**Proiect Baze de Date – gestionare date pentru o aplicatie de muzica tip Spotify**

**Curecheriu Elena**

**grupa 142**

**Facultatea de Matematica si Informatica,**

**Universitatea din Bucuresti**

**Cuprins:**

1. **Descrierea modelului real, a utilității acestuia și a regulilor de funcționare. – pagina 3**
2. **Prezentarea constrângerilor (restricții, reguli) impuse asupra modelului. – pagina 4**
3. **Descrierea entităților, incluzând precizarea cheii primare. – pagina 6**
4. **Descrierea relațiilor, incluzând precizarea cardinalității acestora. – pagina 6**
5. **Descrierea atributelor, incluzând tipul de date și eventualele constrângeri, valori implicite, valori posibile ale atributelor. – pagina 9**
6. **Realizarea diagramei entitate-relație corespunzătoare descrierii de la punctele 3-5. – pagina 11**
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. – pagina 12**
8. **Enumerarea schemelor relaționale corespunzătoare diagramei conceptuale proiectate la punctul 7. – pagina 13**
9. **Realizarea normalizării până la forma normală 3 (FN1-FN3). – pagina 13**
10. **Crearea unei secvențe ce va fi utilizată în inserarea înregistrărilor în tabele (punctul 11). – pagina 16**
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). – pagina 18**
12. **Formulați în limbaj natural și implementați 5 cereri SQL complexe - pagina 41**
13. **Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri. – pagina 46**
14. **Descrierea modelului real, a utilitatii acestuia si a regulilor de functionare**

Tema:

Tema proiectului reprezinta o baza de date realizata pentru o aplicatie mobila de streaming de muzica, similara cu Spotify, Apple Music sau Youtube Music, avand scopul de a gestiona playlist-urile utilizatorilor si activitatea acestora.

Descrierea modelului real:

Aceasta baza de date poate fi utilizata pentru a gestiona in mod eficient continutul muzical pe care il acceseaza utilizatorii, acesta putand fi organizat in playlist-uri.

Entitatea ABONAMENT contine date despre tipul abonamentului (standard, student, etc.), pretul acestuia, data inceperii abonamentului si metoda prin care se efectueaza plata acestuia. Entitatea UTILIZATOR stocheaza informatii despre un utilizator, anume nume, adresa de email, data inregistrarii pe platforma si tara de provenienta. Entitatea PLAYLIST este legata de entitatea mentionata anterior, putand fi creata de catre un utilizator, si contine detalii despre un playlist: un identificator unic, numele playlist-ului si data crearii. De asemenea, entitatea PLAYLIST este legata de entitatea MELODIE, caracterizata de identificatorul unic, titlu, durata, identificatorul artistului, identificatorului albumului de care apartine, identificatorul genului muzical si data lansarii. Entitatea MELODIE este legata la randul ei de entitatile ALBUM, ARTIST si GEN, iar in final, entitatea ARTIST este legata de entitatea CASA\_DISCURI.

Utilitatea:

Utilitatea acestei baze de date se remarca prin gestionarea continutului muzical accesat de catre utilizatori intr-un mod eficient, prin posibilitatea extinderii acesteia pentru a gestiona un numar mare de utilizatori si continut fara a compromite performanta, prin permiterea personalizarii experientei utilizatorului prin recomandari bazate pe istoricul redarilor, activitatea anterioara si preferintele utilizatorului, dar si prin posibilitatea realizarii statisticilor si analizelor privind comportamentul utilizatorilor aplicatiei si popularitatea pieselor si artistilor.

Reguli de functionare:

Un utilizator poate avea un singur abonament pe platforma. Aplicatia poate avea mai multi utilizatori care gestioneaza playlist-uri. Aceste playlist-uri contin una sau mai multe melodii, care la randul lor sunt detinute de unul sau mai multi artisti, se incadreaza intr-un gen muzical si pot apartine unui album, mai multor albume sau niciunui album, putand fi lansat sub forma unui single. Albumele apartin unui singur artist sau mai multor artisti, iar fiecare artist este reprezentat de catre o singura casa de discuri.

