

# **Proiect Baze de Date**

Aguşoaei Alexandru

Grupa 133

An universitar 2023-2024

## **Baza de date a unei platforme de socializare**

# Cerințe

1. *Descrierea modelului real, a utilității acestuia și a regulilor de funcționare.*
2. *Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului.*
3. *Descrierea entităților, incluzând precizarea cheii primare.*
4. *Descrierea relațiilor, incluzând precizarea cardinalității acestora.*
5. *Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor.*
6. *Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5.*
7. *Realizarea diagramei conceptuale corespunzătoare diagramei entitate-relație proiectate la punctul 6. Diagrama conceptuală obținută trebuie să conțină minimum 7 tabele (fără considerarea subentităților), dintre care cel puțin un tabel asociativ.*
8. *Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7.*
9. *Realizarea normalizării până la forma normală 3 (FN1-FN3).*
10. *Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11).*
11. *Crearea tabelelor în SQL și inserarea de date coerente în fiecare dintre acestea (minimum 5 înregistrări în fiecare tabel neasociativ; minimum 10 înregistrări în tabelele asociative; maxim 30 de înregistrări în fiecare tabel).*
12. *Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:*
  - a) *subcereri sincronizate în care intervin cel puțin 3 tabele*
  - b) *subcereri nesincronizate în clauza FROM*
  - c) *grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) în care intervin cel puțin 3 tabele (în cadrul aceleiași cereri)*
  - d) *ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri)*

*e) utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE*

*f) utilizarea a cel puțin 1 bloc de cerere (clauza WITH)*

*Observație: Într-o cerere se vor regăsi mai multe elemente dintre cele enumerate mai sus, astfel încât cele 5 cereri să le cuprindă pe toate.*

*13. Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri.*

*14. Crearea unei vizualizări complexe. Dați un exemplu de operație LMD permisă pe vizualizarea respectivă și un exemplu de operație LMD nepermisă.*

*15. Formulați în limbaj natural și implementați în SQL: o cerere ce utilizează operația outer-join pe minimum 4 tabele, o cerere ce utilizează operația division și o cerere care implementează analiza top-n.*

*Observație: Cele 3 cereri sunt diferite de cererile de la exercițiul 12.*

*16. Optimizarea unei cereri, aplicând regulile de optimizare ce derivă din proprietățile operatorilor algebrei relaționale. Cererea va fi exprimată prin expresie algebrică, arbore algebric și limbaj (SQL), atât anterior cât și ulterior optimizării.*

*17. a. Realizarea normalizării BCNF, FN4, FN5.*

*b. Aplicarea denormalizării, justificând necesitatea acesteia.*

# ***1.Descrierea modelului real***

În acest proiect voi ilustra cum arata o baza de date a unei platforme de socializare. În era digitală actuală, platformele de socializare au devenit componente esențiale ale vieții cotidiene, conectând milioane de utilizatori din întreaga lume. Aceste platforme permit oamenilor să interacționeze, să împărtășească idei, să formeze comunități și să creeze conținut, transformând modul în care comunicăm și ne organizăm societatea. În spatele funcționării eficiente a acestor platforme complexe se află un element tehnologic critic: baza de date. O bază de date bine proiectată și gestionată este fundamentul pe care se sprijină orice platformă de socializare. Aceasta joacă un rol vital în stocarea, gestionarea și accesarea informațiilor utilizatorilor, facilitând o varietate de funcționalități esențiale pentru experiența utilizatorilor.

## ***2. Prezentarea constrângerilor***

- Un utilizator poate avea mai multe albume de fotografii, iar un album este asociat unui singur utilizator.
- Un album de fotografii poate avea mai multe fotografii, iar o fotografie aparține unui singur album.
- Un utilizator poate avea mai multe playlist-uri muzicale, iar un playlist aparține unui singur utilizator.
- O piesă poate aparține mai multor playlist-uri, iar un playlist poate avea mai multe piese.
- Un utilizator are o unică parolă, iar unei parole îi este asociată un singur utilizator.
- Un istoric de parole poate cuprinde mai multe parole, iar o parolă aparține unui singur istoric de parole.
- Un utilizator poate să fie în mai multe grup-uri, iar un grup poate să conțină mai mulți utilizatori.
- Un utilizator are o unică locație, iar o locație poate fi atribuită mai multor utilizatori.
- O locație are o unică adresă, iar o adresă poate fi asociată mai multor locații.

### ***3. Descrierea entităților***

Pentru modelul de date al aplicației de socializare, Utilizatori, Albume\_fotografii, Fotografii, Parole, Istoric\_parole, Grupuri, Grupuri\_și\_utilizatori, Playlisturi\_muzicale, Piese, Playlisturi\_și\_piese, Locații și Adrese reprezintă entitățile.

Despre entități:

- UTILIZATORI – Persoanele care își crează un cont pe platformă (pk – utilizator\_id).
- ALBUME\_FOTOGRAPHII – Albumele adună la un loc mai multe fotografii ale utilizatorului (pk – album\_id).
- FOTOGRAPHII – O imagine postată de către utilizator (pk – fotografie\_id).
- PAROLE – O parolă unică a unui utilizator pentru a putea accesa contul pe platformă (parola actuală pe care utilizatorul o folosește, pk – parola\_id).
- ISTORIC\_PAROLE – Istoricul de parole grupează parolele unui utilizator pe care acesta le-a schimbat în timp (parole pe care nu le mai folosește, pk – utilizator\_id + data\_schimbare).
- GRUPURI – Grupurile conțin mai mulți utilizatori pentru a face mai eficientă comunicarea și socializarea grupurilor de prieteni (pk – grup\_id).
- GRUPURI\_ȘI\_UTILIZATORI – Pentru rezolvarea relației many to many între grupuri și utilizatori (pk – utilizator\_id + grup\_id).
- PLAYLISTURI\_MUZICALE – Playlisturile asimilează la un loc mai multe piese pe placul utilizatorului (pk – playlist\_id).
- PIESE – Piese muzicale care pot fi ascultate de toți utilizatorii (pk – piesa\_id).
- PLAYLISTURI\_ȘI\_PIESE – Pentru rezolvarea relației many to many între piese și playlisturi (pk – piesa\_id + playlist\_id).
- LOCAȚII – Datele despre utilizatori în privința locației lor (pk – locatie\_id).
- ADRESE – Adresele unde locuiesc utilizatorii (pk – adresa\_id).

## ***4. Descrierea relațiilor***

### ***Utilizatori – Parole (1:1)***

Un utilizator trebuie să aibă o parolă, iar o parolă trebuie să fie atribuită unui unic utilizator. Relația are ca scop asocierea unică necesară pentru un utilizator de a avea o parolă pentru a se conecta la platforma de socializare.

### ***Utilizatori – Istoric\_parole (1:1)***

Un utilizator are un unic istoric de parole. Relația are ca scop stocarea parolelor vechi ale utilizatorului.

### ***Parole – Istoric\_parole (m:0)***

O parolă schimbată poate fi stocată într-un istoric de parole asociat utilizatorului, iar un istoric de parole poate conține mai multe parole sau chiar nici una.

### ***Utilizatori – Albume\_fotografii (0:m)***

Un utilizator poate avea mai multe albume sau nici unul, iar un album trebuie să fie asociat unui unic utilizator.

### ***Albume\_fotografii - Fotografii (m:1)***

Un album foto trebuie să aibă minim o fotografie, iar o fotografie trebuie să fie asociată unui album de fotografii. Relația are ca scop gruparea eficientă a fotografiilor utilizatorilor.

### ***Utilizatori – Playlisturi\_și\_piese (1:m)***

Un utilizator poate avea mai multe playlisturi sau poate să nu aibă deloc, iar un playlist trebuie să fie asociat unui unic utilizator.

### ***Piese – Playlisturi\_muzicale (m:m)***

O piesă poate fi asociată unuia sau mai multor playlisturi, iar un playlist poate avea una sau mai multe piese.

### ***Utilizatori – Locații (m:1)***

Un utilizator trebuie să aibă o locație asociată, iar o locație poate fi asociată cu mai mulți utilizatori.

### ***Locații – Adrese (1:1)***

O locație trebuie să aibă o unică adresă, iar o adresă trebuie să fie asociată unei locații.

### ***Utilizatori – Grupuri (m:m)***

Un utilizator poate să fie în mai multe grupuri, iar un grup poate să conțină mai mulți utilizatori.

## ***5. Descrierea atributelor***

### ***1. Utilizatori***

***Utilizator\_id (PK)*** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare utilizator.

***Nume (Not Null)*** – Variabilă de tip caracter de lungime maxim 100, reprezentând numele utilizatorului.

***Prenume (Not Null)*** – Variabilă de tip caracter de lungime maxim 100, reprezentând prenumele utilizatorului.

***Email (Not Null, Unique)*** – Variabilă de tip caracter de lungime maxim 100, reprezentând emailul personal al utilizatorului.

