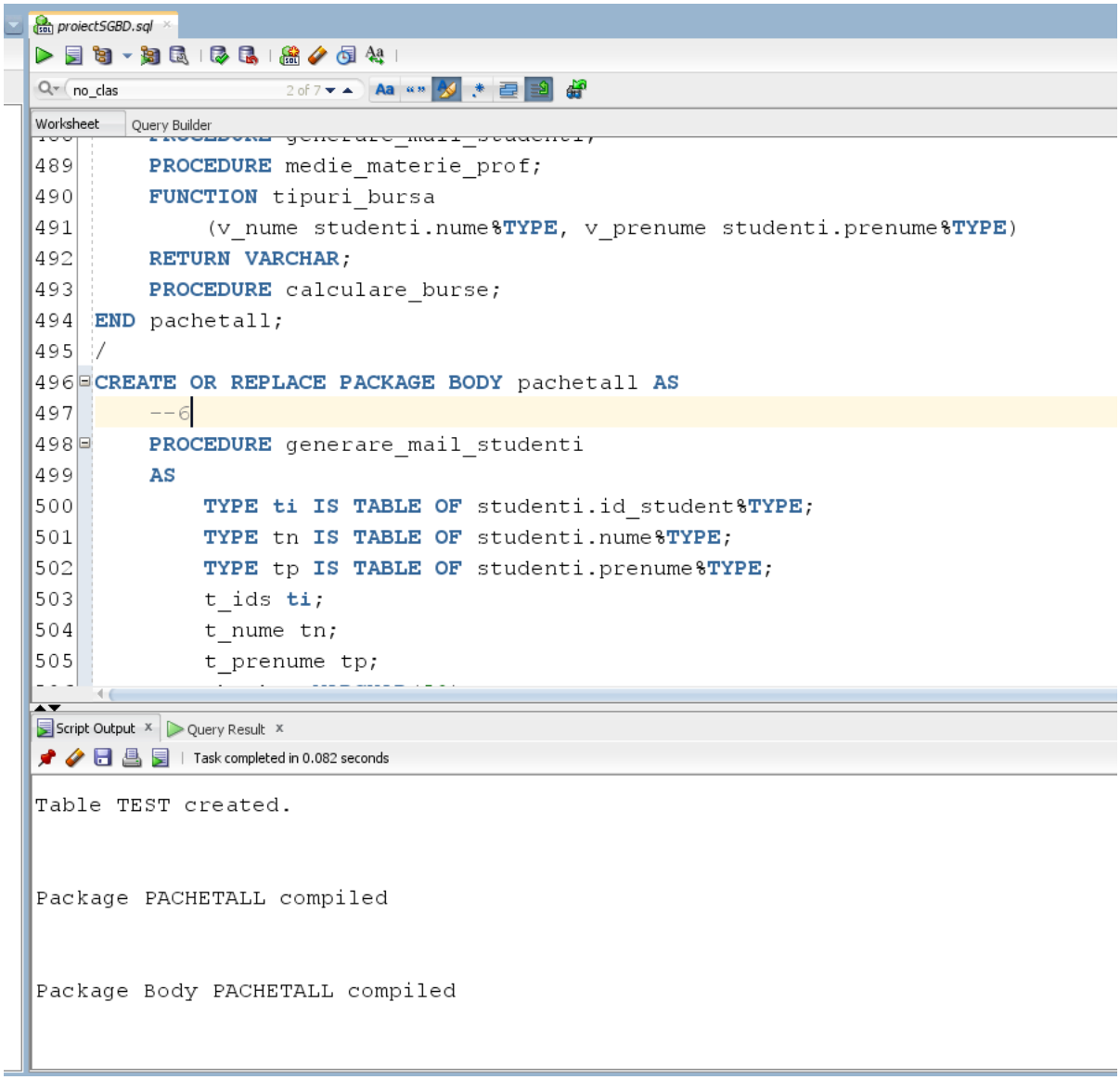
```
            IF lastmat <> tmat(i) THEN
               dbms_output.put_line('Media notelor obtinute de studenti in sesiunea curenta: '
|| sumnote/cntstud);

               SELECT nume_materie INTO v_nume_materie
               FROM materii
               WHERE id_materie = tmat(i);
               dbms_output.put_line('Nume materie: ' || v_nume_materie);
               sumnote := 0;
               cntstud := 0;
            END IF;



            SELECT SUM(nota) INTO v_sum_note
            FROM note n, studenti s
            WHERE s.id_grupa = tgr(i) and n.id_materie = tmat(i) and n.id_student =
s.id_student;

            SELECT COUNT(n.id_student) INTO v_cnt_stud
            FROM note n, studenti s
            WHERE s.id_grupa = tgr(i) and n.id_materie = tmat(i) and n.id_student =
s.id_student
            GROUP BY n.id_materie;

            sumnote := sumnote + v_sum_note;
            cntstud := cntstud + v_cnt_stud;
            lastmat := tmat(i);
         END LOOP;
         dbms_output.put_line('');
      END LOOP;
   END medie_materie_prof;

   --8
   FUNCTION tipuri_bursa
      (v_nume studenti.nume%TYPE, v_prenume studenti.prenume%TYPE)
   RETURN VARCHAR
   AS
      TYPE t IS TABLE OF burse.id_bursa%TYPE;
      t_id_bursa t;
      too_many_students EXCEPTION;
      no_students_found EXCEPTION;
      no_scholarship_found EXCEPTION;
      v_id_student studenti.id_student%TYPE;
```

```
    v_nume_bursa burse.nume_bursa%TYPE;
    err_checker NUMBER(10);
    str_burse VARCHAR(100) := '';
BEGIN
    SELECT COUNT(id_student) INTO err_checker
    FROM studenti
    WHERE nume = v_nume AND prenume = v_prenume;

    IF err_checker > 1 THEN
        RAISE too_many_students;
    END IF;

    IF err_checker = 0 THEN
        RAISE no_students_found;
    END IF;

    SELECT id_student INTO v_id_student
    FROM studenti
    WHERE nume = v_nume AND prenume = v_prenume;

    SELECT COUNT(id_bursa) INTO err_checker
    FROM asignare_burse
    WHERE id_student = v_id_student;

    IF err_checker = 0 THEN
        RAISE no_scholarship_found;
    END IF;

    SELECT id_bursa BULK COLLECT INTO t_id_bursa
    FROM asignare_burse
    WHERE id_student = v_id_student;

    FOR i in t_id_bursa.first..t_id_bursa.last LOOP
        SELECT nume_bursa INTO v_nume_bursa
        FROM burse
        WHERE id_bursa = t_id_bursa(i);
        IF i != t_id_bursa.first THEN
            str_burse := str_burse || ', ' || v_nume_bursa;
        ELSE
            str_burse := v_nume_bursa;
        END IF;
    END LOOP;
    RETURN str_burse;
EXCEPTION
```

```
    WHEN too_many_students THEN
        DBMS_OUTPUT.PUT_LINE('S-au gasit mai multi studenti cu acest nume.');
    WHEN no_students_found THEN
        DBMS_OUTPUT.PUT_LINE('Nu s-au gasit studenti cu acest nume.');
    WHEN no_scholarship_found THEN
        DBMS_OUTPUT.PUT_LINE('Acest student nu beneficiaza de niciun tip de bursa.');
END tipuri_bursa;

--9
PROCEDURE calculare_burse
AS
    suma_note NUMBER(10);
    nr_materii NUMBER(10);
    medie NUMBER(10);
    ok NUMBER(10);