1. **Prezentarea constrangerilor (restrictii, reguli) impuse asupra modelului**

Regulile si constrangerile din baza de date descrisa sunt:

1. Tabela UTILIZATOR:

* id\_utilizator este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* id\_abonament este o cheie externa catre tabela ABONAMENT si trebuie sa se refere la un id\_abonament existent in acea tabela
* nume trebuie sa aiba o lungime maxima de 255 de caractere
* email trebuie sa aiba o lungime maxima de 255 de caractere
* data\_inregistrare trebuie sa fie o valoare de tipul data
* tara trebuie sa aiba o lungime maxima de 2 caractere

1. Tabela ABONAMENT:

* id\_abonament este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* tip\_abonament trebuie sa aiba o lungime maxima de 255 de caractere
* pret trebuie sa fie o valoare reala finita pozitiva
* data\_incepere trebuie sa fie o valoare de tip data
* metoda \_plata trebuie sa aiba o lungime maxima de 255 de caractere

1. Tabela PLAYLIST

* id\_playlist este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* id\_utilizator este o cheie externa catre tabela UTILIZATOR si trebuie sa se refere la un id\_utilizator existent in acea tabela
* nume trebuie sa aiba o lungime maxima de 255 de caractere
* data\_creare trebuie sa fie o valoare de tipul data

1. Tabela MELODIE\_PLAYLIST:

* Are doua coloane cheie care fac referire la cheile primare ale tabelelor “PLAYLIST” si “MELODIE”, respective “id\_playlist” si “id\_melodie”. Aceste doua coloane formeaza impreuna cheia primara a tabelei (care trebuie sa fie unica)

1. Tabela MELODIE:

* id\_melodie este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* titlu trebuie sa aiba o lungime maxima de 255 de caractere
* durata trebuie sa fie o valoare intreaga pozitiva
* data\_lansare trebuie sa fie o valoare de tipul data
* id\_album este o cheie externa catre tabela ALBUM si trebuie sa se refere la un id\_album existent in acea tabela

1. Tabela GEN:

* id\_gen este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* nume trebuie sa aiba o lungime maxima de 255 de caractere

1. Tabela GEN\_MELODIE:

* Are doua coloane cheie care fac referire la cheile primare ale tabelelor “GEN” si “MELODIE”, respective “id\_gen” si “id\_melodie”. Aceste doua coloabe formeaza impreuna cheia primara a tabelei (care trebuie sa fie unica)

1. Tabela ALBUM:

* id\_album este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* titlu trebuie sa aiba o lungime maxima de 255 de caractere
* data\_lansare trebuie sa fie o valoare de tip data
* durata trebuie sa fie o valoare intreaga pozitiva

1. Tabela ARTIST:

* id\_artist este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* nume trebuie sa aiba o lungime maxima de 255 de caractere
* an\_debut trebuie sa fie o valoare numerica intreaga pozitiva de 4 cifre
* id\_casa\_discuri este o cheie externa catre tabela CASA\_DISCURI si trebuie sa se refere la un id\_casa\_discuri existent in acea tabela

1. Tabela CASA\_DISCURI:

* id\_casa\_discuri este cheia primara si trebuie sa fie unica pentru fiecare inregistrare din table
* nume trebuie sa aiba o lungime maxima de 255 de caractere
* nume\_detinator trebuie sa aiba o lungime maxima de 255 de caractere
* prenume\_detinator trebuie sa aiba o lungime maxima de 255 de caractere
* an\_infiintare trebuie sa fie o valoare numerica intreaga pozitiva de 4 cifre

1. Tabela GEN\_ALBUM:

* Are doua coloane cheie care fac referire la cheile primare ale tabelelor “GEN” si “ALBUM”, respective “id\_gen” si “id\_album”. Aceste doua coloabe formeaza impreuna cheia primara a tabelei (care trebuie sa fie unica)