***Parola\_id (FK)*** – Variabilă de tip întreg de lungime maxim 5, reprezentând id ul de parola din tabela Parole.

***Locatie\_id (FK)*** – Variabilă de tip întreg de lungime maxim 5, reprezentând id ul de locație din tabela Locații.

***Data\_inregistrare (Not Null)*** – Variabilă de tip dată calendaristică, reprezentând data la care s-a facut contul utilizatorului.

utilizatori	
utilizator_id	integer
nume	varchar
prenume	varchar
email	varchar
parola_id	integer
locatie_id	integer
data_inregistrare	date

## 2. Parole

**Parola\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru parola fiecarui utilizator.

**Parola\_nume (Not Null)** – Variabilă de tip caracter de lungime maxim 100, reprezentând parola în sine.

**Nivel\_securitate (Not Null)** – Variabilă de tip întreg de lungime maxim 5, reprezentând nivelul de securitate al parolei reprezentat pe o scară de la 1 la 10.

parole	
parola_id	integer
parola_nume	varchar
nivel_securitate	integer

## 3. Istoric\_parole

**Utilizator\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unui utilizator.

**Data\_schimbare (PK)** – Variabilă de tip dată calendaristică, reprezentând data la care s-a schimbat o anumită parolă.

istoric_parole	
utilizator_id	integer
data_schimbare	date
parola_id	integer
parola_nume	varchar

**Parola\_id (FK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru parola schimbată de utilizator.

**Parola\_nume (Not Null)** – Variabilă de tip caracter de lungime maxim 100, reprezentând parola veche a utilizatorului.

## 4. Locații

**Locatie\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unei locații.

**Adresa\_id (FK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unei adrese.

**Oras (Not Null)** – Variabilă de tip caracter de lungime maxim 100, reprezentând orașul locației.

**Tara (Not Null)** – Variabilă de tip caracter de lungime maxim 100, reprezentând țara locației.

locatii	
locatie_id	integer
adresa_id	integer
oras	varchar
tara	varchar



## 5. Adrese

**Adresa\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unei adrese.

**Strada (-)** – Variabilă de tip caracter de lungime maxim 100, reprezentând numele străzii.

**Numar\_strada (-)** – Variabilă de tip întreg de lungime maxim 5, reprezentând numărul strazii.

**Apartament (-)** – Variabilă de tip întreg de lungime maxim 5, reprezentând numărul apartamentului.

adrese	
adresa_id	integer
strada	varchar
numar_strada	integer
apartament	integer

## 6. Fotografii

**Fotografie\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unei fotografii.

**Album\_id (FK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unui album.

**Titlu (-)** – Variabilă de tip caracter de lungime maxim 100, reprezentând titlul fotografiei.

fotografii	
fotografie_id	integer
album_id	integer
titlu	varchar

## 7. Albume\_fotografii

**Album\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unui album.

**Utilizator\_id (FK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare utilizator.

**Album\_nume (-)** – Variabilă de tip caracter lungime maxim 100, reprezentând titlul albumului.

**Data (Not Null)** – Variabilă de tip dată calendaristică, reprezentând data când s-a creat albumul.

albume_fotografii	
album_id	integer
utilizator_id	integer
album_nume	varchar
data	date

## 8. Grupuri

**Grup\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unui grup.

**Denumire (Not Null)** – Variabilă de tip caracter de lungime maxim 100, reprezentând denumirea grupului.

grupuri	
grup_id	integer
denumire	varchar

## 9. Utilizatori\_si\_grupuri

**Utilizator\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare utilizator.

**Grup\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic al unui grup.

utilizatori_si_grupuri	
utilizator_id	integer
grup_id	integer

## 10. Piese

**Piesa\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare piesă.

**Piesa\_numa (Not Null)** – Variabilă de tip caracter de lungime maxim 100, reprezentând numele piesei.

**Artist\_numa (Not Null)** – Variabilă de tip caracter de lungime maxim 100, reprezentând numele artistului.

piese	
piesa_id	integer
piesa_numa	varchar
artist_numa	varchar

## 11. Playlisturi\_muzicale

**Playlist\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare playlist.

**Playlist\_numa (-)** – Variabilă de tip caracter de lungime maxim 100, reprezentând numele playlistului.

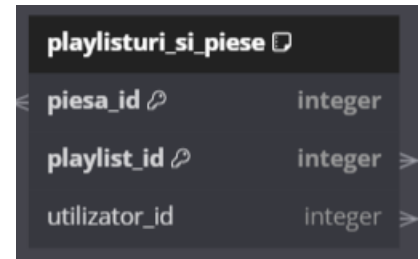
playlisturi_muzicale	
playlist_id	integer
playlist_numa	varchar

## 12. *Playlisturi\_si\_piese*

**Piesa\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare piesă.

**Playlist\_id (PK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare playlist.

**Utilizator\_id (FK)** – Variabilă de tip întreg de lungime maxim 5, reprezentând identificatorul unic pentru fiecare utilizator.

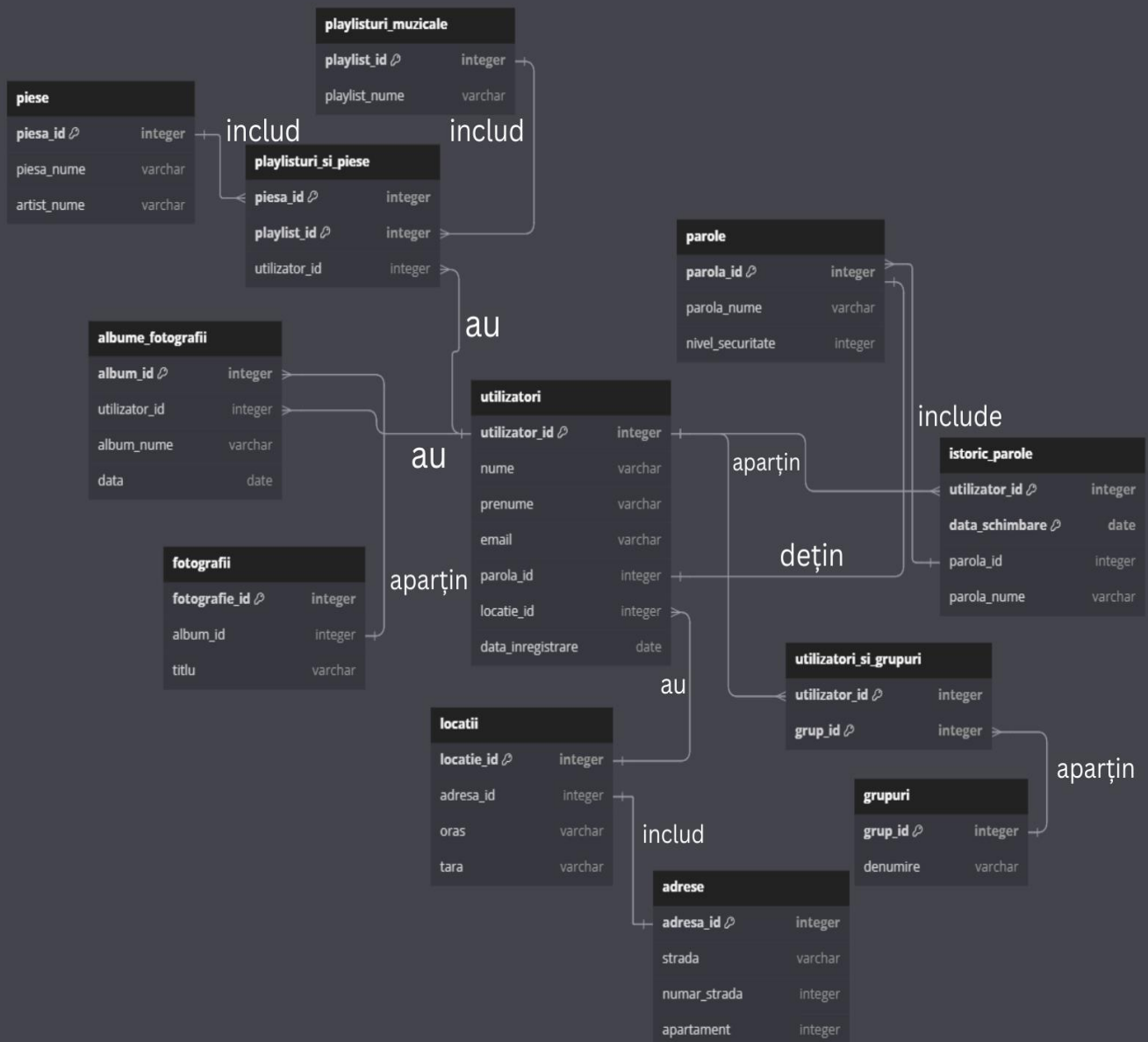


playlisturi_si_piese	
piesa_id	integer
playlist_id	integer
utilizator_id	integer

## 6. Diagrama E/R



## 7. Diagrama Conceptuală



## *8. Scheme relaționale*

Schemele relaționale corespunzătoare diagramei conceptuale sunt următoarele:

- UTILIZATORI (utilizator\_id#, nume, prenume, email, parola\_id, locatie\_id, data\_inregistrare)
- FOTOGRAFII (fotografie\_id#, album\_id, titlu)
- ALBUME\_FOTOGRAFII (album\_id#, utilizator\_id, album\_nume, data)
- PIESE (piesa\_id#, piesa\_nume, artist\_nume)
- PLAYLISTURI\_MUZICALE (playlist\_id#, playlist\_nume)
- PLAYLISTURI\_SI\_PIESE (piesa\_id#, playlist\_id#, utilizator\_id)
- PAROLE (parola\_id#, parola\_nume, nivel\_securitate)
- ISTORIC\_PAROLE (utilizator\_id#, data\_schimbare#, parola\_id, parola\_nume)
- GRUPURI (grup\_id#, denumire)
- UTILIZATORI\_SI\_GRUPURI (utilizator\_id#, grup\_id#)
- LOCAȚII (locatie\_id#, adresa\_id, oras, tara)
- ADRESE (adresa\_id#, strada, numar\_strada, apartament)

## 9. Realizarea normalizării

### Prima Formă Normală (1NF)

**Definiție:** O tabelă este în prima formă normală (1NF) dacă:

1. Toate coloanele conțin doar valori atomice (indivizibile).
2. Toate coloanele din tabelă conțin un singur tip de valoare.
3. Toate valorile dintr-o coloană sunt de același tip de date.
4. Fiecare rând este unic.

