Proiect SGBD

Mihai-Dragoș Preda

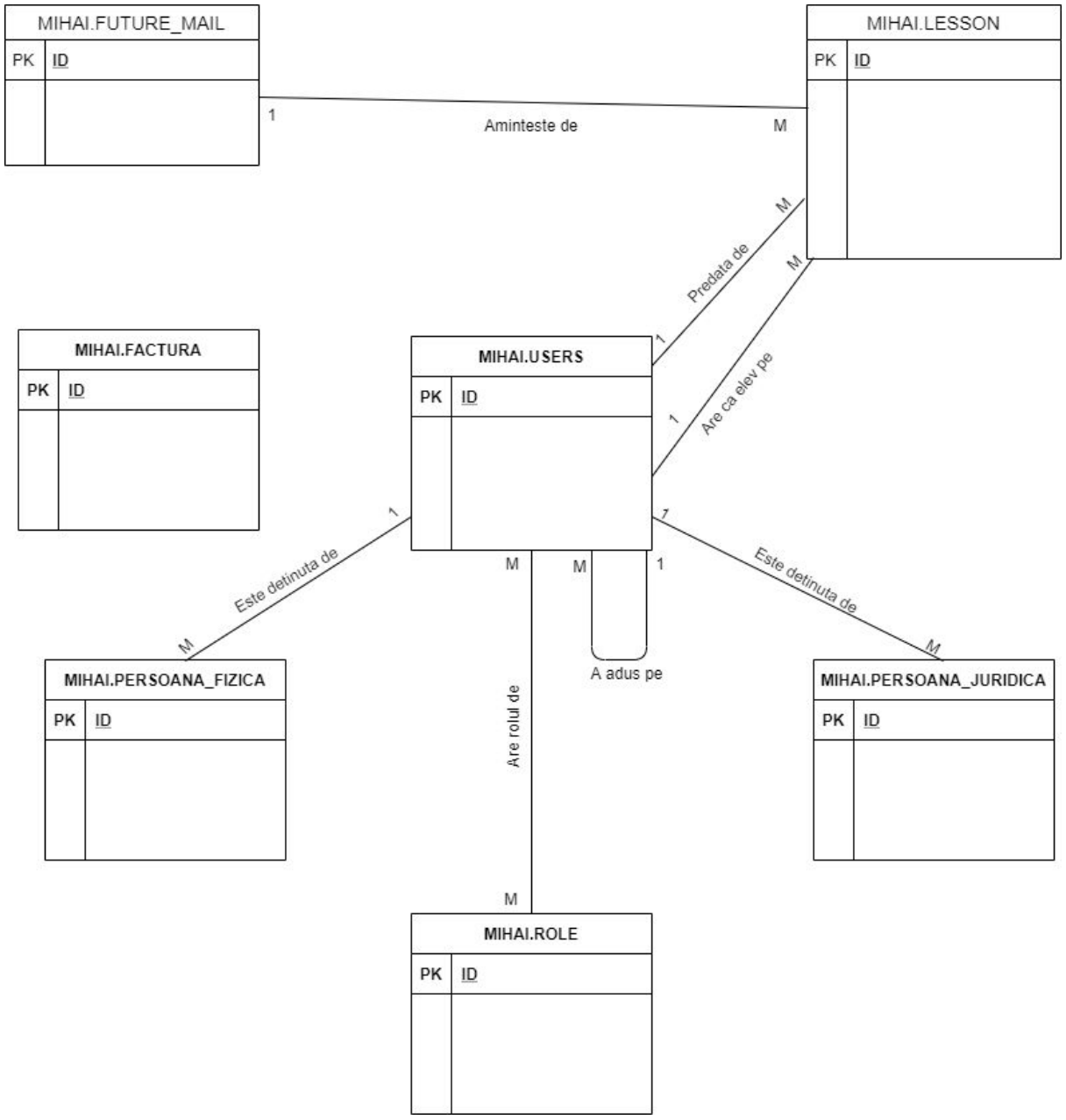
Grupa 234

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

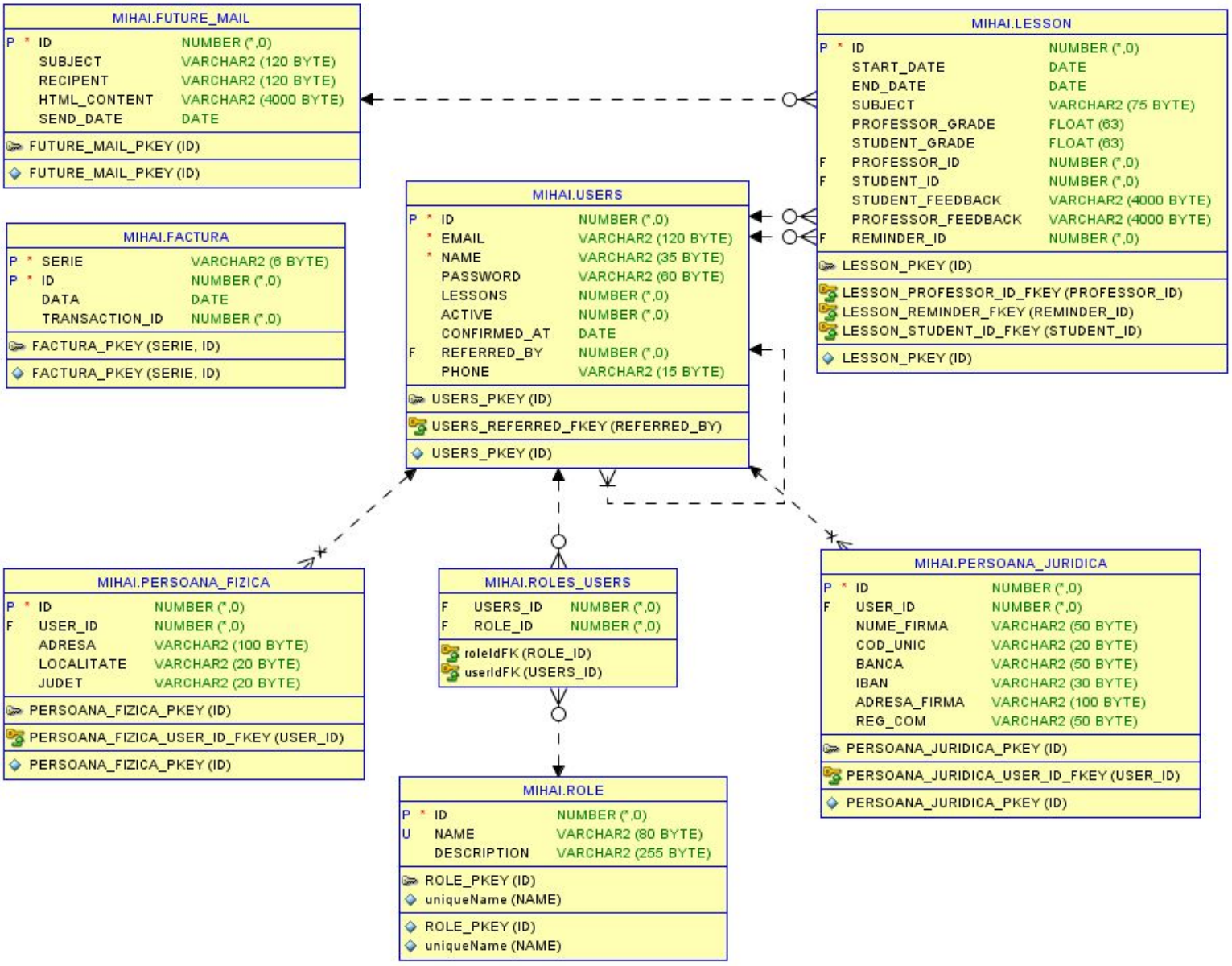
Am realizat o baza de date pentru site-ul web al unei firme ce ofera pregatire personalizată la informatică pentru elevi. Pentru a avea parte de o lecție, un utilizator trebuie sa isi faca un cont, apoi sa cumpere un pachet de lecții (care îi vor fi adaugate în cont), urmand ca apoi sa le rezerve cand dorește el, alegandu-si data și ora dintr-un calendar. Fiecare user inregistrat are atat date personale și informații specifice contului (dacă a fost confirmat contul pe email, câte lecții pe care le-a plătit își mai poate rezerva etc), cat și un rol în site (Admin, Teacher sau Student). Tot în baza de date ținem și un istoric al tranzacțiilor care se fac (fiecare tranzactie fiind însoțită și de o factura), lecțiile desfășurate (cu detalii despre fiecare) și datele de facturare ale utilizatorilor (ca sa nu mai fie nevoie sa le introduca de fiecare data cand fac o plata). De asemenea, site-ul trimite și anumite “remindere” prin mail elevilor, de exemplu înainte cu o oră de o lecție și ne trebuie un tabel pentru a stoca mailurile care trebuie trimise și la ce ora.

Site-ul despre care este vorba este start-up-ul fondat de mine - https://itoit.ro/ - la care eu m-am ocupat de tot backend-ul, inclusiv baza de date.

2. Realizați diagrama entitate-relație (ERD).

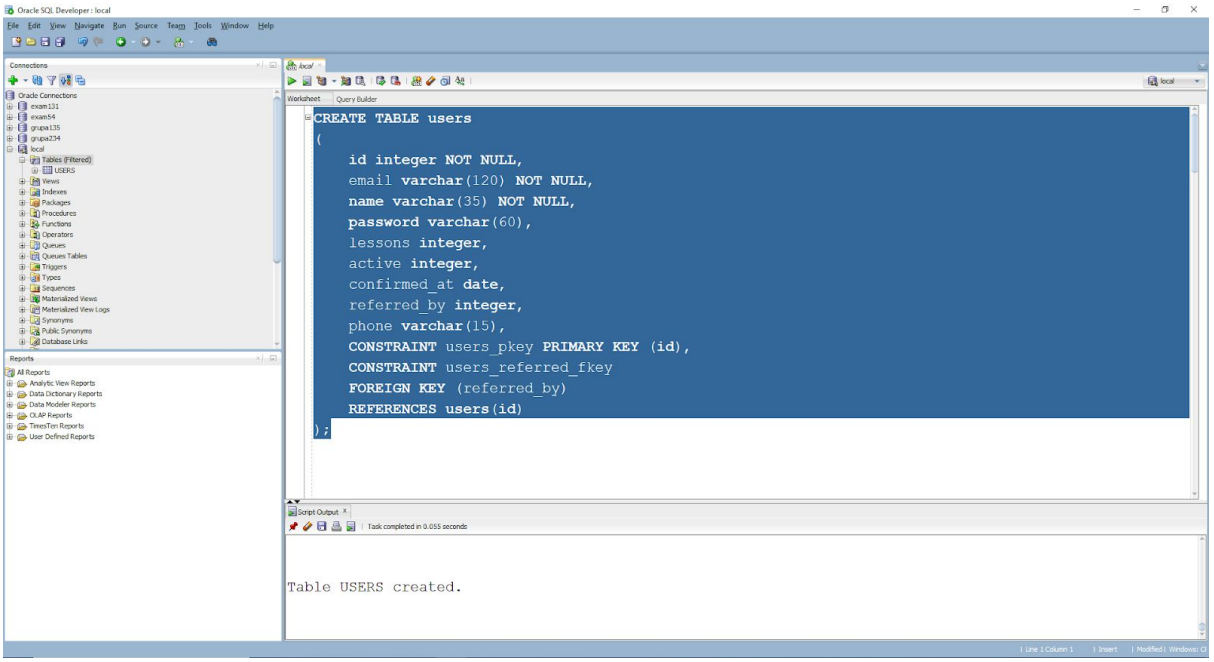


3. Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare.