1. Tabela ARTIST\_MELODIE:

* Are doua coloane cheie care fac referire la cheile primare ale tabelelor “MELODIE” si “ARTIST”, respective “id\_melodie” si “id\_artist”. Aceste doua coloabe formeaza impreuna cheia primara a tabelei (care trebuie sa fie unica)

Descrierea restrictiilor in limbaj natural:

* O melodie poate sa apartina cel mult unui singur album. Daca aceasta apartine unui album, data lansarii acesteia coincide cu data lansarii albumului, asadar putem memora data lansarii melodiei cu NULL.
* O melodie poate fi interpretata de mai multi artisti.
* Un playlist poate fi gestionat de catre un singur utilizator.
* Un utilizator poate avea o melodie o singura data intr-un playlist.
* Un utilizator are un singur abonament.
* Un abonament poate fi utilizat de catre un singur utilizator.
* Un artist poate fi reprezentat de catre o singura casa de discuri.
* O melodie se incadreaza in cel putin un gen muzical.
* Un album se incadreaza in cel putin un gen muzical.
* Un album poate fi lansat de catre un singur artist.

1. **Descrierea entitatilor, incluzand precizarea cheii primare**
2. Entitatea UTILIZATOR: persoana care utilizeaza aplicatia; cheia primara este id\_utilizator
3. Entitatea ABONAMENT: abonamentul pe care il detine fiecare utilizator, permitandu-i sa acceseze continutul media al aplicatiei; cheia primara este id\_abonament
4. Entitatea PLAYLIST: mod de grupare a melodiilor, gestionat de catre utilizator; cheia primara este id\_playlist
5. Entitatea MELODIE: elementul de baza al aplicatiei care alcatuieste playlist-urile si albumele si apartin artistilor; cheia primara este id\_melodie
6. Entitatea GEN: gen muzical de care apartine fiecare melodie; cheia primara este id\_gen
7. Entitatea ALBUM: mod de grupare a melodiilor, specific unui artist; cheia primara este id\_album
8. Entitatea ARTIST: persoana care interpreteaza melodiile si detine albume; cheia primara este id\_artist
9. Entitatea CASA\_DISCURI: casa de discuri de care apartine fiecare artist; cheia primara este id\_casa\_discuri
10. Entitatea MELODIE\_PLAYLIST: tabela intermediara intr-o relatie many-to-many intre tabelele “PLAYLIST” si “MELODIE”; aceasta contine doua coloane cheie, id\_playlist si id\_melodie, referindu-se la cheile primare ale tabelelor “PLAYLIST” si “MELODIE”, acestea formand impreuna cheia primara a tabelei
11. Entitatea GEN\_MELODIE: tabela intermediara intr-o relatie many-to-many intre tabelele “GEN” si “MELODIE”; aceasta contine doua coloane cheie, id\_melodie si id\_gen, acestea formand impreuna cheia primara a tabelei
12. Entitatea GEN\_ALBUM: tabela intermediara intr-o relatie many-to-many intre tabelele “GEN” si “ALBUM”; aceasta contine doua coloane cheie, id\_album si id\_gen, acestea formand impreuna cheia primara a tabelei
13. Entitatea ARTIST\_MELODIE: tabela intermediara intr-o relatie many-to-many intre tabelele “ARTIST” si “MELODIE”; aceasta contine doua coloane cheie, id\_artist si id\_melodie, acestea formand impreuna cheia primara a tabelei
14. **Descrierea relatiilor, incluzand precizarea cardinalitatii acestora**
15. **Tabela UTILIZATOR:**

* relatie de tip “one-to-many” cu tabela PLAYLIST, deoarece un utilizator poate gestiona mai multe playlist-uri, dar un playlist apartine unui singur utilizator; cardinalitatea este 1:M(0)
* relatie de tip “one-to-one” cu tabela ABONAMENT, deoarece un utilizator poate avea un singur tip de abonament, iar un abonament apartine unui singur utilizator; cardinalitatea este 1:1