### Exemplu (*albume\_fotografii*):

Poate exista cazul în care un utilizator crează 3 albume de fotografii în aceeași zi. Fără FN1 am avea în tabela albume\_fotografii o înregistrare de genul:

Utilizator_id	Album_nume	Data
100	“Primele poze”, “La mare”, “Ziua de naștere“	03.08.2023

Astfel, se vor introduce 3 înregistrări diferite:

Album_id	Utilizator_id	Album_nume	Data
1	100	“Primele poze”	03.08.2023
2	100	“La mare”	03.08.2023
3	100	“Ziua de naștere“	03.08.2023

## A Doua Formă Normală (2NF)

**Definiție:** O tabelă este în a doua formă normală (2NF) dacă:

1. Este în 1NF.
2. Toate atributele non-cheie sunt funcțional dependente de întreaga cheie primară (elimină dependențele parțiale).

**Exemplu** (*utilizatori\_si\_grupuri*):

În tabela *utilizatori\_si\_grupuri* avem cheie primară compusă *utilizator\_id*+*grup\_id*, fără să avem alte atribute, deci este în FN2. Ar fi fost greșit să fie în tabela *utilizatori\_si\_grupuri* și atributul *grup\_nume*, deoarece numele grupului este legat strict de grup, nu și de utilizator. Astfel, modul corect de a proceda este de a pune atributul *grup\_nume* în tabela GRUPURI.

utilizatori_si_grupuri	
<i>utilizator_id</i>	integer
<i>grup_id</i>	integer

grupuri	
<i>grup_id</i>	integer
denumire	varchar

## A Treia Formă Normală (3NF)

**Definiție:** O tabelă este în a treia formă normală (3NF) dacă:

1. Este în 2NF.
2. Toate atributele non-cheie sunt funcțional dependente doar de cheie primară (elimină dependențele tranzitive).

**Exemplu** (*albume\_fotografii*):

Deoarece toate atributele care nu sunt chei (primary sau foreign) nu depind decât de chei, și nu depind unele de altele, ne aflăm în FN3. Ar fi fost greșit să



avem în tabela albume\_fotografii un camp legat de foreign key, în cazul de față utilizator\_id (cum ar fi utilizator\_nume, email, nume, prenume, etc.)

albume_fotografii	
album_id	integer
utilizator_id	integer
album_nume	varchar
data	date

## *10. Crearea unei secvențe utilizată în inserarea înregistrărilor în tabele*

```
1 CREATE SEQUENCE seq
2 START WITH 1
3 INCREMENT BY 1
4 MINVALUE 1
5 MAXVALUE 30;
```

```
CREATE SEQUENCE seq
START WITH 1
INCREMENT BY 1
MINVALUE 1
MAXVALUE 30;
```

## *11. Crearea tabelelor și inserarea datelor.*

### *Utilizatori*

```
104 CREATE TABLE utilizatori
105 (
106     utilizator_id number(5) constraint pk_utilizator primary key,
107     nume varchar(100) constraint nume_utilizator not null,
108     prenume varchar(100) constraint prenume_utilizator not null,
109     email varchar(100) constraint email_utilizator not null,
110     parola_id number(5),
111     locatie_id number(5),
112     data_inregistrare date,
113     constraint fk_parola FOREIGN KEY (parola_id) REFERENCES parole(parola_id),
114     constraint fk_locatie FOREIGN KEY (locatie_id) REFERENCES locatii(locatie_id)
115 );
116
```

CREATE TABLE utilizatori

```
(
    utilizator_id number(5) constraint pk_utilizator primary key,
    nume varchar(100) constraint nume_utilizator not null,
    prenume varchar(100) constraint prenume_utilizator not null,
    email varchar(100) constraint email_utilizator not null,
    parola_id number(5),
    locatie_id number(5),
    data_inregistrare date,
    constraint fk_parola FOREIGN KEY (parola_id) REFERENCES parole(parola_id),
    constraint fk_locatie FOREIGN KEY (locatie_id) REFERENCES locatii(locatie_id)
);
```

```

118 INSERT INTO utilizatori
119 VALUES (seq.NEXTVAL, 'Popescu', 'Andrei', 'popescu_andrei@gmail.com', 1, 1, TO_DATE('04/12/2023', 'DD/MM/YYYY'));
120
121 INSERT INTO utilizatori
122 VALUES (seq.NEXTVAL, 'Dimitrie', 'Roxana', 'dimitrie_roxx@gmail.com', 3, 4, TO_DATE('12/06/2023', 'DD/MM/YYYY'));
123
124 INSERT INTO utilizatori
125 VALUES (seq.NEXTVAL, 'Ionescu', 'Maria', 'ionescu_maria@gmail.com', 2, 2, TO_DATE('05/11/2023', 'DD/MM/YYYY'));
126
127 INSERT INTO utilizatori
128 VALUES (seq.NEXTVAL, 'Georgescu', 'Mihai', 'georgescu_mihai@gmail.com', 5, 3, TO_DATE('06/10/2023', 'DD/MM/YYYY'));
129
130 INSERT INTO utilizatori
131 VALUES (seq.NEXTVAL, 'Vasilescu', 'Elena', 'vasilescu_elena@gmail.com', 4, 3, TO_DATE('22/12/2023', 'DD/MM/YYYY'));
132
133 INSERT INTO utilizatori
134 VALUES (seq.NEXTVAL, 'Ionescu', 'Elena', 'ionescu_elena@gmail.com', 7, 2, TO_DATE('05/11/2023', 'DD/MM/YYYY'));
135
136 select *
137 from utilizatori;
138

```

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Popescu', 'Andrei', 'popescu\_andrei@gmail.com', 1, 1,  
TO\_DATE('04/12/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Dimitrie', 'Roxana', 'dimitrie\_roxx@gmail.com', 3, 4,  
TO\_DATE('12/06/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Ionescu', 'Maria', 'ionescu\_maria@gmail.com', 2, 2, TO\_DATE('05/11/2023',  
'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Georgescu', 'Mihai', 'georgescu\_mihai@gmail.com', 5, 3,  
TO\_DATE('06/10/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Vasilescu', 'Elena', 'vasilescu\_elena@gmail.com', 4, 3,  
TO\_DATE('22/12/2023', 'DD/MM/YYYY'));

INSERT INTO utilizatori

VALUES (seq.NEXTVAL, 'Ionescu', 'Elena', 'ionescu\_elena@gmail.com', 7, 2, TO\_DATE('05/11/2023',  
'DD/MM/YYYY'));

UTILIZATOR_ID	NUME	PRENUME	EMAIL	PAROLA_ID	LOCATIE_ID	DATA_INREGISTRARE
1	21 Popescu	Andrei	popescu andrei@gmail.com	1	1	04-DEC-23
2	22 Dimitrie	Roxana	dimitrie roxx@gmail.com	3	4	12-JUN-23
3	23 Ionescu	Maria	ionescu maria@gmail.com	2	2	05-NOV-23
4	24 Georgescu	Mihai	georgescu mihai@gmail.com	5	3	06-OCT-23
5	25 Vasilescu	Elena	vasilescu elena@gmail.com	4	3	22-DEC-23
6	26 Ionescu	Elena	ionescu elena@gmail.com	7	2	05-NOV-23

## Parole

```

8 CREATE TABLE parole
9 (
10     parola_id number(5) constraint pk_parole primary key,
11     parola_nume varchar(100) constraint parola_nume not null,
12     nivel_securitate number(5) constraint securitate not null
13 );

```

CREATE TABLE parole

```

(
    parola_id number(5) constraint pk_parole primary key,
    parola_nume varchar(100) constraint parola_nume not null,
    nivel_securitate number(5) constraint securitate not null
);

```

```

15 INSERT INTO parole
16 VALUES (1, 'andrei2004', 5);
17
18 INSERT INTO parole
19 VALUES (2, 'asdasdasdasd', 7);
20
21 INSERT INTO parole
22 VALUES (3, '!Bucuresti2024', 9);
23
24 INSERT INTO parole
25 VALUES (4, '!AlexAvg 20', 10);
26
27 INSERT INTO parole
28 VALUES (5, 'tat', 2);
29
30 INSERT INTO parole
31 VALUES (6, 'Ianuarie2004', 8);
32
33 INSERT INTO parole
34 VALUES (7, 'a', 1);
35
36 INSERT INTO parole
37 VALUES (8, 'abc', 2);
38
39 INSERT INTO parole
40 VALUES (9, 'abc', 2);
41
42 INSERT INTO parole
43 VALUES (10, '!Ale 20', 9);