    no_grades EXCEPTION;
    no_classes EXCEPTION;
    grupa_invalida EXCEPTION;
    v_an NUMBER(10);
    v_domeniu grupe.domeniu%TYPE;
    v_id_grupa grupe.id_grupa%TYPE;
    v_localizare NUMBER(10);
BEGIN
    --calculare burse
    FOR i IN (SELECT id_student FROM studenti) LOOP

        v_localizare := 1;
        SELECT SUM(nota) INTO suma_note
        FROM note
        WHERE id_student = i.id_student;

        SELECT id_grupa INTO v_id_grupa
        FROM studenti
        WHERE id_student = i.id_student;

        ok := 0;
        FOR j IN (SELECT id_grupa FROM grupe) LOOP
            IF j.id_grupa = v_id_grupa THEN
                ok := 1;
            END IF;
        END LOOP;

        IF ok = 0 THEN
            RAISE grupa_invalida;
```

```
END IF;

v_localizare := 2;
SELECT COUNT(id_materie) INTO nr_materii
FROM materii m, studenti s, grupe g
WHERE s.id_student = i.id_student AND s.id_grupa = g.id_grupa
      AND m.an = g.an AND m.domeniu = g.domeniu
GROUP BY m.an;

IF nr_materii = 0 THEN
   RAISE no_classes;
END IF;

medie := suma_note/nr_materii;
-- asignare merit II
IF medie = 10 THEN
   ok := 1;
   FOR j IN (SELECT id_bursa
             FROM asignare_burse
             WHERE id_student = i.id_student) LOOP
      --studentul are deja merit II trecut in tabel
      IF j.id_bursa = 2 THEN
         ok := 0;
      END IF;
   END LOOP;

   IF ok = 1 THEN
      INSERT INTO asignare_burse VALUES (i.id_student, 2);
   END IF;
END IF;

--asignare merit 1
IF medie >= 8 AND medie <> 10 THEN
   ok := 1;
   FOR j IN (SELECT id_bursa
             FROM asignare_burse
             WHERE id_student = i.id_student) LOOP
      --studentul are deja merit II trecut in tabel
      IF j.id_bursa = 1 THEN
         ok := 0;
      END IF;
   END LOOP;

   IF ok = 1 THEN
```

```
                INSERT INTO asignare_burse VALUES (i.id_student, 1);
            END IF;
        END IF;
    END LOOP;
  EXCEPTION
  -- 1. studentul nu are notele trecute
    WHEN no_data_found THEN
        IF v_localizare = 1 THEN
            DBMS_OUTPUT.PUT_LINE('Exista studenti care nu au notele trecute');
        END IF;

        IF v_localizare = 2 THEN
            DBMS_OUTPUT.PUT_LINE('Nu a putut fi calculat totalul materiilor');
        END IF;
  -- 2. nu sunt materii trecute pt un anumit an
    WHEN no_classes THEN
        DBMS_OUTPUT.PUT_LINE('Exista domenii/ani de studiu pentru care nu au fost
introduse materii.');
  -- 3. grupa invalida
    WHEN grupa_invalida THEN
        dbms_output.put_line('Nu exista grupa ' || v_id_grupa);
  END calculare_burse;
END pachetall;
/
```



14. pachet care afla pentru un student dat daca exista, ce note are, ce medie are, daca este eligibil pentru vreun tip de bursa

```
CREATE OR REPLACE PACKAGE pachet_studenti AS
   TYPE r_note IS RECORD (nume_materie materii.nume_materie%TYPE,
                          nota note.nota%TYPE);
   TYPE t_note IS TABLE OF r_note;

   FUNCTION exista_student(v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE) RETURN BOOLEAN;
```

```
   PROCEDURE afisare_note_student(v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE);