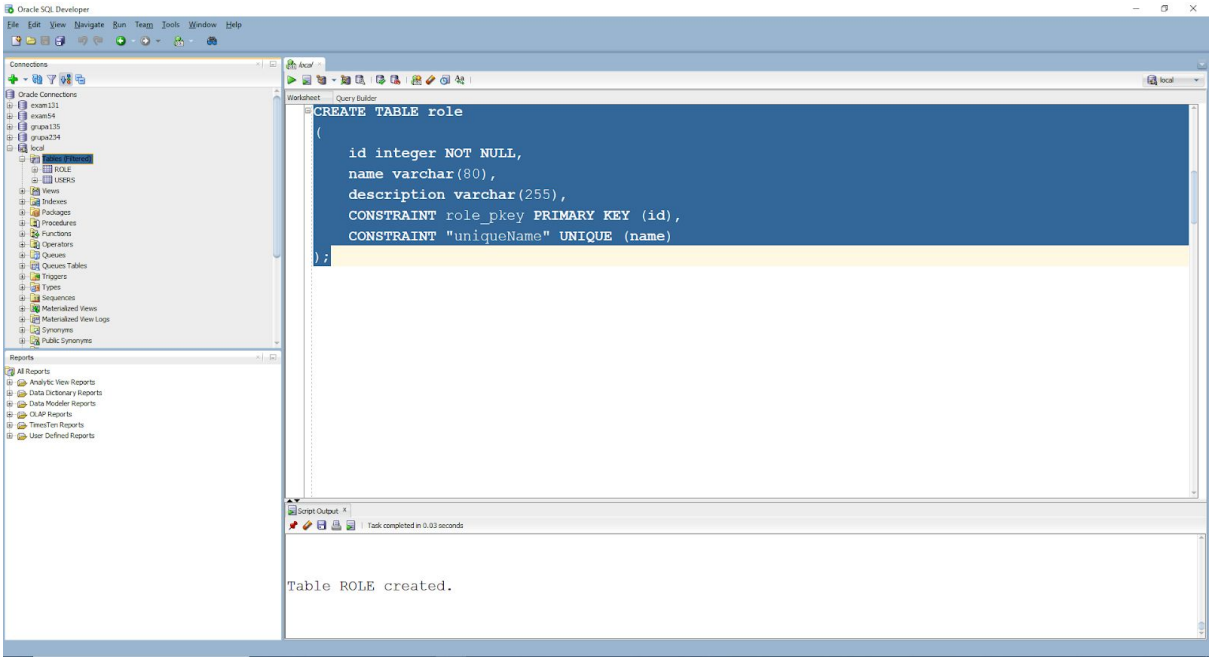


4. Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, implementând toate constrângerile de integritate necesare (chei primare, cheile externe etc).

```
CREATE TABLE users
(
    id integer NOT NULL,
    email varchar(120) NOT NULL,
    name varchar(35) NOT NULL,
    password varchar(60),
    lessons integer,
    active integer,
    confirmed_at date,
    referred_by integer,
    phone varchar(15),
    CONSTRAINT users_pkey PRIMARY KEY (id),
    CONSTRAINT users_referred_fkey
    FOREIGN KEY (referred_by)
    REFERENCES users(id)
);
```

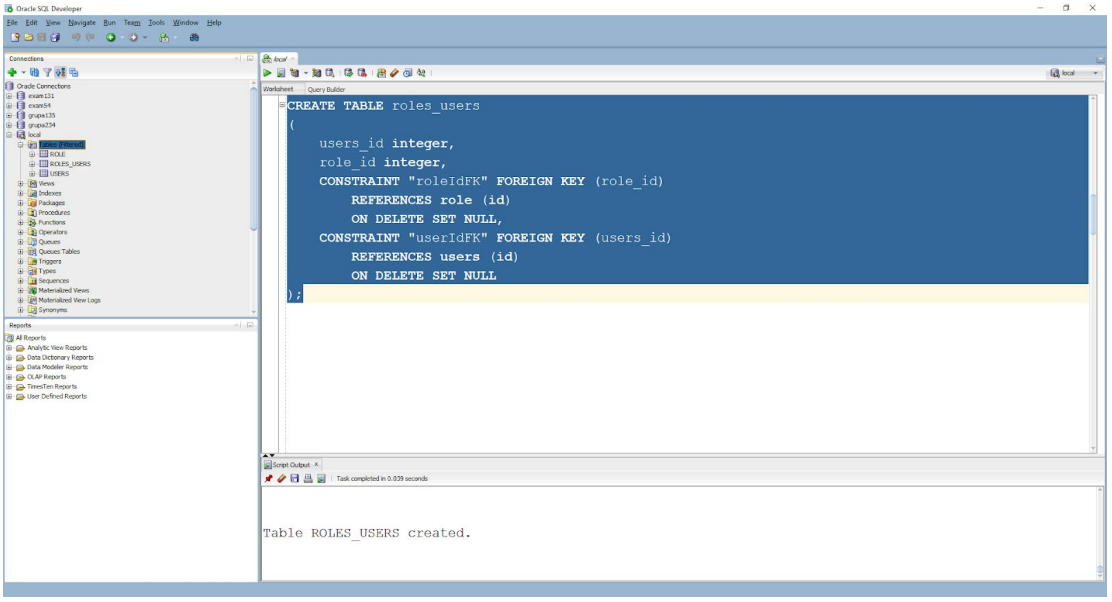



```
CREATE TABLE role
(
    id integer NOT NULL,
    name varchar(80),
    description varchar(255),
    CONSTRAINT role_pkey PRIMARY KEY (id),
    CONSTRAINT "uniqueName" UNIQUE (name)
);
```

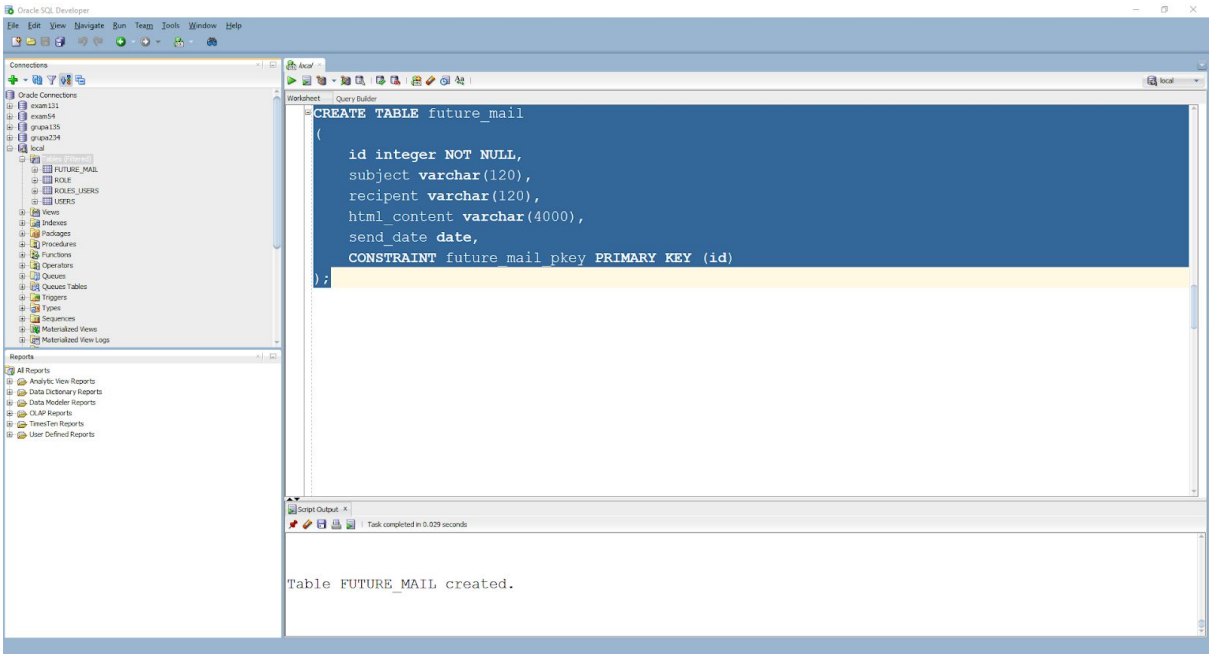



```
CREATE TABLE roles_users
(
```

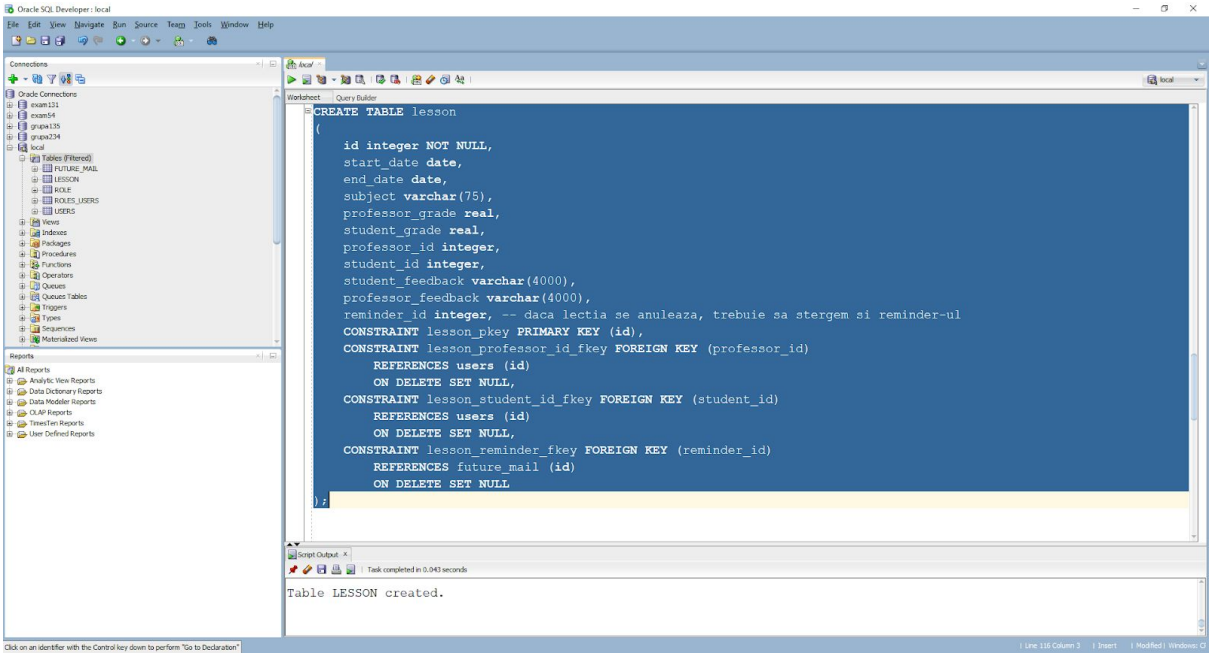
```
    users_id integer,
    role_id integer,
    CONSTRAINT "roleIdFK" FOREIGN KEY (role_id)
        REFERENCES role (id)
        ON DELETE SET NULL,
    CONSTRAINT "userIdFK" FOREIGN KEY (users_id)
        REFERENCES users (id)
        ON DELETE SET NULL
);
```