```

PAROLA_ID	PAROLA_NUME	NIVEL_SECURITATE
1	1 andrei2004	5
2	2 asdasdasdasd	7
3	3 !Bucuresti2024	8
4	4 !AlexAvg 20	10
5	5 tat	2
6	6 Ianuarie2004	8
7	7 a	1
8	8 abc	2
9	9 abc	2
10	10 !Ale 20	9

```
INSERT INTO parole
VALUES (1, 'andrei2004', 5);
```

```
INSERT INTO parole
VALUES (2, 'asdasdasdasd', 7);
```

```
INSERT INTO parole
VALUES (3, '!Bucuresti2024', 9);
```

```
INSERT INTO parole
VALUES (4, '!AlexAvg 20', 10);
```

```
INSERT INTO parole
VALUES (5, 'tat', 2);
```

```
INSERT INTO parole
VALUES (6, 'Ianuarie2004', 8);
```

```
INSERT INTO parole
VALUES (7, 'a', 1);
```

```
INSERT INTO parole
VALUES (8, 'abc', 2);
```

```
INSERT INTO parole
VALUES (9, 'abc', 2);
```

```
INSERT INTO parole
VALUES (10, '!Ale 20', 9);
```

```
select *
from parole;
```

## *Adrese*

```
49 CREATE TABLE adrese|
50 (
51     adresa_id number(5) constraint pk_adrese primary key,
52     strada varchar(100) constraint strada not null,
53     numar_strada number(5) constraint nr_strada not null,
54     apartament number(5) constraint apartament not null
55 );
```

CREATE TABLE adrese

```
(
    adresa_id number(5) constraint pk_adrese primary key,
    strada varchar(100) constraint strada not null,
    numar_strada number(5) constraint nr_strada not null,
    apartament number(5) constraint apartament not null
);
```

```
57 INSERT INTO adrese
58 VALUES (1, 'Aleea laleleor', 8, 12);
59
60 INSERT INTO adrese
61 VALUES (2, 'Aleea bratescu', 2, 3);
62
63 INSERT INTO adrese
64 VALUES (3, 'Aleea bratescu', 2, 23);
65
66 INSERT INTO adrese
67 VALUES (4, 'Aleea Drumul Taberei', 6, 1);
68
69 INSERT INTO adrese
70 VALUES (5, 'Aleea Arena', 11, 8);
71
72 select *
73 from adrese;
```

```
INSERT INTO adrese
VALUES (1, 'Aleea laleleor', 8, 12);
```

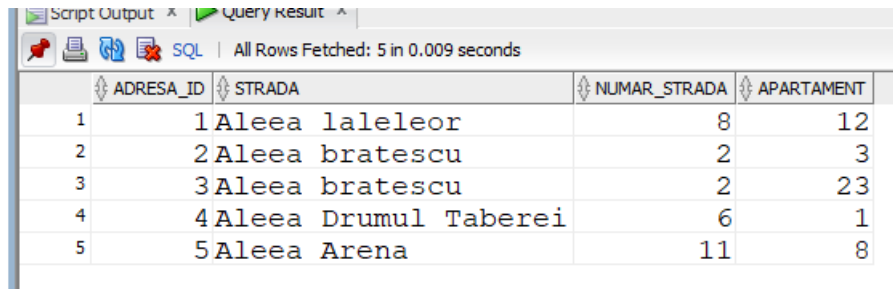
```
INSERT INTO adrese
VALUES (2, 'Aleea bratescu', 2, 3);
```

```
INSERT INTO adrese
VALUES (3, 'Aleea bratescu', 2, 23);
```

```
INSERT INTO adrese
VALUES (4, 'Aleea Drumul Taberei', 6, 1);
```

```
INSERT INTO adrese
VALUES (5, 'Aleea Arena', 11, 8);
```

```
select *
from adrese;
```



The screenshot shows a SQL query result window with the title 'Query Result'. It displays the results of a query that fetched all rows from the 'adrese' table. The window shows a table with 5 rows and 4 columns: ADRESA\_ID, STRADA, NUMAR\_STRADA, and APARTAMENT. The data is as follows:

	ADRESA_ID	STRADA	NUMAR_STRADA	APARTAMENT
1	1	Aleea laleleor	8	12
2	2	Aleea bratescu	2	3
3	3	Aleea bratescu	2	23
4	4	Aleea Drumul Taberei	6	1
5	5	Aleea Arena	11	8

## *Locatii*

```
76 CREATE TABLE locatii
77 (
78     locatie_id number(5) constraint pk_locatii primary key,
79     adresa_id number(5),
80     oras varchar(100) constraint oras not null,
81     tara varchar(100) constraint tara not null,
82     constraint fk_adresa FOREIGN KEY (adresa_id) REFERENCES adrese(adresa_id)
83 );
```

```

CREATE TABLE locatii
(
    locatie_id number(5) constraint pk_locatii primary key,
    adresa_id number(5),
    oras varchar(100) constraint oras not null,
    tara varchar(100) constraint tara not null,
    constraint fk_adresa FOREIGN KEY (adresa_id) REFERENCES adrese(adresa_id)
);

```

```

85 INSERT INTO locatii
86 VALUES (1, 1, 'Bucuresti', 'Romania');
87
88 INSERT INTO locatii
89 VALUES (2, 5, 'Bucuresti', 'Romania');
90
91 INSERT INTO locatii
92 VALUES (3, 2, 'Suceava', 'Romania');
93
94 INSERT INTO locatii
95 VALUES (4, 3, 'Atena', 'Grecia');
96
97 INSERT INTO locatii
98 VALUES (5, 4, 'Bucuresti', 'Romania');
99
100 select *
101 from locatii;

```

INSERT INTO locatii

VALUES (1, 1, 'Bucuresti', 'Romania');

INSERT INTO locatii

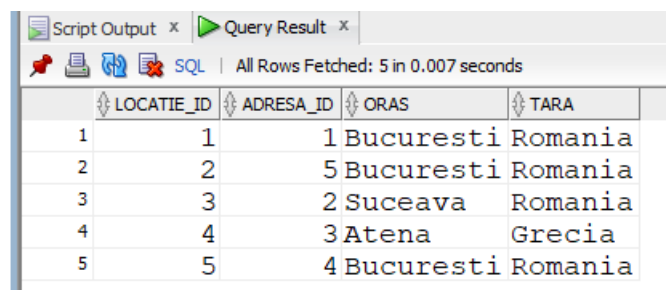
VALUES (2, 5, 'Bucuresti', 'Romania');

INSERT INTO locatii

VALUES (3, 2, 'Suceava', 'Romania');

INSERT INTO locatii

VALUES (4, 3, 'Atena', 'Grecia');



The screenshot shows a 'Query Result' window with the following data:

LOCATIE_ID	ADRESA_ID	ORAS	TARA
1	1	Bucuresti	Romania
2	5	Bucuresti	Romania
3	2	Suceava	Romania
4	3	Atena	Grecia
5	4	Bucuresti	Romania



```
INSERT INTO locatii  
VALUES (5, 4, 'Bucuresti', 'Romania');
```

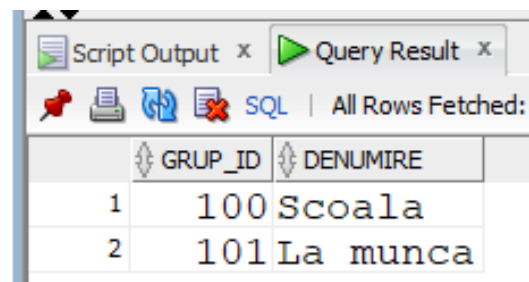
```
select *  
from locatii;
```

## Grupuri

```
140 CREATE TABLE grupuri  
141 (  
142     grup_id number(5) constraint pk_grup primary key,  
143     denumire varchar(100) constraint denumire_grup not null  
144 );  
145  
146 INSERT INTO grupuri  
147 VALUES (100, 'Scoala');  
148  
149 INSERT INTO grupuri  
150 VALUES (101, 'La munca');  
151  
152 select *  
153 from grupuri;
```

```
CREATE TABLE grupuri  
(  
    grup_id number(5) constraint pk_grup primary key,  
    denumire varchar(100) constraint denumire_grup not null  
);
```

```
INSERT INTO grupuri  
VALUES (100, 'Scoala');  
  
INSERT INTO grupuri  
VALUES (101, 'La munca');  
  
select *  
from grupuri;
```



The screenshot shows a database query result window with two tabs: 'Script Output' and 'Query Result'. The 'Query Result' tab is active, displaying a table with two columns: 'GRUP\_ID' and 'DENUMIRE'. The table contains two rows of data: the first row has '100' for 'GRUP\_ID' and 'Scoala' for 'DENUMIRE'; the second row has '101' for 'GRUP\_ID' and 'La munca' for 'DENUMIRE'.