1. **Tabela ABONAMENT:**

* relatie de tip “one-to-one” cu tabela UTILIZATOR (descrisa anterior); cardinalitatea este 1:1

1. **Tabela PLAYLIST :**

* relatie de tip “many-to-many” cu tabela MELODIE, iar intre aceste entitati se afla tabela MELODIE\_PLAYLIST, fiind o tabela intermediara
* asadar, relatie de tip “one-to-many” cu tabela MELODIE\_PLAYLIST;
* relatie de tip “many-to-one” cu tabela UTILIZATOR (descrisa anterior); cardinalitatea este M(0):1
* cardinalitatea relatiei PLAYLIST – MELODIE este M(0):M(1) (un playlist trebuie sa aiba minimum o melodie, iar o melodie nu trebuie sa se afle neaparat intr-un playlist)

1. **Tabela MELODIE**:

* relatie de tip “one-to-many” cu tabela MELODIE\_PLAYLIST, cea din urma fiind o tabela intermediara pentru a rezolva relatia de tip “many-to-many” dintre tabela MELODIE si tabela PLAYLIST
* relatie de tip “many-to-one” cu tabela GEN\_MELODIE, o tabela intermediara pentru a rezolva relatia de tip “many-to-many” cu tabela GEN (un gen muzical cuprinde mai multe melodii, iar o melodie se poate incadra in mai multe genuri muzicale)
* relatie de tip “many-to-one” cu tabela ALBUM, intrucat un album are mai multe melodii, dar o melodie apartine unui singur album; cardinalitatea este M(1):1(0) (un album trebuie sa contina minimum o melodie, iar o melodie se afla in cel mult un album)
* relatie de tip “many-to-many” cu tabela ARTIST, deoarece un artist poate avea mai multe melodii, iar o melodie poate fi interpretata de mai multi artisti; cardinalitatea este M(1):M(1)
* relatie de tip “one-to-many” cu tabela ARTIST\_MELODIE, cea din urma fiind o tabela intermediara pentru a rezolva relatia de tip “many-to-many” dintre tabela MELODIE si tabela ARTIST;
* relatia MELODIE – GEN are cardinalitate M(0):M(1)
* relatia MELODIE – PLAYLIST are cardinalitate M(1):M(0)

1. **Tabela MELODIE\_PLAYLIST:**

* relatie de tipul “many-to-one” cu tabela MELODIE, dar si cu tabela PLAYLIST, fiind o tabela intermediara in relatia “many-to-many” dintre PLAYLIST si MELODIE

1. **Tabela ARTIST\_MELODIE:**

* relatie de tipul “many-to-one” cu tabela MELODIE, dar si cu tabela ARTIST, fiind o tabela intermediara in relatia “many-to-many” dintre ARTIST si MELODIE

1. **Tabela GEN\_MELODIE:**

* relatie de tipul “many-to-one” cu tabela MELODIE, dar si cu tabela GEN, fiind o tabela intermediara in relatia “many-to-many” dintre GEN si MELODIE

1. **Tabela ALBUM:**

* relatie de tipul “one-to-many” cu tabela MELODIE (relatie descrisa anterior);
* relatie de tipul “many-to-one” cu tabela ARTIST, deoarece un artist poate avea mai multe albume, dar un album apartine unui singur artist (conform restrictiilor impuse modelului la un punct anterior); relatia are cardinalitate M(1):1
* relatie de tipul “one-to-many” cu tabela GEN\_ALBUM, o tabela intermediara pentru a rezolva relatia de tip “many-to-many” cu tabela GEN (un gen muzical cuprinde mai multe albume, iar un album se poate incadra in mai multe genuri muzicale)
* relatia ALBUM – GEN are cardinalitate M(0):M(1)

1. **Tabela GEN:**

* relatie de tipul “one-to-many” cu tabela GEN\_MELODIE, dar si cu tabela GEN\_ALBUM, cele doua fiind tabelele intermediare din relatiile “many-to-many” cu MELODIE si ALBUM;
* relatie de tipul “many-to-many” cu tabela ALBUM (descrisa anterior)
* relatie de tipul “many-to-many” cu tabela MELODIE (descrisa anterior)