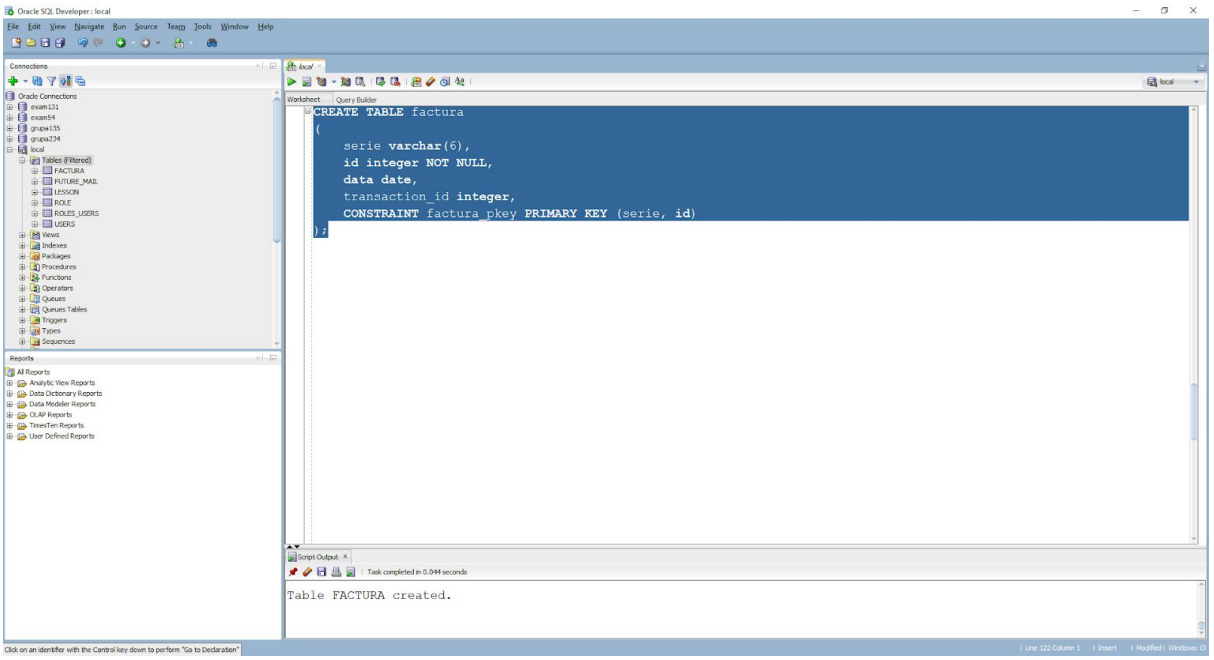

```
CREATE TABLE future_mail
(
    id integer NOT NULL,
    subject varchar(120),
    recipent varchar(120),
    html_content varchar(4000),
    send_date date,
    CONSTRAINT future_mail_pkey PRIMARY KEY (id)
);
```




```
CREATE TABLE lesson
(
    id integer NOT NULL,
    start_date date,
    end_date date,
    subject varchar(75),
    professor_grade real,
    student_grade real,
    professor_id integer,
    student_id integer,
    student_feedback varchar(4000),
    professor_feedback varchar(4000),
    reminder_id integer, -- daca lectia se anuleaza, trebuie sa stergem si
reminder-ul
    CONSTRAINT lesson_pkey PRIMARY KEY (id),
    CONSTRAINT lesson_professor_id_fkey FOREIGN KEY (professor_id)
        REFERENCES users (id)
        ON DELETE SET NULL,
    CONSTRAINT lesson_student_id_fkey FOREIGN KEY (student_id)
        REFERENCES users (id)
        ON DELETE SET NULL,
    CONSTRAINT lesson_reminder_fkey FOREIGN KEY (reminder_id)
        REFERENCES future_mail (id)
        ON DELETE SET NULL
);
```

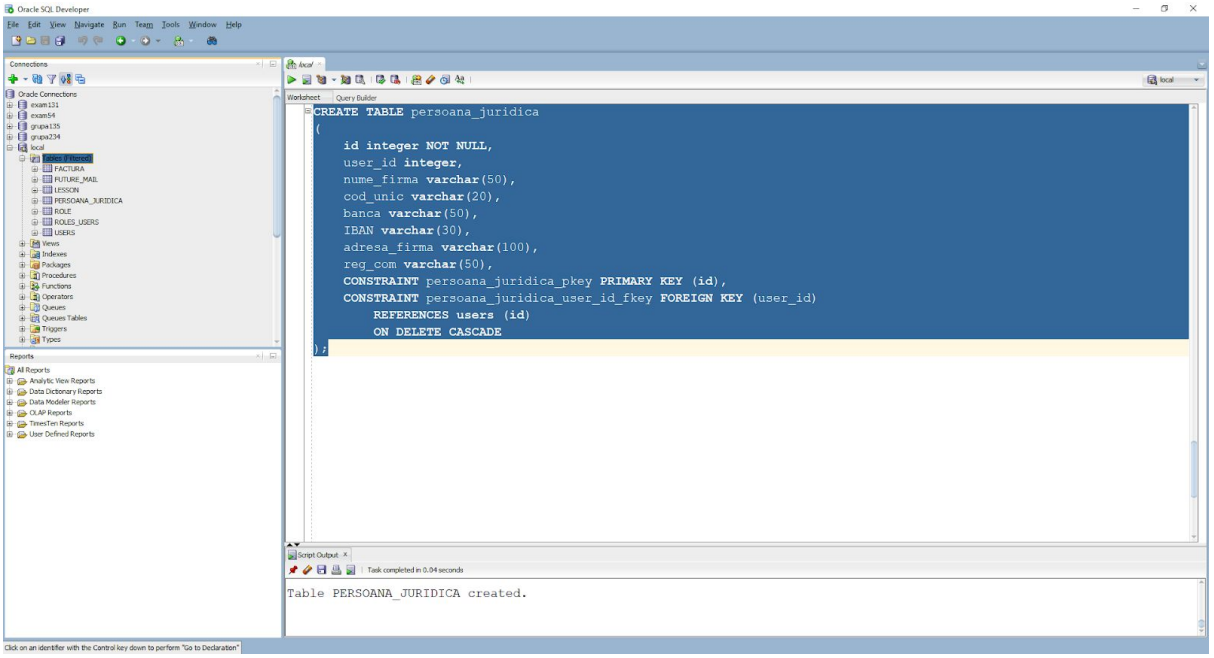



```
CREATE TABLE factura
(
    serie varchar(6),
    id integer NOT NULL,
    data date,
    transaction_id integer,
    CONSTRAINT factura_pkey PRIMARY KEY (serie, id)
);
```

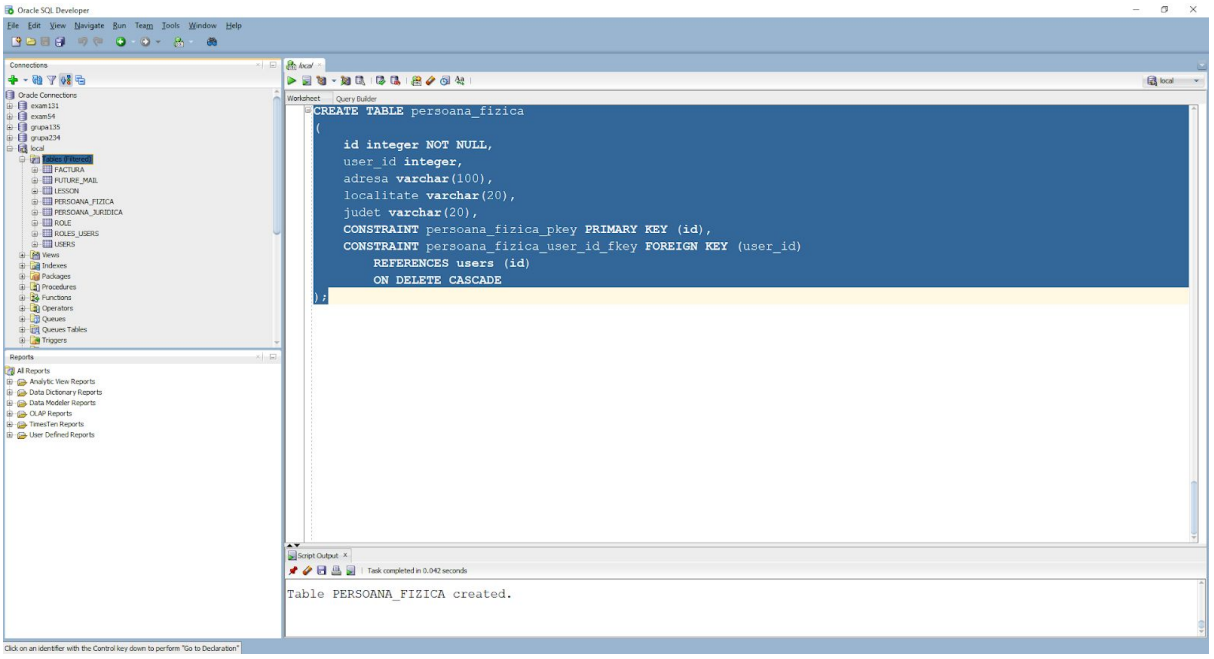



```
CREATE TABLE persoana_juridica
(
    id integer NOT NULL,
    user_id integer,
```

```
    nume_firma varchar(50),
    cod_unic varchar(20),
    banca varchar(50),
    IBAN varchar(30),
    adresa_firma varchar(100),
    reg_com varchar(50),
    CONSTRAINT persoana_juridica_pkey PRIMARY KEY (id),
    CONSTRAINT persoana_juridica_user_id_fkey FOREIGN KEY (user_id)
        REFERENCES users (id)
        ON DELETE CASCADE
);
```




```
CREATE TABLE persoana_fizica
(
    id integer NOT NULL,
    user_id integer,
    adresa varchar(100),
    localitate varchar(20),
    judet varchar(20),
    CONSTRAINT persoana_fizica_pkey PRIMARY KEY (id),
    CONSTRAINT persoana_fizica_user_id_fkey FOREIGN KEY (user_id)
        REFERENCES users (id)
        ON DELETE CASCADE
);
```



5. Adăugați informații coerente în tabelele create (minim 3-5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru tabela asociativă).

```
INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (1, 'predamihaidragos@gmail.com', 'Mihai-Dragos Preda',

'$2y$12$1zeZvb80.c4lK/BEmRa4U.lqS09WyBuKEdRT/ejUkmjyG/5qqDMPO',
        4, 0, TO_DATE('2020-03-15', 'yyyy-mm-dd'), null, '0740764496');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (2, 'mpreda@google.com', 'Mihai Preda Jrr',
        '$2y$12$xvsMfGI.IwZHqs1AUcJeAumCefogcXmPX9a3ALtpsbs7yShI8teja',
        0, 1, TO_DATE('2019-03-15', 'yyyy-mm-dd'), 1, '0740496764');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (3, 'ionmoromete@yahoo.ro', 'Moromete Ion',

'$2y$12$E4bWTnn0hheeFnLTauq/Gesu3eZjyqpxEcbAyLiKzGGCQNyXH0auu',
```

```
        3, 0, TO_DATE('1930-04-26', 'yyyy-mm-dd'), null, '0754231765');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (4, 'paraschiv@hotmail.com', 'Paraschiv Moromete',
        '$2y$12$rbsPx7bfQeC.qkvMN0EqfOosT.FndFwh3ELVb3PTrHqji00jon4Y2',
        5, 0, TO_DATE('1932-07-24', 'yyyy-mm-dd'), 3, '0740193200');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (5, 'achim@hotmail.com', 'Achim Moromete',
        '$2y$12$rPSlFzOaVbYSFp8H09nFtec.rNMjFLA/2tbtAdSRAJs4AqEBB5p8i',
        1, 0, TO_DATE('1934-08-12', 'yyyy-mm-dd'), 3, '0757214302');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (6, 'stefan@gheorghidiu.com', 'Stefan Gheorghidiu',

'$2y$12$mWeeqm.wIme7zT9e3CcjSO7/Ba/KlMdW6/F3GBYz8CXFEjH9l.Wg6',
        2, 1, TO_DATE('1915-08-01', 'yyyy-mm-dd'), null, '0745123546');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (7, 'ela@gheorghidiu.com', 'Ela Gheorghidiu',
        '$2y$12$/4mrGbr2M19If8nnNtcQpeLdj6IXGtaBzBC48s.W0VGlboIxHfFGS',
        5, 0, TO_DATE('1938-07-08', 'yyyy-mm-dd'), 6, '0745712346');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (8, 'omarculescu@pascalopol.com', 'Otilia Marculescu',

'$2y$12$WBHiqwGOaFOiv/tr8KG2tOkqjsSDzeWvM7ToGfxPEWq.pV54dHzXe',
        12, 0, TO_DATE('1939-08-07', 'yyyy-mm-dd'), null, '0789214579');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (9, 'fsima@pascalopol.com', 'Felix Sima',
        '$2y$12$ZUkqlnYdk59AasbcYLecVek3.iwraN2cE9ZiGXKr.dRC0CzKb7jES',
        15, 1, TO_DATE('1938-08-15', 'yyyy-mm-dd'), 8, '0774124367');

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (10, 'harapalb@regatulverde.com', 'Harap Alb',
        '$2y$12$2BQa9SuoSIssj/K66z11.uYPTzvm6d7oNYj/PjfHmZwnLYd87Zgf.',
        2, 0, TO_DATE('2102-12-01', 'yyyy-mm-dd'), null, '0747854124');
```