	GRUP_ID	DENUMIRE
1	100	Scoala
2	101	La munca

## *Utilizatori\_si\_grupuri*

```
156 CREATE TABLE utilizatori_si_grupuri
157 (
158     utilizator_id number(5),
159     grup_id number(5),
160     constraint pk_util_gr primary key (utilizator_id, grup_id)
161 );
```

```
CREATE TABLE utilizatori_si_grupuri
(
    utilizator_id number(5),
    grup_id number(5),
    constraint pk_util_gr primary key (utilizator_id, grup_id)
);
```

```
163 INSERT INTO utilizatori_si_grupuri
164 VALUES (21, 100);
165
166 INSERT INTO utilizatori_si_grupuri
167 VALUES (22, 100);
168
169 INSERT INTO utilizatori_si_grupuri
170 VALUES (21, 101);
171
172 INSERT INTO utilizatori_si_grupuri
173 VALUES (23, 101);
174
175 INSERT INTO utilizatori_si_grupuri
176 VALUES (24, 101);
177
178 INSERT INTO utilizatori_si_grupuri
179 VALUES (25, 101);
180
181 INSERT INTO utilizatori_si_grupuri
182 VALUES (26, 101);
183
184 select *
185 from utilizatori_si_grupuri;
```

```
INSERT INTO utilizatori_si_grupuri  
VALUES (21, 100);
```

```
INSERT INTO utilizatori_si_grupuri  
VALUES (22, 100);
```

```
INSERT INTO utilizatori_si_grupuri  
VALUES (21, 101);
```

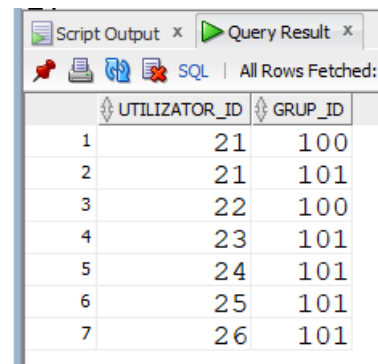
```
INSERT INTO utilizatori_si_grupuri  
VALUES (23, 101);
```

```
INSERT INTO utilizatori_si_grupuri  
VALUES (24, 101);
```

```
INSERT INTO utilizatori_si_grupuri  
VALUES (25, 101);
```

```
INSERT INTO utilizatori_si_grupuri  
VALUES (26, 101);
```

```
select *  
from utilizatori_si_grupuri;
```



	UTILIZATOR_ID	GRUP_ID
1	21	100
2	21	101
3	22	100
4	23	101
5	24	101
6	25	101
7	26	101

## *Fotografii*

```
188 CREATE TABLE fotografiii  
189 (  
190     fotografie_id number(5) constraint pk_foto primary key,  
191     album_id number(5),  
192     titlu varchar(100) not null,  
193     constraint fk_album FOREIGN KEY (album_id) REFERENCES albume_fotografii(album_id)  
194 );
```

```
CREATE TABLE fotografii
```

```
(
```

```
    fotografie_id number(5) constraint pk_foto primary key,
```

```
    album_id number(5),
```

```
    titlu varchar(100) not null,
```

```
    constraint fk_album FOREIGN KEY (album_id) REFERENCES albume_fotografii(album_id)
```

```
);
```

```
INSERT INTO fotografii
```

```
VALUES (1, 100, 'Poza1');
```

```
INSERT INTO fotografii
```

```
VALUES (2, 100, 'Piscina');
```

```
INSERT INTO fotografii
```

```
VALUES (3, 100, 'Poza speciala');
```

```
INSERT INTO fotografii
```

```
VALUES (4, 101, 'Prima poza');
```

```
INSERT INTO fotografii
```

```
VALUES (5, 102, 'Rasarit');
```

```
INSERT INTO fotografii
```

```
VALUES (6, 102, 'La meci');
```

```
INSERT INTO fotografii
```

```
VALUES (7, 103, 'Poza Anglia');
```

```
INSERT INTO fotografii
```



```
VALUES (8, 104, 'Gratar');
```

```
select *
```

```
from fotografii;
```

```
196 INSERT INTO fotografii
197 VALUES (1, 100, 'Poza1');
198
199 INSERT INTO fotografii
200 VALUES (2, 100, 'Piscina');
201
202 INSERT INTO fotografii
203 VALUES (3, 100, 'Poza speciala');
204
205 INSERT INTO fotografii
206 VALUES (4, 101, 'Prima poza');
207
208 INSERT INTO fotografii
209 VALUES (5, 102, 'Rasarit');
210
211 INSERT INTO fotografii
212 VALUES (6, 102, 'La meci');
213
214 INSERT INTO fotografii
215 VALUES (7, 103, 'Poza Anglia');
216
217 INSERT INTO fotografii
218 VALUES (8, 104, 'Gratar');
219
220 select *
221 from fotografii;
```

Script Output x Query Result x

   SQL | All Rows Fetched: 8 in 0.002 seconds

	FOTOGRAFIE_ID	ALBUM_ID	TITLU
1	1	100	Poza1
2	2	100	Piscina
3	3	100	Poza speciala
4	4	101	Prima poza
5	5	102	Rasarit
6	6	102	La meci
7	7	103	Poza Anglia
8	8	104	Gratar

## *Albume\_fotografii*

```
224 CREATE TABLE albume_fotografii
225 (
226     album_id number(5) constraint pk_album primary key,
227     utilizator_id number(5),
228     album_nume varchar(100) constraint nume_album not null,
229     data_album date,
230     constraint fk_util FOREIGN KEY (utilizator_id) REFERENCES utilizatori(utilizator_id)
231 );
232
233 INSERT INTO albume_fotografii
234 VALUES (100, 21, 'La mare', TO_DATE('25/12/2023', 'DD/MM/YYYY'));
235
236 INSERT INTO albume_fotografii
237 VALUES (101, 21, 'Ziua mea', TO_DATE('01/02/2024', 'DD/MM/YYYY'));
238
239 INSERT INTO albume_fotografii
240 VALUES (102, 23, 'Scoala', TO_DATE('29/11/2023', 'DD/MM/YYYY'));
241
242 INSERT INTO albume_fotografii
243 VALUES (103, 24, 'Facultate', TO_DATE('15/08/2023', 'DD/MM/YYYY'));
244
245 INSERT INTO albume_fotografii
246 VALUES (104, 24, 'La mare', TO_DATE('02/03/2024', 'DD/MM/YYYY'));
247
248 select *
249 from albume_fotografii;
```

CREATE TABLE albume\_fotografii

```
(
    album_id number(5) constraint pk_album primary key,
    utilizator_id number(5),
    album_nume varchar(100) constraint nume_album not null,
    data_album date,
    constraint fk_util FOREIGN KEY (utilizator_id) REFERENCES utilizatori(utilizator_id)
);
```

INSERT INTO albume\_fotografii

VALUES (100, 21, 'La mare', TO\_DATE('25/12/2023', 'DD/MM/YYYY'));

INSERT INTO albume\_fotografii

VALUES (101, 21, 'Ziua mea', TO\_DATE('01/02/2024', 'DD/MM/YYYY'));

INSERT INTO albume\_fotografii

```
VALUES (102, 23, 'Scoala', TO_DATE('29/11/2023', 'DD/MM/YYYY'));
```

```
INSERT INTO albume_fotografii
```

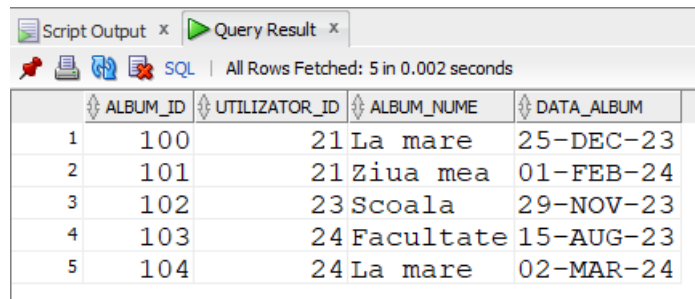
```
VALUES (103, 24, 'Facultate', TO_DATE('15/08/2023', 'DD/MM/YYYY'));
```

```
INSERT INTO albume_fotografii
```

```
VALUES (104, 24, 'La mare', TO_DATE('02/03/2024', 'DD/MM/YYYY'));
```

```
select *
```

```
from albume_fotografii;
```



The screenshot shows a 'Query Result' window with 5 rows of data. The columns are ALBUM\_ID, UTILIZATOR\_ID, ALBUM\_NUME, and DATA\_ALBUM. The data is as follows:

	ALBUM_ID	UTILIZATOR_ID	ALBUM_NUME	DATA_ALBUM
1	100	21	La mare	25-DEC-23
2	101	21	Ziua mea	01-FEB-24
3	102	23	Scoala	29-NOV-23
4	103	24	Facultate	15-AUG-23
5	104	24	La mare	02-MAR-24