   FUNCTION obtinere_medie_student(v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE) RETURN NUMBER;
   PROCEDURE afisare_eligibilitate_bursa (v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE);

END pachet_studenti;
/

CREATE OR REPLACE PACKAGE BODY pachet_studenti AS
   FUNCTION exista_student(v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE)
   RETURN BOOLEAN
   IS
      checker NUMBER(10);
   BEGIN
      SELECT COUNT(id_student) INTO checker
      FROM studenti
      WHERE nume = v_nume and prenume = v_prenume
      GROUP BY nume;

      IF checker = 0 THEN
         RETURN FALSE;
      ELSE
         RETURN TRUE;
      END IF;
   END exista_student;

   PROCEDURE afisare_note_student(v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE) AS
      v_id_student studenti.id_student%TYPE;
      tn t_note;
   BEGIN
      IF exista_student(v_nume, v_prenume) THEN
         SELECT id_student INTO v_id_student
         FROM studenti
         WHERE nume = v_nume and prenume = v_prenume;

         SELECT nume_materie, nota BULK COLLECT INTO tn
         FROM note n, materii m
         WHERE id_student = v_id_student and n.id_materie = m.id_materie;

         FOR i IN tn.first..tn.last LOOP
```

```
            DBMS_OUTPUT.PUT_LINE(tn(i).nume_materie || ': ' || tn(i).nota);
         END LOOP;
      ELSE
         DBMS_OUTPUT.PUT_LINE('Studentul nu exista');
      END IF;
   END afisare_note_student;

   FUNCTION obtinere_medie_student(v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE)
   RETURN NUMBER
   IS
      v_id_student studenti.id_student%TYPE;
      sum_note NUMBER(10);
      cnt_mat NUMBER(10);
   BEGIN
      IF exista_student(v_nume, v_prenume) THEN
         SELECT id_student INTO v_id_student
         FROM studenti
         WHERE nume = v_nume and prenume = v_prenume;

         SELECT SUM(nota) INTO sum_note
         FROM note n, materii m
         WHERE id_student = v_id_student and n.id_materie = m.id_materie;

         SELECT COUNT(id_materie) INTO cnt_mat
         FROM note
         WHERE id_student = v_id_student
         GROUP BY id_student;

         RETURN sum_note/cnt_mat;
      ELSE
         DBMS_OUTPUT.PUT_LINE('Studentul nu exista');
         RETURN 0;
      END IF;
   END obtinere_medie_student;

   PROCEDURE afisare_eligibilitate_bursa (v_nume studenti.nume%TYPE, v_prenume
studenti.prenume%TYPE)
   AS
   BEGIN
      IF obtinere_medie_student(v_nume, v_prenume) = 0 THEN
         DBMS_OUTPUT.PUT_LINE('Studentul nu a fost gasit');
      ELSE
         IF obtinere_medie_student(v_nume, v_prenume) = 10 THEN
```

```
            DBMS_OUTPUT.PUT_LINE('Studentul este eligibil pentru Merit II');
        ELSE
            IF obtinere_medie_student(v_nume, v_prenume) >= 8 THEN
                DBMS_OUTPUT.PUT_LINE('Studentul este eligibil pentru Merit I');
            ELSE
                DBMS_OUTPUT.PUT_LINE('Studentul nu este eligibil pentru niciun tip de bursa');
            END IF;
        END IF;
    END IF;
  END afisare_eligibilitate_bursa;
END pachet_studenti;
```