ID	EMAIL	NAME	PASSWORD	LESSONS	ACTIVE	CONFIRMED_AT	REFERRED_BY	PHONE
1	predamihaidragos@gmail.com	Mihai-Dragos Preda	$2y$12$1zeZvb80.c4lK/BEmRa4U.lqS09WyBuKEdRT/ejUkmjyG/5qqDMPO	4	0	15-MAR-20	(null)	0740764496
2	mpreda@google.com	Mihai Preda Jrr	$2y$12$xvsMfGI.IwZHqslAUcJeAumCefoqcXmPX9a3ALtpsbs7yShI8teja	0	1	15-MAR-19	1	0740496764
3	ionmoromete@yahoo.ro	Moromete Ion	$2y$12$E4bWTnn0hheeFnLTauq/Gesu3eZjyqpxEcbAyLiKzGGCQNyXH0auu	3	0	26-APR-30	(null)	0754231765
4	paraschiv@hotmail.com	Paraschiv Moromete	$2y$12$rbsPx7bfQeC.qkvMN0EqfOosT.FndFwh3ELVb3PTrHqji00jon4Y2	5	0	24-JUL-32	3	0740193200
5	achim@hotmail.com	Achim Moromete	$2y$12$rPSlFzOaVbYSFp8H09nFtec.rNMjFLA/2tbtAdSRAJs4AqEBB5p8i	1	0	12-AUG-34	3	0757214302
6	stefan@gheorghidiu.com	Stefan Gheorghidiu	$2y$12$mWeeqm.wIme7zT9e3CcjSO7/Ba/KlMdW6/F3GBYz8CXFEjH9l.Wq6	2	1	01-AUG-15	(null)	0745123546
7	ela@gheorghidiu.com	Ela Gheorghidiu	$2y$12$/4mrGbr2M19If8nnNtcQpeLdj6IXGtaBzBC48s.W0VGlboIxHfFGS	5	0	08-JUL-38	6	0745712346
8	omarculescu@pascalopol.com	Otilia Marculescu	$2y$12$WBHiqwGOaFOiv/tr8KG2tOkqjsSDzeWvM7ToGfxPEWq.pV54dHzXe	12	0	07-AUG-39	(null)	0789214579
9	fsima@pascalopol.com	Felix Sima	$2y$12$ZUkqlnYdk59AasbcYLecVek3.iwraN2cE9ZiGXKr.dRC0CzKb7jES	15	1	15-AUG-38	8	0774124367
10	harapalb@regatulverde.com	Harap Alb	$2y$12$2BQa9SuoSIssj/K66z11.uYPTzvm6d7oNYi/PjfHmZwnLYd87Zqf.	2	0	01-DEC-02	(null)	0747854124

```
INSERT INTO role(ID, NAME, DESCRIPTION)
VALUES (1, 'Administrator', 'Se ocupa de buna functionare a site-ului');

INSERT INTO role(ID, NAME, DESCRIPTION)
VALUES (2, 'Moderator', 'Se ocupa de continutul de pe site.');

INSERT INTO role(ID, NAME, DESCRIPTION)
VALUES (3, 'Profesor', 'Se ocupa de tinutul lectiilor pe site.');

INSERT INTO role(ID, NAME, DESCRIPTION)
VALUES (4, 'Elev', 'Un utilizator capata statutul de elev dupa prima lectie.');

INSERT INTO role(ID, NAME, DESCRIPTION)
VALUES (5, 'Utilizator', 'Un utilizator normal.');
```

ID	NAME	DESCRIPTION
1	Administrator	Se ocupa de buna functionare a site-ului
2	Moderator	Se ocupa de continutul de pe site.
3	Profesor	Se ocupa de tinutul lectiilor pe site.
4	Elev	Un utilizator capata statutul de elev dupa prima lectie.
5	Utilizator	Un utilizator normal.

```
INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(1, 1);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(2, 1);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(1, 2);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(2, 2);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(1, 3);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(3, 3);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
```

```
VALUES(6, 3);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(10, 2);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(4, 4);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(5, 4);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(7, 4);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(8, 4);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(9, 4);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(10, 4);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(3, 5);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(4, 5);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(5, 5);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(6, 5);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(7, 5);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(8, 5);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
VALUES(9, 5);

INSERT INTO roles_users(USERS_ID, ROLE_ID)
```

VALUES(10, 5);

	USERS_ID	ROLE_ID
1	1	1
2	2	1
3	1	2
4	2	2
5	1	3
6	3	3
7	6	3
8	10	2
9	4	4
10	5	4
11	7	4
12	8	4
13	9	4
14	10	4
15	3	5
16	4	5
17	5	5
18	6	5
19	7	5
20	8	5
21	9	5
22	10	5

INSERT INTO persoana_juridica(ID, USER_ID, NUME_FIRMA, COD_UNIC, BANCA, IBAN, ADRESA_FIRMA, REG_COM)
VALUES(1, 1, 'ITOIT MEDITATII SRL', '43192694', 'Garanti Bank', 'RO63UGBI0000632008603RON', 'Strada 1907, nr 47, Alexandria, Teleorman', 'J34/540/2020');

INSERT INTO persoana_juridica(ID, USER_ID, NUME_FIRMA, COD_UNIC, BANCA, IBAN, ADRESA_FIRMA, REG_COM)
VALUES(2, 3, 'GOOGLE BUCHAREST SRL', '23047266', 'Citi Bank', 'RO47UGBI0000247512349RON', 'Str. C. A. Rosetti 17 A, Sectorul 2, Bucuresti', 'J40/357/2008');

INSERT INTO persoana_juridica(ID, USER_ID, NUME_FIRMA, COD_UNIC, BANCA, IBAN, ADRESA_FIRMA, REG_COM)
VALUES(3, 5, 'ORACLE ROMANIA SRL', '15058256', 'Citi Bank', 'RO47UGBI0000145478568RON', 'Str. Pipera 43 B, Sectorul 2, Bucuresti', 'J40/12387/2002');

INSERT INTO persoana_juridica(ID, USER_ID, NUME_FIRMA, COD_UNIC, BANCA, IBAN, ADRESA_FIRMA, REG_COM)
VALUES(4, 6, 'AMAZON COM SRL', '14376650', 'Raifeissen Bank', 'RO47UGBI0000474231245RON', 'Str. Cpt. Mircea Vasilescu 13 C, Sectorul 4, Bucuresti', 'J40/73/2002');

INSERT INTO persoana_juridica(ID, USER_ID, NUME_FIRMA, COD_UNIC, BANCA, IBAN, ADRESA_FIRMA, REG_COM)
VALUES(5, 10, 'UIPATH SRL', '34737997', 'ING Bank', 'RO47UGBI0000457854126RON', 'Str. Vasile Alecsandri 4 Si 11 C, Sectorul 1, Bucuresti', 'J40/8216/2015');

	ID	USER_ID	NUME_FIRMA	COD_UNIC	BANCA	IBAN	ADRESA_FIRMA	REG_COM
1	1	1	ITOIT MEDITATII SRL	43192694	Garanti Bank	RO63UGBI0000632008603RON	Strada 1907, nr 47, Alexandria, Teleorman	J34/540/2020
2	2	3	GOOGLE BUCHAREST SRL	23047266	Citi Bank	RO47UGBI0000247512349RON	Str. C. A. Rosetti 17 A, Sectorul 2, Bucuresti	J40/357/2008
3	3	5	ORACLE ROMANIA SRL	15058256	Citi Bank	RO47UGBI0000145478568RON	Str. Pipera 43 B, Sectorul 2, Bucuresti	J40/12387/2002
4	4	6	AMAZON COM SRL	14376650	Raifeissen Bank	RO47UGBI0000474231245RON	Str. Cpt. Mircea Vasilescu 13 C, Sectorul 4, Bucuresti	J40/73/2002
5	5	10	UIPATH SRL	34737997	ING Bank	RO47UGBI0000457854126RON	Str. Vasile Alecsandri 4 Si 11 C, Sectorul 1, Bucuresti	J40/8216/2015

INSERT INTO persoana_fizica(ID, USER_ID, ADRESA, LOCALITATE, JUDET)
VALUES(1, 1, 'Strada 1907, nr 47', 'Alexandria', 'Teleorman');

INSERT INTO persoana_fizica(ID, USER_ID, ADRESA, LOCALITATE, JUDET)
VALUES(2, 2, 'Strada Dunarii, nr 87', 'Slatina', 'Olt');

INSERT INTO persoana_fizica(ID, USER_ID, ADRESA, LOCALITATE, JUDET)
VALUES(3, 4, 'Strada Ceahlaul, nr 45', 'Bucuresti', 'Bucuresti');

INSERT INTO persoana_fizica(ID, USER_ID, ADRESA, LOCALITATE, JUDET)
VALUES(4, 7, 'Strada Lotrioara, nr 3', 'Brasov', 'Brasov');

INSERT INTO persoana_fizica(ID, USER_ID, ADRESA, LOCALITATE, JUDET)
VALUES(5, 8, 'Strada George Valsan, nr 48', 'Timisoara', 'Timisoara');

	ID	USER_ID	ADRESA	LOCALITATE	JUDET
1	1	1	Strada 1907, nr 47	Alexandria	Teleorman
2	2	2	Strada Dunarii, nr 87	Slatina	Olt
3	3	4	Strada Ceahlaul, nr 45	Bucuresti	Bucuresti
4	4	7	Strada Lotrioara, nr 3	Brasov	Brasov
5	5	8	Strada George Valsan, nr 48	Timisoara	Timisoara

INSERT INTO future_mail(ID, SUBJECT, RECIPENT, HTML_CONTENT, SEND_DATE)
VALUES(1, 'Nu ai inceput inca?', 'predamihaidragos@gmail.com', '<html>Incepe acum</html>', TO_DATE('2020-12-28 12:00:00', 'yyyy-mm-dd HH24:MI:SS'));