## *Piese*

```
252 CREATE TABLE piese
253 (
254     piesa_id number(5) constraint pk_piesa primary key,
255     piesa_numa varchar(100) not null,
256     artist_numa varchar(100) not null
257 );
258
259 INSERT INTO piese (piesa_id, piesa_numa, artist_numa)
260 VALUES (1, 'Shape of You', 'Ed Sheeran');
261
262 INSERT INTO piese (piesa_id, piesa_numa, artist_numa)
263 VALUES (2, 'Blinding Lights', 'The Weeknd');
264
265 INSERT INTO piese (piesa_id, piesa_numa, artist_numa)
266 VALUES (3, 'Levitating', 'Dua Lipa');
267
```

```
CREATE TABLE piese
```

```
(
```

```
    piesa_id number(5) constraint pk_piesa primary key,
```

```
    piesa_numa varchar(100) not null,
```

```
    artist_ume varchar(100) not null  
);
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (1, 'Shape of You', 'Ed Sheeran');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (2, 'Blinding Lights', 'The Weeknd');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (3, 'Levitating', 'Dua Lipa');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (4, 'Bad Guy', 'Billie Eilish');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (5, 'Rockstar', 'Post Malone');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (6, 'Old Town Road', 'Lil Nas X');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (7, 'Senorita', 'Camila Cabello');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (8, 'Circles', 'Post Malone');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)  
VALUES (9, 'Havana', 'Camila Cabello');
```

```
INSERT INTO piese (piesa_id, piesa_ume, artist_ume)
```

VALUES (10, 'My Eyes', 'Travis Scott');

select \*

from piese;

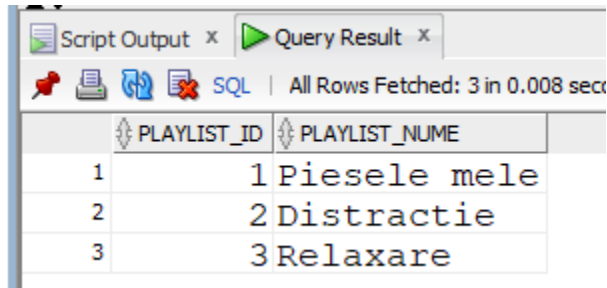
```
268 INSERT INTO piese (piesa_id, piesa_num, artist_num)
269 VALUES (4, 'Bad Guy', 'Billie Eilish');
270
271 INSERT INTO piese (piesa_id, piesa_num, artist_num)
272 VALUES (5, 'Rockstar', 'Post Malone');
273
274 INSERT INTO piese (piesa_id, piesa_num, artist_num)
275 VALUES (6, 'Old Town Road', 'Lil Nas X');
276
277 INSERT INTO piese (piesa_id, piesa_num, artist_num)
278 VALUES (7, 'Senorita', 'Camila Cabello');
279
280 INSERT INTO piese (piesa_id, piesa_num, artist_num)
281 VALUES (8, 'Circles', 'Post Malone');
282
283 INSERT INTO piese (piesa_id, piesa_num, artist_num)
284 VALUES (9, 'Havana', 'Camila Cabello');
285
286 INSERT INTO piese (piesa_id, piesa_num, artist_num)
287 VALUES (10, 'My Eyes', 'Travis Scott');
288
289 select *
290 from piese;
```

Script Output x Query Result x		
SQL   All Rows Fetched: 10 in 0.002 seconds		
PIESA_ID	PIESA_NUM	ARTIST_NUM
1	10 My Eyes	Travis Scott
2	9 Havana	Camila Cabello
3	8 Circles	Post Malone
4	7 Senorita	Camila Cabello
5	6 Old Town Road	Lil Nas X
6	5 Rockstar	Post Malone
7	4 Bad Guy	Billie Eilish
8	3 Levitating	Dua Lipa
9	2 Blinding Lights	The Weeknd
10	1 Shape of You	Ed Sheeran



## *Playlisturi\_muzicale*

```
293 CREATE TABLE playlisturi_muzicale
294 (
295     playlist_id number(5) constraint pk_playlist primary key,
296     playlist_nume varchar(100) not null
297 );
298
299 INSERT INTO playlisturi_muzicale
300 VALUES (1, 'Pieseles mele');
301
302 INSERT INTO playlisturi_muzicale
303 VALUES (2, 'Distractie');
304
305 INSERT INTO playlisturi_muzicale
306 VALUES (3, 'Relaxare');
307
308 select *
309 from playlisturi_muzicale;
```



The screenshot shows a database interface with a 'Query Result' window. It displays the results of a SELECT query on the 'playlisturi\_muzicale' table. The window has a toolbar with icons for saving, printing, and other actions. The status bar indicates 'All Rows Fetched: 3 in 0.008 sec'. The table has two columns: 'PLAYLIST\_ID' and 'PLAYLIST\_NUME'. The data is as follows:

PLAYLIST_ID	PLAYLIST_NUME
1	Pieseles mele
2	Distractie
3	Relaxare

```
CREATE TABLE playlisturi_muzicale
(
    playlist_id number(5) constraint pk_playlist primary key,
    playlist_nume varchar(100) not null
);
```

```
INSERT INTO playlisturi_muzicale
VALUES (1, 'Pieseles mele');
```

```
INSERT INTO playlisturi_muzicale
VALUES (2, 'Distractie');
```

```
INSERT INTO playlisturi_muzicale  
VALUES (3, 'Relaxare');
```

```
select *  
from playlisturi_muzicale;
```

### *Playlisturi\_si\_piese*

```
312 CREATE TABLE playlisturi_si_piese  
313 (  
314     piesa_id number(5),  
315     playlist_id number(5),  
316     utilizator_id number(5),  
317     constraint pk_playlist_piese primary key (piesa_id, playlist_id),  
318     constraint fk_utiliz foreign key (utilizator_id) REFERENCES utilizatori(utilizator_id)  
319 );  
320  
321 INSERT INTO playlisturi_si_piese  
322 VALUES (1, 1, 21);  
323  
324 INSERT INTO playlisturi_si_piese  
325 VALUES (2, 1, 21);  
326  
327 INSERT INTO playlisturi_si_piese  
328 VALUES (5, 1, 21);  
329  
330 INSERT INTO playlisturi_si_piese  
331 VALUES (1, 2, 23);  
332  
333 INSERT INTO playlisturi_si_piese  
334 VALUES (8, 3, 24);  
335  
336 INSERT INTO playlisturi_si_piese  
337 VALUES (9, 3, 24);  
338  
339 INSERT INTO playlisturi_si_piese  
340 VALUES (2, 3, 24);
```

```
CREATE TABLE playlisturi_si_piese  
(  
    piesa_id number(5),  
    playlist_id number(5),  
    utilizator_id number(5),
```

```

constraint pk_playlist_piese primary key (piesa_id, playlist_id),
constraint fk_utiliz foreign key (utilizator_id) REFERENCES utilizatori(utilizator_id)
);

```

```

INSERT INTO playlisturi_si_piese
VALUES (1, 1, 21);

```

```

INSERT INTO playlisturi_si_piese
VALUES (2, 1, 21);

```

```

INSERT INTO playlisturi_si_piese
VALUES (5, 1, 21);

```

```

INSERT INTO playlisturi_si_piese
VALUES (1, 2, 23);

```

```

INSERT INTO playlisturi_si_piese
VALUES (8, 3, 24);

```

```

INSERT INTO playlisturi_si_piese
VALUES (9, 3, 24);

```

```

INSERT INTO playlisturi_si_piese
VALUES (2, 3, 24);

```

```

select *
from playlisturi_si_piese;

```

	PIESA_ID	PLAYLIST_ID	UTILIZATOR_ID
1	1	1	21
2	2	1	21
3	5	1	21
4	1	2	23
5	8	3	24
6	9	3	24
7	2	3	24

## *Istoric\_parole*

```
345  --ISTORIC_PAROLE-----
346  CREATE TABLE istoric_parole
347  (
348      utilizator_id number(5),
349      data_schimbare date,
350      parola_id number(5),
351      constraint pk_istoric primary key(utilizator_id, data_schimbare),
352      constraint fk_parola_istoric foreign key (parola_id) references parole(parola_id)
353  );
354
355  INSERT INTO istoric_parole
356  VALUES (22, TO_DATE('27/12/2023', 'DD/MM/YYYY'), 8);
357
358  INSERT INTO istoric_parole
359  VALUES (22, TO_DATE('01/01/2024', 'DD/MM/YYYY'), 9);
360
361  INSERT INTO istoric_parole
362  VALUES (24, TO_DATE('30/12/2023', 'DD/MM/YYYY'), 10);
363
364  select *
365  from istoric_parole;
```

CREATE TABLE istoric\_parole

(  
    utilizator\_id number(5),  
    data\_schimbare date,  
    parola\_id number(5),  
    constraint pk\_istoric primary key(utilizator\_id, data\_schimbare),  
    constraint fk\_parola\_istoric foreign key (parola\_id) references parole(parola\_id)  
);

INSERT INTO istoric\_parole

VALUES (22, TO\_DATE('27/12/2023', 'DD/MM/YYYY'), 8);

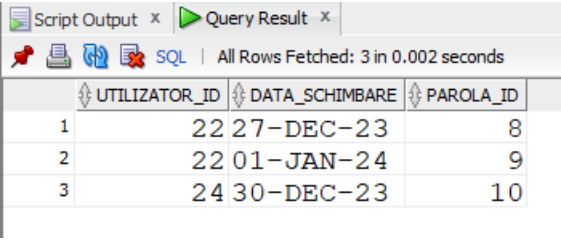
INSERT INTO istoric\_parole

VALUES (22, TO\_DATE('01/01/2024', 'DD/MM/YYYY'), 9);

INSERT INTO istoric\_parole

VALUES (24, TO\_DATE('30/12/2023', 'DD/MM/YYYY'), 10);

select \*  
from istoric\_parole;



	UTILIZATOR_ID	DATA_SCHIMBARE	PAROLA_ID
1	22	27-DEC-23	8
2	22	01-JAN-24	9
3	24	30-DEC-23	10

## 12. SQL

Formulați în limbaj natural și implementați 5 cereri SQL complexe ce vor utiliza, în ansamblul lor, următoarele elemente:

- a) subcereri sincronizate în care intervin cel puțin 3 tabele
- b) subcereri nesincronizate în clauza FROM
- c) grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) în care intervin cel puțin 3 tabele (în cadrul aceleiași cereri)
- d) ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri)
- e) utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE
- f) utilizarea a cel puțin 1 bloc de cerere (clauza WITH)

**Observație:** Într-o cerere se vor regăsi mai multe elemente dintre cele enumerate mai sus, astfel încât cele 5 cereri să le cuprindă pe toate.





## Cerința 1

Să se afișeze utilizatorii care au creat playlisturi, piesele din aceste playlisturi și artistul fiecărei piese, în ordine crescătoare după nume.

```
371 SELECT u.num, u.prenume, pm.playlist_num, p.piesa_num, p.artist_num
372 FROM utilizatori u, playlisturi_muzicale pm, piese p, playlisturi_si_piese pp
373 WHERE u.utilizator_id = pp.utilizator_id
374 and pp.piesa_id = p.piesa_id
375 and pp.playlist_id = pm.playlist_id
376 order by 1;
```

```
SELECT u.num, u.prenume, pm.playlist_num, p.piesa_num, p.artist_num
FROM utilizatori u, playlisturi_muzicale pm, piese p, playlisturi_si_piese pp
WHERE u.utilizator_id = pp.utilizator_id
and pp.piesa_id = p.piesa_id
and pp.playlist_id = pm.playlist_id
order by 1;
```

Script Output x Query Result x

    All Rows Fetched: 7 in 0.003 seconds

NUM	PRENUME	PLAYLIST_NUM	PIESA_NUM	ARTIST_NUM	
1	Georgescu	Mihai	Relaxare	Circles	Post Malone
2	Georgescu	Mihai	Relaxare	Blinding Lights	The Weeknd
3	Georgescu	Mihai	Relaxare	Havana	Camila Cabello
4	Ionescu	Maria	Distractie	Shape of You	Ed Sheeran
5	Popescu	Andrei	Piese mele	Shape of You	Ed Sheeran
6	Popescu	Andrei	Piese mele	Rockstar	Post Malone
7	Popescu	Andrei	Piese mele	Blinding Lights	The Weeknd

## Cerința 2


Să se afișeze utilizatorii care au o parolă cu nivelul de securitate mai mare de 6, numele și prenumele lor, împreună cu numele orașului, strada și parola.

Utilizează funcții pe șiruri de caractere pentru a transforma numele și prenumele în majuscule și pentru a concatena numele și prenumele. Filtrează rezultatele pentru a include doar utilizatorii înregistrați în ultimele 10 luni.

```
381 SELECT UPPER(u.num) num, UPPER(u.prenume) prenume, CONCAT(u.num, CONCAT(' ',u.prenume)) num_complet, l.oras, a.strada,
382 p.parola_num, p.nivel_securitate
383 FROM utilizatori u
384 JOIN parole p ON u.parola_id = p.parola_id
385 JOIN locatii l ON u.locatie_id = l.locatie_id
386 JOIN adrese a ON l.adresa_id = a.adresa_id
387 WHERE p.nivel_securitate > 6
388 AND u.data_inregistrare > ADD_MONTHS(SYSDATE, -10);
```

```
SELECT UPPER(u.num) num, UPPER(u.prenume) prenume,
CONCAT(u.num, CONCAT(' ',u.prenume)) num_complet, l.oras, a.strada,
p.parola_num, p.nivel_securitate
FROM utilizatori u
JOIN parole p ON u.parola_id = p.parola_id
JOIN locatii l ON u.locatie_id = l.locatie_id
JOIN adrese a ON l.adresa_id = a.adresa_id
WHERE p.nivel_securitate > 6
AND u.data_inregistrare > ADD_MONTHS(SYSDATE, -10);
```

Script Output x Query Result x

 All Rows Fetched: 2 in 0.014 seconds

	NUME	PRENUME	NUME_COMPLET	ORAS	STRADA	PAROLA_NUME	NIVEL_SECURITATE
1	IONESCU	MARIA	Ionescu Maria	Bucuresti	Aleea Arena	asdadasdasd	7
2	VASILESCU	ELENA	Vasilescu Elena	Suceava	Aleea bratescu	!AlexAvg 20	10

## Cerința 3

Să utilizatorii care sunt membri ai grupului "La muncă", împreună cu numele albumelor lor și numărul total de fotografii din aceste albume. Utilizează DECODE pentru a afișa un mesaj personalizat dacă sunt una sau mai multe fotografii. Sa se ordoneze rezultatele obtinute dupa coloana numelui utilizatorului.

```
393 = SELECT u.utilizator_id, u.num, u.prenume, a.album_nume, (SELECT COUNT(*)
394                                     FROM fotografii f
395                                     WHERE f.album_id = a.album_id) numar_fotografii,
396                                     DECODE((SELECT COUNT(*)
397                                     FROM fotografii f
398                                     WHERE f.album_id = a.album_id),
399                                     1, 'O singura fotografie',
400                                     'Mai multe fotografii') status_fotografii
401 FROM utilizatori u, utilizatori_si_grupuri ug, grupuri g, albume_fotografii a
402 WHERE u.utilizator_id = ug.utilizator_id
403 and ug.grup_id = g.grup_id
404 and u.utilizator_id = a.utilizator_id
405 and g.denumire = 'La munca'
406 ORDER BY u.num;
```

```
SELECT u.utilizator_id, u.num, u.prenume, a.album_nume, (SELECT COUNT(*)
FROM fotografii f
WHERE f.album_id = a.album_id)
numar_fotografii,
DECODE((SELECT COUNT(*)
FROM fotografii f
WHERE f.album_id = a.album_id),
1, 'O singura fotografie',
'Mai multe fotografii') status_fotografii
FROM utilizatori u, utilizatori_si_grupuri ug, grupuri g, albume_fotografii a
WHERE u.utilizator_id = ug.utilizator_id
and ug.grup_id = g.grup_id
and u.utilizator_id = a.utilizator_id
and g.denumire = 'La munca'
```



ORDER BY u.ume;

	UTILIZATOR_ID	NUME	PRENUME	ALBUM_NUME	NUMAR_FOTOGRAFII	STATUS_FOTOGRAFII
1	24	Georgescu	Mihai	Facultate	1	0 singura fotografie
2	24	Georgescu	Mihai	La mare	1	0 singura fotografie
3	23	Ionescu	Maria	Scoala	2	Mai multe fotografii
4	21	Popescu	Andrei	La mare	3	Mai multe fotografii
5	21	Popescu	Andrei	Ziua mea	1	0 singura fotografie

## Cerința 4

Să se afișeze utilizatorii care sunt membri ai grupului "Școala", împreună cu orașul în care locuiesc. Utilizează un bloc de cerere WITH pentru a obține utilizatorii din grupul "Școala".

```
410 WITH utilizatori_grup_scoala as (  
411     SELECT ug.utilizator_id  
412     FROM utilizatori_si_grupuri ug, grupuri g  
413     WHERE ug.grup_id = g.grup_id  
414           and g.denumire = 'Scoala'  
415 )  
416 SELECT INITCAP(u.ume), INITCAP(u.prenume), l.oras  
417 FROM utilizatori u, locatii l  
418 WHERE u.locatie_id = l.locatie_id  
419 and u.utilizator_id IN (SELECT utilizator_id FROM utilizatori_grup_scoala);
```

```
WITH utilizatori_grup_scoala as (  
    SELECT ug.utilizator_id  
    FROM utilizatori_si_grupuri ug, grupuri g  
    WHERE ug.grup_id = g.grup_id  
          and g.denumire = 'Scoala'  
)  
SELECT INITCAP(u.ume), INITCAP(u.prenume), l.oras  
FROM utilizatori u, locatii l  
WHERE u.locatie_id = l.locatie_id  
and u.utilizator_id IN (SELECT utilizator_id FROM utilizatori_grup_scoala);
```

Script Output x Query Result x		
SQL   All Rows Fetched: 2 in 0.029 seconds		
INITCAP(U.NUME)	INITCAP(U.PRENUME)	ORAS
1 Popescu	Andrei	Bucuresti
2 Dimitrie	Roxana	Atena

## Cerința 5

Să se afișeze orașele și numărul de utilizatori din fiecare oraș. Filtrează rezultatele pentru a avea doar orașe care conțin litera 'u'.

```

423 SELECT l.oras, COUNT(u.utilizator_id) numar_utilizatori
424 FROM utilizatori u, locatii l
425 WHERE u.locatie_id = l.locatie_id
426 GROUP BY l.oras
427 HAVING l.oras like '%u%';

```

SELECT l.oras, COUNT(u.utilizator\_id) numar\_utilizatori

FROM utilizatori u, locatii l

WHERE u.locatie\_id = l.locatie\_id

GROUP BY l.oras

HAVING l.oras like '%u%';