INSERT INTO future_mail(ID, SUBJECT, RECIPENT, HTML_CONTENT, SEND_DATE)

```
VALUES(2, 'Ai o lectie in 24h!', 'ionmoromete@yahoo.ro',
'<html>Reminder</html>', TO_DATE('1930-12-12 14:00:00', 'yyyy-mm-dd
HH24:MI:SS'));

INSERT INTO future_mail(ID, SUBJECT, RECIPENT, HTML_CONTENT,
SEND_DATE)
VALUES(3, 'Ai o lectie in 24h!', 'stefan@gheorghidiu.com',
'<html>Reminder</html>', TO_DATE('1954-09-12 18:00:00', 'yyyy-mm-dd
HH24:MI:SS'));

INSERT INTO future_mail(ID, SUBJECT, RECIPENT, HTML_CONTENT,
SEND_DATE)
VALUES(4, 'Ai o lectie in 24h!', 'omarculescu@pascalopol.com',
'<html>Reminder</html>', TO_DATE('1939-01-11 10:00:00', 'yyyy-mm-dd
HH24:MI:SS'));

INSERT INTO future_mail(ID, SUBJECT, RECIPENT, HTML_CONTENT,
SEND_DATE)
VALUES(5, 'Ai o lectie in 24h!', 'harapalb@regatulverde.com',
'<html>Reminder</html>', TO_DATE('2101-01-15 12:00:00', 'yyyy-mm-dd
HH24:MI:SS'));
```

	ID	SUBJECT	RECIPENT	HTML_CONTENT	SEND_DATE
1	1	Nu ai inceput inca?	predamihaidragos@gmail.com	Incepe acum	28-DEC-20
2	2	Ai o lectie in 24h!	ionmoromete@yahoo.ro	Reminder	12-DEC-30
3	3	Ai o lectie in 24h!	stefan@gheorghidiu.com	Reminder	12-SEP-54
4	4	Ai o lectie in 24h!	omarculescu@pascalopol.com	Reminder	11-JAN-39
5	5	Ai o lectie in 24h!	harapalb@regatulverde.com	Reminder	15-JAN-01

```
INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 1, TO_DATE('1939-08-07', 'yyyy-mm-dd'), 459697848);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 2, TO_DATE('1945-09-02', 'yyyy-mm-dd'), 587668756);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 3, TO_DATE('1968-04-15', 'yyyy-mm-dd'), 575174752);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 4, TO_DATE('1914-02-05', 'yyyy-mm-dd'), 578578587);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 5, TO_DATE('1968-09-25', 'yyyy-mm-dd'), 657867857);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 6, TO_DATE('1924-12-06', 'yyyy-mm-dd'), 578568767);
```

```
INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 7, TO_DATE('2016-07-24', 'yyyy-mm-dd'), 578578785);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 8, TO_DATE('2020-11-03', 'yyyy-mm-dd'), 357445535);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 9, TO_DATE('2018-06-14', 'yyyy-mm-dd'), 213213212);

INSERT INTO factura(SERIE, ID, DATA, TRANSACTION_ID)
VALUES('IT', 10, TO_DATE('2019-09-01', 'yyyy-mm-dd'), 345425441);
```

	SERIE	ID	DATA	TRANSACTION_ID
1	IT	1	07-AUG-39	459697848
2	IT	2	02-SEP-45	587668756
3	IT	3	15-APR-68	575174752
4	IT	4	05-FEB-14	578578587
5	IT	5	25-SEP-68	657867857
6	IT	6	06-DEC-24	578568767
7	IT	7	24-JUL-16	578578785
8	IT	8	03-NOV-20	357445535
9	IT	9	14-JUN-18	213213212
10	IT	10	01-SEP-19	345425441

```
INSERT INTO lesson(ID, START_DATE, END_DATE, SUBJECT,
PROFESSOR_GRADE, STUDENT_GRADE, PROFESSOR_ID, STUDENT_ID,
                STUDENT_FEEDBACK, PROFESSOR_FEEDBACK, REMINDER_ID)
VALUES(1, TO_DATE('2020-12-27 12:00:00', 'yyyy-mm-dd HH24:MI:SS'),
TO_DATE('2020-12-27 14:00:00', 'yyyy-mm-dd HH24:MI:SS'),
     'Cautare binara', 8.6, 9.7, 1, 3, 'Totul foarte bine. Am inteles tot.', 'Elevul
s-a descurcat bine.', null);

INSERT INTO lesson(ID, START_DATE, END_DATE, SUBJECT,
PROFESSOR_GRADE, STUDENT_GRADE, PROFESSOR_ID, STUDENT_ID,
                STUDENT_FEEDBACK, PROFESSOR_FEEDBACK, REMINDER_ID)
VALUES(2, TO_DATE('1930-12-13 14:00:00', 'yyyy-mm-dd HH24:MI:SS'),
TO_DATE('1930-12-13 16:00:00', 'yyyy-mm-dd HH24:MI:SS'),
     null, null, null, 1, 3, null, null, 2);

INSERT INTO lesson(ID, START_DATE, END_DATE, SUBJECT,
PROFESSOR_GRADE, STUDENT_GRADE, PROFESSOR_ID, STUDENT_ID,
                STUDENT_FEEDBACK, PROFESSOR_FEEDBACK, REMINDER_ID)
VALUES(3, TO_DATE('1954-09-13 18:00:00', 'yyyy-mm-dd HH24:MI:SS'),
TO_DATE('1954-09-13 20:00:00', 'yyyy-mm-dd HH24:MI:SS'),
     null, null, null, 3, 6, null, null, 3);
```

```
INSERT INTO lesson(ID, START_DATE, END_DATE, SUBJECT,
PROFESSOR_GRADE, STUDENT_GRADE, PROFESSOR_ID, STUDENT_ID,
                   STUDENT_FEEDBACK, PROFESSOR_FEEDBACK, REMINDER_ID)
VALUES(4, TO_DATE('1939-01-12 10:00:00', 'yyyy-mm-dd HH24:MI:SS'),
TO_DATE('1939-01-12 12:00:00', 'yyyy-mm-dd HH24:MI:SS'),
      null, null, null, 6, 8, null, null, 4);

INSERT INTO lesson(ID, START_DATE, END_DATE, SUBJECT,
PROFESSOR_GRADE, STUDENT_GRADE, PROFESSOR_ID, STUDENT_ID,
                   STUDENT_FEEDBACK, PROFESSOR_FEEDBACK, REMINDER_ID)
VALUES(5, TO_DATE('2101-01-16 12:00:00', 'yyyy-mm-dd HH24:MI:SS'),
TO_DATE('2101-01-16 14:00:00', 'yyyy-mm-dd HH24:MI:SS'),
      null, null, null, 1, 10, null, null, 5);
```

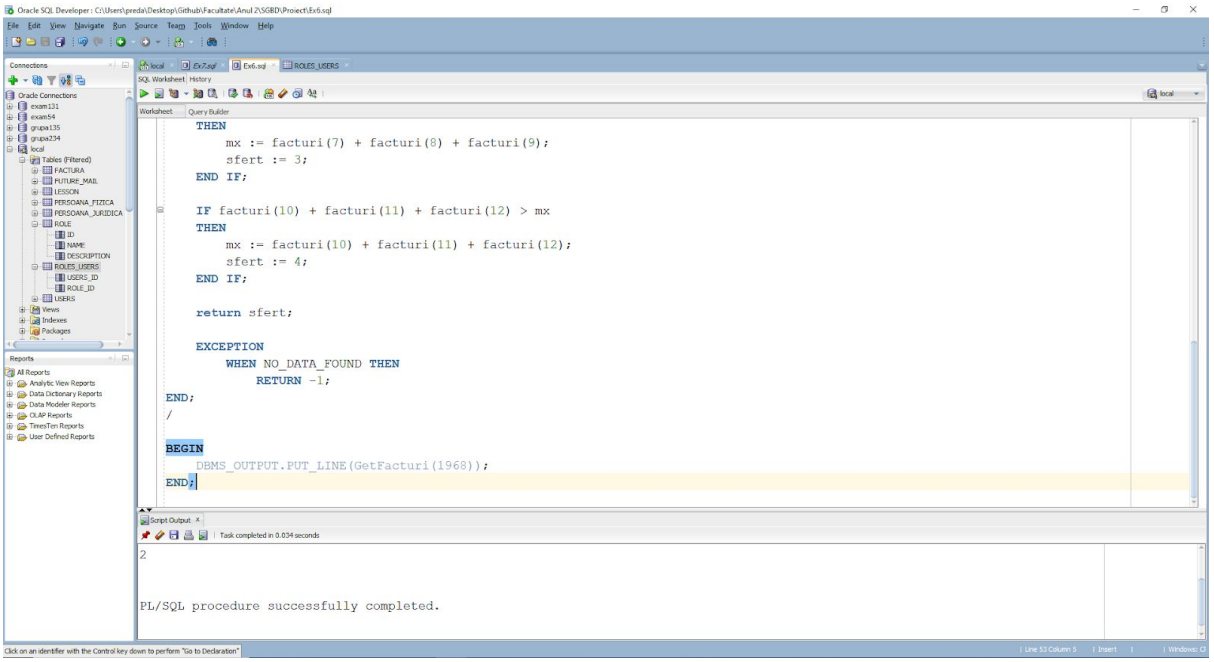
	ID	START_DATE	END_DATE	SUBJECT	PROFESSOR_GRADE	STUDENT_GRADE	PROFESSOR_ID	STUDENT_ID	STUDENT_FEEDBACK	PROFESSOR_FEEDBACK	REMINDER_ID
1	1	27-DEC-20	27-DEC-20	Cautare binara	8.6	9.7	1	3	Totul foarte bine. Am inteles tot.	Elevul s-a descurcat bine.	(null)
2	2	13-DEC-30	13-DEC-30	(null)	(null)	(null)	1	3	(null)	(null)	2
3	3	13-SEP-54	13-SEP-54	(null)	(null)	(null)	3	6	(null)	(null)	3
4	4	12-JAN-39	12-JAN-39	(null)	(null)	(null)	6	8	(null)	(null)	4
5	5	16-JAN-01	16-JAN-01	(null)	(null)	(null)	1	10	(null)	(null)	5

6. Definiți un subprogram stocat care să utilizeze un tip de colecție studiat. Apelați subprogramul.

Cerinta: Definiți un subprogram stocat care returnează în care sfert al anului dat au fost cele mai multe facturi.