Script Output x Query Result x	
SQL   All Rows Fetched: 2 in 0.006 se	
ORAS	NUMAR_UTILIZATORI
1 Bucuresti	3
2 Suceava	2

## 13. Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri.

### Cerința 1

Șterge toate albumele care nu conțin nicio fotografie.

```
429 --a. Șterge toate albumele care nu conțin nicio fotografie.
430
431 INSERT INTO albume_fotografii
432 VALUES (110, 21, 'Album gol', TO_DATE('02/02/2024', 'DD/MM/YYYY'));
433
434 DELETE FROM albume_fotografii a
435 WHERE a.album_id NOT IN (
436     SELECT f.album_id
437     FROM fotografii f
438 );
```

```
INSERT INTO albume_fotografii
VALUES (110, 21, 'Album gol', TO_DATE('02/02/2024', 'DD/MM/YYYY'));
```

```
DELETE FROM albume_fotografii a
WHERE a.album_id NOT IN (
    SELECT f.album_id
    FROM fotografii f
);
```

### Cerința 2

Actualizează adresa utilizatorilor care au creat cel puțin un album, setând numărul străzii la 99.

```
440 --b. Actualizează adresa utilizatorilor care au creat cel puțin un album, setând numărul străzii la 99.
441 UPDATE adrese a
442 SET a.numar_strada = 99
443 WHERE a.adresa_id IN (
444     SELECT l.adresa_id
445     FROM locatii l
446     JOIN utilizatori u ON l.locatie_id = u.locatie_id
447     JOIN albume_fotografii af ON u.utilizator_id = af.utilizator_id
448 );
```

```

UPDATE adrese a
SET a.numar_strada = 99
WHERE a.adresa_id IN (
    SELECT l.adresa_id
    FROM locatii l
    JOIN utilizatori u ON l.locatie_id = u.locatie_id
    JOIN albume_fotografii af ON u.utilizator_id = af.utilizator_id
);

```

### ***Cerința 3***

Actualizează nivelul de securitate al parolelor la 6 pentru utilizatorii care locuiesc în "Suceava".

```

450 | --c. Actualizează nivelul de securitate al parolelor la 6 pentru utilizatorii care locuiesc în "Suceava".
451 | UPDATE parole p
452 | SET p.nivel_securitate = 6
453 | WHERE p.parola_id IN (
454 |     SELECT u.parola_id
455 |     FROM utilizatori u
456 |     JOIN locatii l ON u.locatie_id = l.locatie_id
457 |     WHERE l.oras = 'Suceava'
458 | );

```

```

UPDATE parole p
SET p.nivel_securitate = 6
WHERE p.parola_id IN (
    SELECT u.parola_id
    FROM utilizatori u
    JOIN locatii l ON u.locatie_id = l.locatie_id
    WHERE l.oras = 'Suceava'
);

```

### 13. Formulați în limbaj natural și implementați în SQL:


A) O cerere ce utilizează operația outer-join pe minimum 4 tabele.

**Cerinta:** Afisati toți utilizatorii, locațiile lor, grupurile din care fac parte și numele albumelor lor, inclusiv utilizatorii care nu au locații, grupuri sau albume asociate.

```
462 SELECT u.ume, u.prenume, u.email, l.oras, l.tara, g.denumire, af.album_ume
463 FROM utilizatori u
464 LEFT JOIN locatii l ON u.locatie_id = l.locatie_id
465 LEFT JOIN utilizatori_si_grupuri ug ON u.utilizator_id = ug.utilizator_id
466 LEFT JOIN grupuri g ON ug.grup_id = g.grup_id
467 LEFT JOIN albume_fotografii af ON u.utilizator_id = af.utilizator_id;
```

```
SELECT u.ume, u.prenume, u.email, l.oras, l.tara, g.denumire, af.album_ume
FROM utilizatori u
LEFT JOIN locatii l ON u.locatie_id = l.locatie_id
LEFT JOIN utilizatori_si_grupuri ug ON u.utilizator_id = ug.utilizator_id
LEFT JOIN grupuri g ON ug.grup_id = g.grup_id
LEFT JOIN albume_fotografii af ON u.utilizator_id = af.utilizator_id;
```

Script Output x Query Result x

 All Rows Fetched: 10 in 0.015 seconds

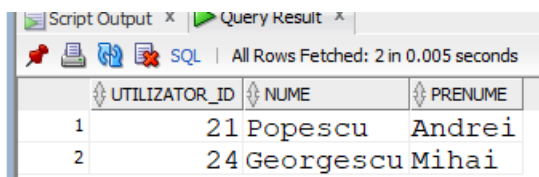
NUME	PRENUME	EMAIL	ORAS	TARA	DENUMIRE	ALBUM_NUME
1	Popescu	Andrei popescu andrei@gmail.com	Bucuresti	Romania	Scoala	La mare
2	Popescu	Andrei popescu andrei@gmail.com	Bucuresti	Romania	La munca	La mare
3	Popescu	Andrei popescu andrei@gmail.com	Bucuresti	Romania	Scoala	Ziua mea
4	Popescu	Andrei popescu andrei@gmail.com	Bucuresti	Romania	La munca	Ziua mea
5	Ionescu	Maria ionescu maria@gmail.com	Bucuresti	Romania	La munca	Scoala
6	Georgescu	Mihai georgescu mihai@gmail.com	Suceava	Romania	La munca	Facultate
7	Georgescu	Mihai georgescu mihai@gmail.com	Suceava	Romania	La munca	La mare
8	Ionescu	Elena ionescu elena@gmail.com	Bucuresti	Romania	La munca	(null)
9	Dimitrie	Roxana dimitrie roxx@gmail.com	Atena	Grecia	Scoala	(null)
10	Vasilescu	Elena vasilescu elena@gmail.com	Suceava	Romania	La munca	(null)

## B) O cerere ce utilizează operația division

**Cerinta: Găsiți utilizatorii care au adăugat toate piesele artistului 'The Weeknd' în oricare dintre playlisturile lor.**

```
472 SELECT u.utilizator_id, u.num, u.prenume
473 FROM utilizatori u
474 WHERE NOT EXISTS (
475     SELECT p.piesa_id
476     FROM piese p
477     WHERE p.artist_nume = 'The Weeknd'
478     MINUS
479     SELECT pp.piesa_id
480     FROM playlisturi_si_piese pp
481     WHERE pp.utilizator_id = u.utilizator_id
482 );
```

```
SELECT u.utilizator_id, u.num, u.prenume
FROM utilizatori u
WHERE NOT EXISTS (
    SELECT p.piesa_id
    FROM piese p
    WHERE p.artist_nume = 'The Weeknd'
    MINUS
    SELECT pp.piesa_id
    FROM playlisturi_si_piese pp
    WHERE pp.utilizator_id = u.utilizator_id
```



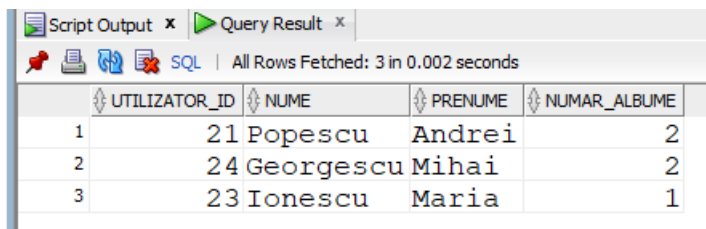
The screenshot shows a database interface with a 'Query Result' tab. It displays the results of the SQL query, showing two rows of data. The columns are UTILIZATOR\_ID, NUM, and PRENUME. The first row shows user 21, Popescu Andrei, and the second row shows user 24, Georgescu Mihai.

	UTILIZATOR_ID	NUM	PRENUME
1	21	Popescu	Andrei
2	24	Georgescu	Mihai

**C) O cerere care implementează analiza top-n.**  
**Cerinta: Sa se afiseze top 3 utilizatori care au creat cele mai multe albume.**

```
486
487 SELECT *
488 FROM (
489     SELECT u.utilizator_id, u.numa, u.prenume, COUNT(af.album_id) numar_albume
490     FROM utilizatori u
491     JOIN albume_fotografii af ON u.utilizator_id = af.utilizator_id
492     GROUP BY u.utilizator_id, u.numa, u.prenume
493     ORDER BY numar_albume DESC
494 )
495 WHERE ROWNUM <= 3;
496
```

```
SELECT *
FROM (
    SELECT u.utilizator_id, u.numa, u.prenume, COUNT(af.album_id)
numar_albume
    FROM utilizatori u
    JOIN albume_fotografii af ON u.utilizator_id = af.utilizator_id
    GROUP BY u.utilizator_id, u.numa, u.prenume
    ORDER BY numar_albume DESC
)
WHERE ROWNUM <= 3;
```



The screenshot shows a database interface with a 'Query Result' tab. It displays the results of the SQL query, showing 3 rows of data. The columns are UTILIZATOR\_ID, NUMA, PRENUME, and NUMAR\_ALBUME. The data is as follows:

	UTILIZATOR_ID	NUMA	PRENUME	NUMAR_ALBUME
1	21	Popescu	Andrei	2
2	24	Georgescu	Mihai	2
3	23	Ionescu	Maria	1