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION GetFacturi(
   an NUMBER)
   RETURN NUMBER
IS
   type Tablou IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
   facturi    Tablou;
   mx NUMBER;
   sfert NUMBER;
BEGIN
   FOR i IN 1..12
   LOOP
      SELECT COUNT(*)
      INTO facturi(i)
      FROM factura f
```

```
      WHERE EXTRACT(YEAR FROM f.data) = an AND EXTRACT(MONTH FROM
f.data) = i;
   END LOOP;

   mx := facturi(1) + facturi(2) + facturi(3);
   sfert := 1;

   IF facturi(4) + facturi(5) + facturi(6) > mx
   THEN
      mx := facturi(4) + facturi(5) + facturi(6);
      sfert := 2;
   END IF;

   IF facturi(7) + facturi(8) + facturi(9) > mx
   THEN
      mx := facturi(7) + facturi(8) + facturi(9);
      sfert := 3;
   END IF;

   IF facturi(10) + facturi(11) + facturi(12) > mx
   THEN
      mx := facturi(10) + facturi(11) + facturi(12);
      sfert := 4;
   END IF;

   return sfert;

   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         RETURN -1;
END;
/

BEGIN
   DBMS_OUTPUT.PUT_LINE(GetFacturi(1968));
END;
```

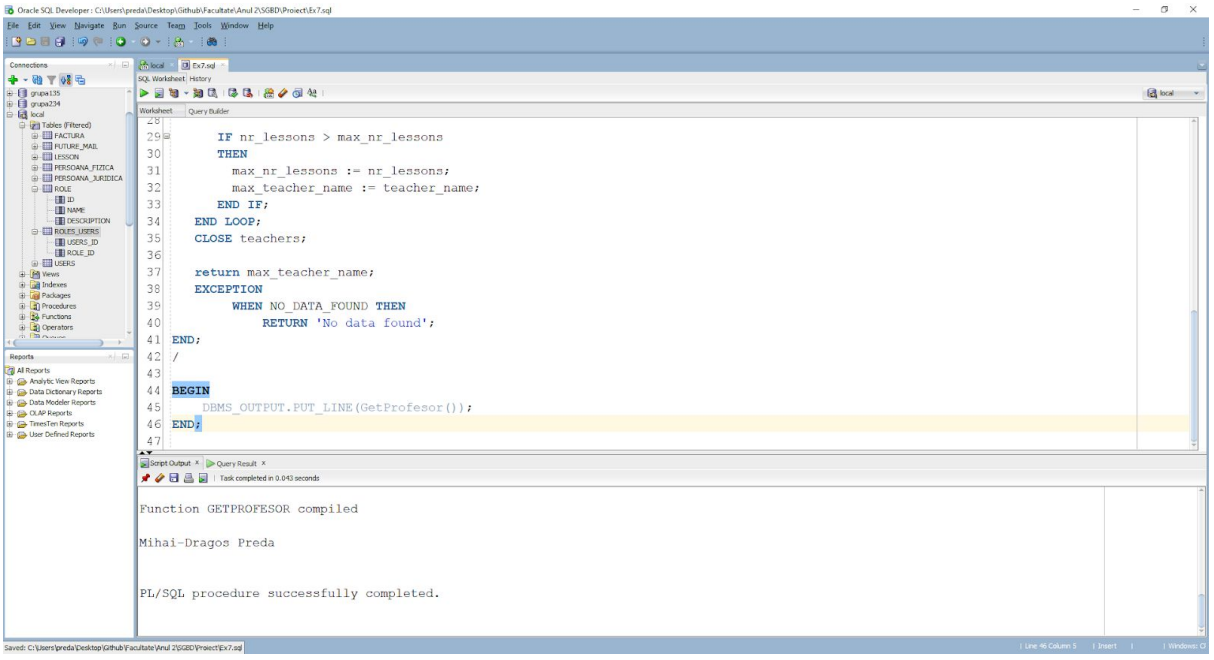


7. Definiți un subprogram stocat care să utilizeze un tip de cursor studiat. Apelați subprogramul.

Cerința: Definiți un subprogram stocat care returneaza numele profesorului cu cele mai multe lecții.

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE FUNCTION GetProfesor
    RETURN users.name%type IS
    CURSOR teachers is (SELECT u.id, u.name
                        FROM users u
                        JOIN roles_users rou ON rou.users_id = u.id
                        JOIN role r ON r.ID = rou.role_id
                        WHERE r.name = 'Profesor');
   teacher_id users.id%type;
   teacher_name users.name%type;
   nr_lessons NUMBER;
   max_nr_lessons NUMBER;
   max_teacher_name users.name%type;
BEGIN
   max_nr_lessons := -1;
   OPEN teachers;
```

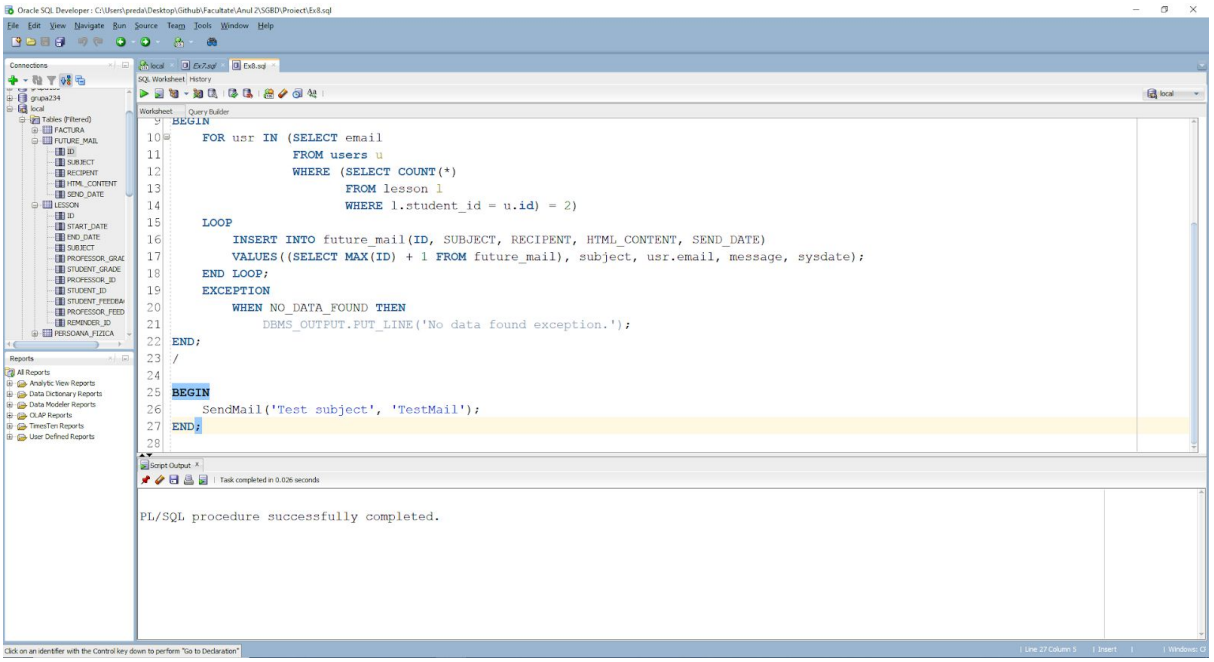
```
  LOOP
  FETCH teachers into teacher_id, teacher_name;
    EXIT WHEN teachers%notfound;
    SELECT COUNT(*)
    INTO nr_lessons
    FROM lesson
    WHERE professor_id = teacher_id;

    IF nr_lessons > max_nr_lessons
    THEN
      max_nr_lessons := nr_lessons;
      max_teacher_name := teacher_name;
    END IF;
  END LOOP;
  CLOSE teachers;

  return max_teacher_name;
  EXCEPTION
      WHEN NO_DATA_FOUND THEN
          RETURN 'No data found';
END;
/

BEGIN
   DBMS_OUTPUT.PUT_LINE(GetProfesor());
END;
```



8. Definiți un subprogram stocat de tip funcție care să utilizeze 3 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Cerința: Definiti un subprogram stocat care insereaza un mail de trimis astăzi userilor care au avut exact 2 lecții.

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE SendMail(
   subject future_mail.subject%type,
   message future_mail.html_content%type)
IS
BEGIN
   FOR usr IN (SELECT email
            FROM users u
            WHERE (SELECT COUNT(*)
                FROM lesson l
                WHERE l.student_id = u.id) = 2)
   LOOP
      INSERT INTO future_mail(ID, SUBJECT, RECIPENT, HTML_CONTENT, SEND_DATE)
      VALUES((SELECT MAX(ID) + 1 FROM future_mail), subject, usr.email, message,
sysdate);
   END LOOP;
   EXCEPTION
      WHEN NO_DATA_FOUND THEN
         DBMS_OUTPUT.PUT_LINE('No data found exception.');
END;
/

BEGIN
   SendMail('Test subject', 'TestMail');
END;
```

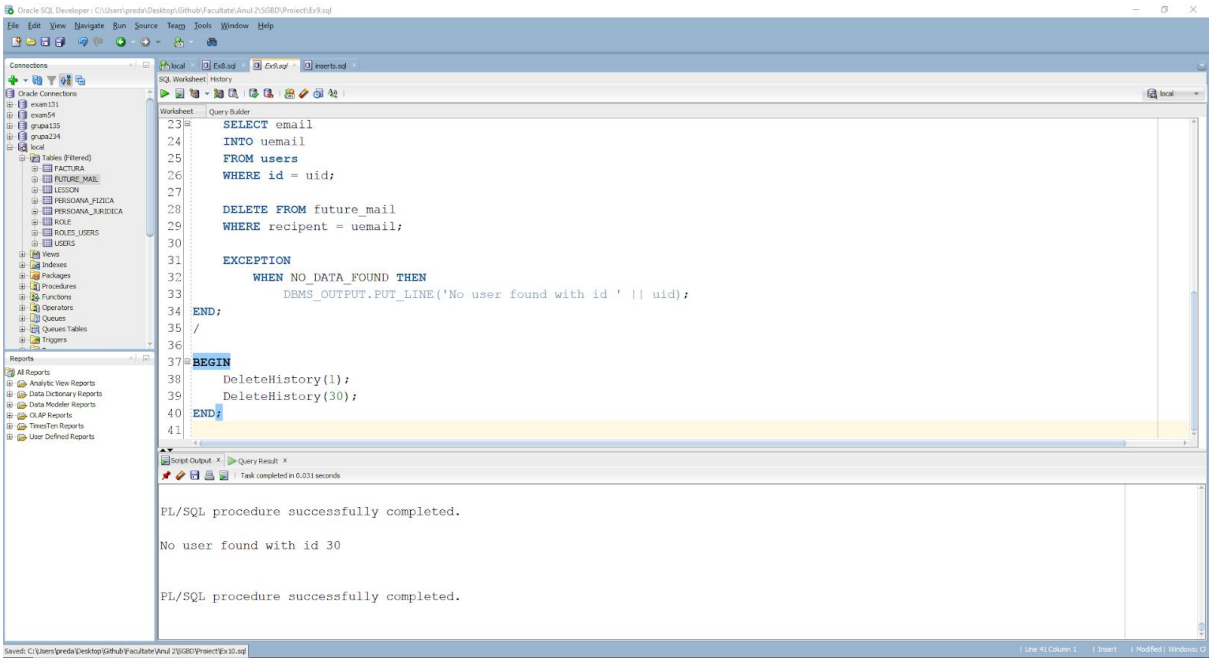


9. Definiți un subprogram stocat de tip procedură care să utilizeze 5 dintre tabelele definite. Tratați toate excepțiile care pot apărea. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Cerinta: Definiti un subprogram stocat sterge tot istoricul unui utilizator (lecțiile, numărul de telefon, datele persoanelor fizice, juridice, mailurile).

```
SET SERVEROUTPUT ON;

CREATE OR REPLACE PROCEDURE DeleteHistory(
    uid users.id%type)
IS
    uemail users.email%type;
BEGIN
    DELETE FROM lesson
    WHERE student_id = uid;

    UPDATE users
    SET phone = NULL
```

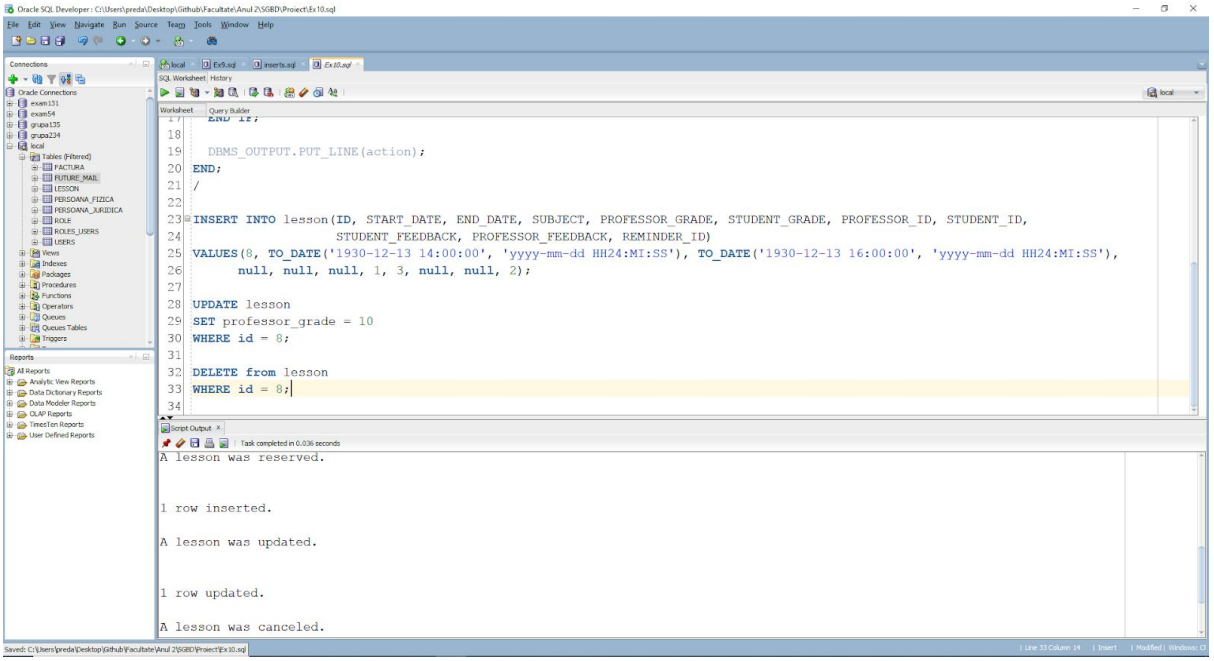
```
    WHERE id = uid;

    DELETE FROM persoana_fizica
    WHERE user_id = uid;

    DELETE FROM persoana_juridica
    WHERE user_id = uid;

    SELECT email
    INTO uemail
    FROM users
    WHERE id = uid;

    DELETE FROM future_mail
    WHERE recipent = uemail;

    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No user found with id ' || uid);
END;
/

BEGIN
    DeleteHistory(1);
    DeleteHistory(30);
END;
```



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

Cerinta: Definiti un trigger pe tabelul lesson care atunci cand se executa o actiune o afiseaza şi în consola pentru a putea avea un istoric a lor.

```
SET SERVEROUTPUT ON;
CREATE OR REPLACE TRIGGER LESSON_TRIGGER
  AFTER INSERT OR UPDATE OR DELETE
  ON lesson
DECLARE
  action VARCHAR(120);
BEGIN
  IF INSERTING THEN
    action := 'A lesson was reserved.';
  ELSIF UPDATING THEN
    action := 'A lesson was updated.';
  ELSIF DELETING THEN
    action := 'A lesson was canceled.';
  ELSE
    action := 'Other action';
  END IF;

  DBMS_OUTPUT.PUT_LINE(action);
END;
/

INSERT INTO lesson(ID, START_DATE, END_DATE, SUBJECT,
PROFESSOR_GRADE, STUDENT_GRADE, PROFESSOR_ID, STUDENT_ID,
                STUDENT_FEEDBACK, PROFESSOR_FEEDBACK, REMINDER_ID)
VALUES(8, TO_DATE('1930-12-13 14:00:00', 'yyyy-mm-dd HH24:MI:SS'),
TO_DATE('1930-12-13 16:00:00', 'yyyy-mm-dd HH24:MI:SS'),
      null, null, null, 1, 3, null, null, 2);

UPDATE lesson
SET professor_grade = 10
WHERE id = 8;

DELETE from lesson
WHERE id = 8;
```

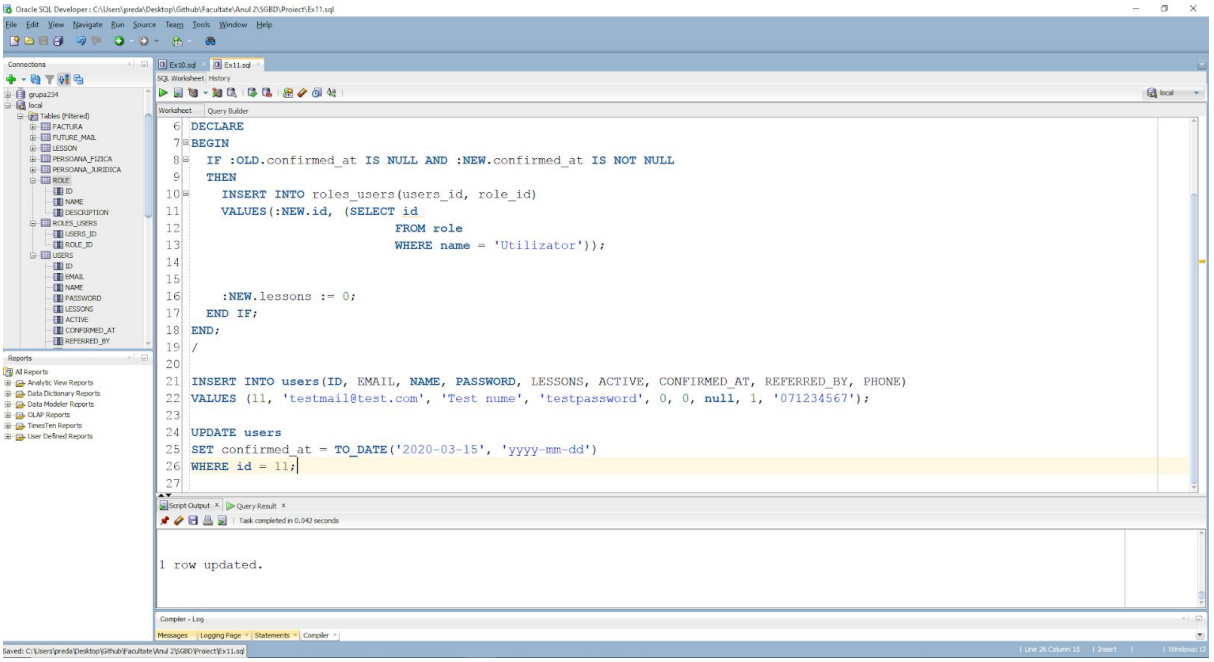


11. Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul.

Cerinta: Definiti un trigger care in momentul in care un user isi confirma contul, ii adauga rolul de utilizator si ii seteaza numarul de lectii la 0.

```
CREATE OR REPLACE TRIGGER ADD_REFERRAL_TRIGGER
BEFORE UPDATE ON users
FOR EACH ROW
DECLARE
BEGIN
  IF :OLD.confirmed_at IS NULL AND :NEW.confirmed_at IS NOT NULL
  THEN
    INSERT INTO roles_users(users_id, role_id)
    VALUES(:NEW.id, (SELECT id
                     FROM role
                     WHERE name = 'Utilizator'));


    :NEW.lessons := 0;
  END IF;
```

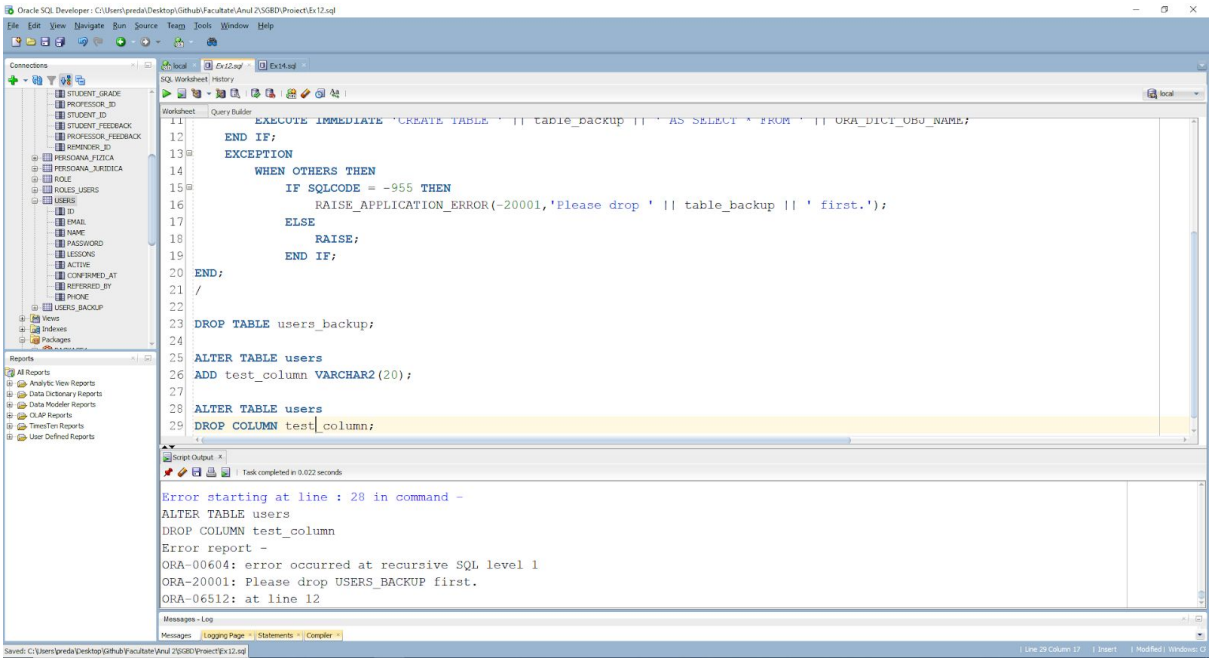
```
END;
/

INSERT INTO users(ID, EMAIL, NAME, PASSWORD, LESSONS, ACTIVE,
CONFIRMED_AT, REFERRED_BY, PHONE)
VALUES (11, 'testmail@test.com', 'Test nume', 'testpassword', 0, 0, null, 1,
'071234567');

UPDATE users
SET confirmed_at = TO_DATE('2020-03-15', 'yyyy-mm-dd')
WHERE id = 11;
```



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

Cerinta: Definiti un trigger LDD care, atunci cand se modifica un tabel, creaza si o versiune de backup a vechiului tabel nemodificat in caz ca ceva merge prost si va fi necesara revenirea la o versiune anterioara.

```
CREATE OR REPLACE TRIGGER BACKUP_TRIGGER_ALTER BEFORE ALTER ON
SCHEMA
DECLARE
   table_backup VARCHAR(50);
BEGIN
   IF ORA_DICT_OBJ_TYPE='TABLE'
   THEN
      table_backup := ORA_DICT_OBJ_NAME || '_BACKUP';
      EXECUTE IMMEDIATE 'CREATE TABLE ' || table_backup || ' AS SELECT *
FROM ' || ORA_DICT_OBJ_NAME;
   END IF;
   EXCEPTION
      WHEN OTHERS THEN
         IF SQLCODE = -955 THEN
            RAISE_APPLICATION_ERROR(-20001,'Please drop ' || table_backup
|| ' first.');
         ELSE
            RAISE;
         END IF;
END;
/

DROP TABLE users_backup;

ALTER TABLE users
ADD test_column VARCHAR2(20);

ALTER TABLE users
DROP COLUMN test_column;
```



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

```
CREATE OR REPLACE PACKAGE sgbd_project AS
   -- 6
   FUNCTION GetFacturi(an NUMBER) RETURN NUMBER;

   -- 7
   FUNCTION GetProfesor RETURN users.name%type;

   -- 8
   PROCEDURE SendMail(subject future_mail.subject%type, message
future_mail.html_content%type);

   -- 9
   PROCEDURE DeleteHistory(uid users.id%type);
END sgbd_project;
/

CREATE OR REPLACE PACKAGE BODY sgbd_project AS
   FUNCTION GetFacturi(an NUMBER)
   RETURN NUMBER
   IS
      type Tablou IS TABLE OF NUMBER INDEX BY BINARY_INTEGER;
      facturi    Tablou;
      mx NUMBER;
      sfert NUMBER;
   BEGIN
      FOR i IN 1..12
      LOOP
         SELECT COUNT(*)
         INTO facturi(i)
         FROM factura f
         WHERE EXTRACT(YEAR FROM f.data) = an AND EXTRACT(MONTH FROM
f.data) = i;
      END LOOP;

      mx := facturi(1) + facturi(2) + facturi(3);
      sfert := 1;

      IF facturi(4) + facturi(5) + facturi(6) > mx
```

```
    THEN
       mx := facturi(4) + facturi(5) + facturi(6);
       sfert := 2;
    END IF;

    IF facturi(7) + facturi(8) + facturi(9) > mx
    THEN
       mx := facturi(7) + facturi(8) + facturi(9);
       sfert := 3;
    END IF;

    IF facturi(10) + facturi(11) + facturi(12) > mx
    THEN
       mx := facturi(10) + facturi(11) + facturi(12);
       sfert := 4;
    END IF;

    return sfert;

    EXCEPTION
       WHEN NO_DATA_FOUND THEN
          RETURN -1;
END GetFacturi;

FUNCTION GetProfesor RETURN users.name%type IS
    CURSOR teachers is (SELECT u.id, u.name
                        FROM users u
                        JOIN roles_users rou ON rou.users_id = u.id
                        JOIN role r ON r.ID = rou.role_id
                        WHERE r.name = 'Profesor');
    teacher_id users.id%type;
    teacher_name users.name%type;
    nr_lessons NUMBER;
    max_nr_lessons NUMBER;
    max_teacher_name users.name%type;
BEGIN
   max_nr_lessons := -1;
   OPEN teachers;
   LOOP
   FETCH teachers into teacher_id, teacher_name;
      EXIT WHEN teachers%notfound;
      SELECT COUNT(*)
      INTO nr_lessons
      FROM lesson
      WHERE professor_id = teacher_id;
```

```
        IF nr_lessons > max_nr_lessons
        THEN
          max_nr_lessons := nr_lessons;
          max_teacher_name := teacher_name;
        END IF;
      END LOOP;
      CLOSE teachers;

      return max_teacher_name;
      EXCEPTION
          WHEN NO_DATA_FOUND THEN
             RETURN 'No data found';
   END GetProfesor;

   PROCEDURE SendMail(
      subject future_mail.subject%type,
      message future_mail.html_content%type) IS
   BEGIN
      FOR usr IN (SELECT email
                  FROM users u
                  WHERE (SELECT COUNT(*)
                         FROM lesson l
                         WHERE l.student_id = u.id) = 2)
      LOOP
         INSERT INTO future_mail(ID, SUBJECT, RECIPENT, HTML_CONTENT,
SEND_DATE)
         VALUES((SELECT MAX(ID) + 1 FROM future_mail), subject, usr.email,
message, sysdate);
      END LOOP;
      EXCEPTION
         WHEN NO_DATA_FOUND THEN
            DBMS_OUTPUT.PUT_LINE('No data found exception.');
   END SendMail;

   PROCEDURE DeleteHistory(uid users.id%type) IS
      uemail users.email%type;
   BEGIN
      DELETE FROM lesson
      WHERE student_id = uid;

      UPDATE users
      SET phone = NULL
      WHERE id = uid;
```

```
        DELETE FROM persoana_fizica
        WHERE user_id = uid;

        DELETE FROM persoana_juridica
        WHERE user_id = uid;

        SELECT email
        INTO uemail
        FROM users
        WHERE id = uid;

        DELETE FROM future_mail
        WHERE recipent = uemail;

        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                DBMS_OUTPUT.PUT_LINE('No user found with id ' || uid);
    END DeleteHistory;
END sgbd_project;
/
```

14. Definiți un pachet care să includă tipuri de date complexe și obiecte necesare pentru acțiuni integrate.

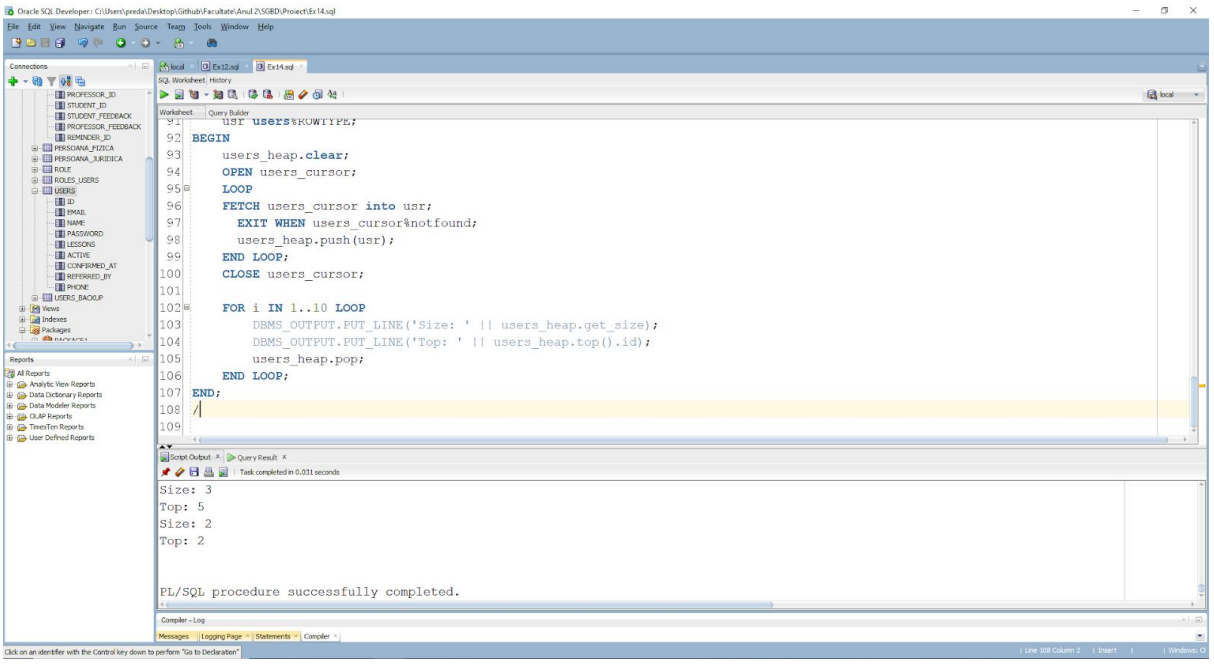
Cerinta: Definiti un pachet care sa actioneze ca un heap cu useri: sa se poata adauga un user, sterge un user și sa se gaseasca userul cu cele mai multe lectii (și sterge).

```
CREATE OR REPLACE PACKAGE users_heap AS
    TYPE Tablou IS TABLE OF users%ROWTYPE;
    users_table Tablou := Tablou();

    -- Adauga un user.
    PROCEDURE push(usr users%ROWTYPE);

    -- Returneaza userul cu cele mai multe lectii.
    FUNCTION top RETURN users%ROWTYPE;

    -- Sterge userul cu cele mai multe lectii.
    PROCEDURE pop;
```

```
   FUNCTION get_size RETURN NUMBER;
   PROCEDURE clear;
END users_heap;
/

CREATE OR REPLACE PACKAGE BODY users_heap AS
   PROCEDURE push(usr users%ROWTYPE) IS
      nr NUMBER;
   BEGIN
      users_table.extend;
      nr := users_table.COUNT;
      users_table(nr) := usr;
   END push;

   FUNCTION top RETURN users%ROWTYPE IS
      max_lessons NUMBER;
      ret users%ROWTYPE;
   BEGIN
      ret := users_table(1);
      max_lessons := ret.lessons;
      FOR i IN 2..users_table.COUNT
      LOOP
         IF users_table(i).lessons > max_lessons
         THEN
            ret := users_table(i);
            max_lessons := ret.lessons;
         END IF;
      END LOOP;
      return ret;
   END top;

   PROCEDURE pop IS
      id_to_delete NUMBER;
      found BOOLEAN;
      aux users%ROWTYPE;
   BEGIN
      id_to_delete := top().id;
      found := false;

      -- Ducem la final userul cu cele mai multe lectii.
      FOR i IN 1..(users_table.COUNT-1)
      LOOP
         IF users_table(i).id = id_to_delete
         THEN
            found := true;
```

```
            END IF;

            IF found = true
            THEN
               aux := users_table(i);
               users_table(i) := users_table(i+1);
               users_table(i+1) := aux;
            END IF;
         END LOOP;

         -- Il stergem pentru ca acum e la final.
         users_table.TRIM;
      END pop;

      FUNCTION get_size RETURN NUMBER IS
      BEGIN
         return users_table.COUNT;
      END get_size;

      PROCEDURE clear IS
      BEGIN
        users_table := Tablou();
      END clear;
   END users_heap;
   /

   SET SERVEROUTPUT ON;

   DECLARE
      CURSOR users_cursor IS (SELECT * FROM users);
      usr users%ROWTYPE;
   BEGIN
      users_heap.clear;
      OPEN users_cursor;
      LOOP
      FETCH users_cursor into usr;
        EXIT WHEN users_cursor%notfound;
        users_heap.push(usr);
      END LOOP;
      CLOSE users_cursor;

      FOR i IN 1..10 LOOP
         DBMS_OUTPUT.PUT_LINE('Size: ' || users_heap.get_size);
         DBMS_OUTPUT.PUT_LINE('Top: ' || users_heap.top().id);
         users_heap.pop;
```

```
    END LOOP;
END;
/
```


Oracle SQL Developer : C:\Users\preda\Desktop\Github\Facultate\Anul 2\SGBD\Proiect\Ex14.sql
File Edit View Navigate Run Source Team Tools Window Help
Connections
PROFESSOR_ID
STUDENT_ID
STUDENT_FEEDBACK
PROFESSOR_FEEDBACK
REMINDER_ID
PERSOANA_FIZICA
PERSOANA_JURIDICA
ROLE
ROLES_USERS
USERS
ID
EMAIL
NAME
PASSWORD
LESSONS
ACTIVE
CONFIRMED_AT
REFERRED_BY
PHONE
USERS_BACKUP
Views
Indexes
Packages
Reports
All Reports
Analytic View Reports
Data Dictionary Reports
Data Modeler Reports
OLAP Reports
TimesTen Reports
User Defined Reports
local
Ex12.sql
Ex14.sql
SQL Worksheet History
Worksheet
Query Builder
usr users%ROWTYPE;
BEGIN
users_heap.clear;
OPEN users_cursor;
LOOP
FETCH users_cursor into usr;
EXIT WHEN users_cursor%notfound;
users_heap.push(usr);
END LOOP;
CLOSE users_cursor;
FOR i IN 1..10 LOOP
DBMS_OUTPUT.PUT_LINE('Size: ' || users_heap.get_size);
DBMS_OUTPUT.PUT_LINE('Top: ' || users_heap.top().id);
users_heap.pop;
END LOOP;
END;
/
Script Output
Query Result
Task completed in 0.031 seconds
Size: 3
Top: 5
Size: 2
Top: 2
PL/SQL procedure successfully completed.
Compiler - Log
Messages
Logging Page
Statements
Compiler
Click on an identifier with the Control key down to perform "Go to Declaration"
Line 108 Column 2
Insert