1. **Tabela GEN\_ALBUM:**

* relatie de tipul “many-to-one” cu tabela ALBUM, dar si cu tabela GEN, fiind o tabela intermediara in relatia “many-to-many” dintre GEN si ALBUM;

1. **Tabela ARTIST:**

* relatie de tipul “many-to-many” cu tabela MELODIE (descrisa anterior)
* relatie de tipul “one-to-many” cu tabela ALBUM (descrisa anterior)
* relatie “many-to-one” cu tabela CASA\_DISCURI, deoarece o casa de discuri poate reprezenta mai multi artisti, dar un artist apartine si are contract cu o singura casa de discuri; are cardinalitate M(0):1

1. **Tabela CASA\_DISCURI:**

* relatie de tipul “one-to-many” cu tabela ARTIST (descrisa anterior)

1. **Descrierea atributelor, incluzand tipul de date si eventualele constrangeri, valori implicite, valori posibile ale atributelor**
2. **Tabela UTILIZATOR:**

* id\_utilizator: identificator unic de tip INT, cheie primara
* id\_abonament: cheie straina care face referire la tabela ABONAMENT
* nume: numele de utilizator, de tip VARCHAR(255)
* email: email-ul utilizatorului
* data\_inregistrare: data la care si-a facut utilizatorul contul, de tip DATE
* tara: tara de provenienta a utilizatorului

1. **Tabela ABONAMENT:**

* id\_abonament: identificator unic de tip INT, cheie primara
* tip\_abonament: tipul abonamentului, de tip VARCHAR(255), poate avea una dintre valorile ‘student’, ‘standard’ si ‘business’
* pret: pretul abonamentului, de tip DECIMAL(10,2)
* data\_incepere: data inceperii abonamentului, de tip DATE
* metoda\_plata: metoda de plata si de facturare a abonamentului, de tip VARCHAR(255), poate avea una dintre valorile ‘paypal’, ‘visa’ si ‘gift card’

1. **Tabela PLAYLIST:**

* id\_playlist: identificator unic de tip INT, cheie primara
* id\_utilizator: cheie straina care face referire la tabela UTILIZATOR
* nume: numele playlistului, de tip VARCHAR(255)
* data\_creare: data crearii playlistului de catre utilizator, de tip DATE

1. **Tabela MELODIE\_PLAYLIST:**

* (id\_melodie, id\_playlist): cele doua coloane formeaza cheia primara a tabelei

1. **Tabela MELODIE:**

* id\_melodie: identificator unic de tip INT, cheie primara
* durata: durata melodiei exprimata in secunde, de tip INT
* titlu: titlul melodiei, de tip VARCHAR(255)
* data\_lansare: data lansarii melodiei, de tip VARCHAR(255); daca apartine unui album, atunci aceasta poate fi NULL

1. **Tabela GEN:**

* id\_gen: identificator unic de tip INT, cheie primara
* nume: numele genului, de tip VARCHAR(255)

1. **Tabela GEN\_MELODIE:**

* (id\_gen, id\_melodie): cele doua coloane formeaza cheia primara a tabelei

1. **Tabela GEN\_ALBUM:**

* (id\_gen, id\_album): cele doua coloane formeaza cheia primara a tabelei

1. **Tabela ALBUM:**

* id\_album: identificator unic de tip INT, cheie primara
* titlu: titlul albumului, de tip VARCHAR(255)
* data\_lansare: data lansarii albumului, de tip DATE
* durata: durata albumului exprimata in secunde, de tip INT

1. **Tabela ARTIST:**

* id\_artist: identificator unic de tip INT, cheie primara
* id\_casa\_discuri: cheie straina care face referire la tabela CASA\_DISCURI
* nume: numele de scena al artistului, de tip VARCHAR(255)
* an\_debut: anul de debut al artistului, de tip INT

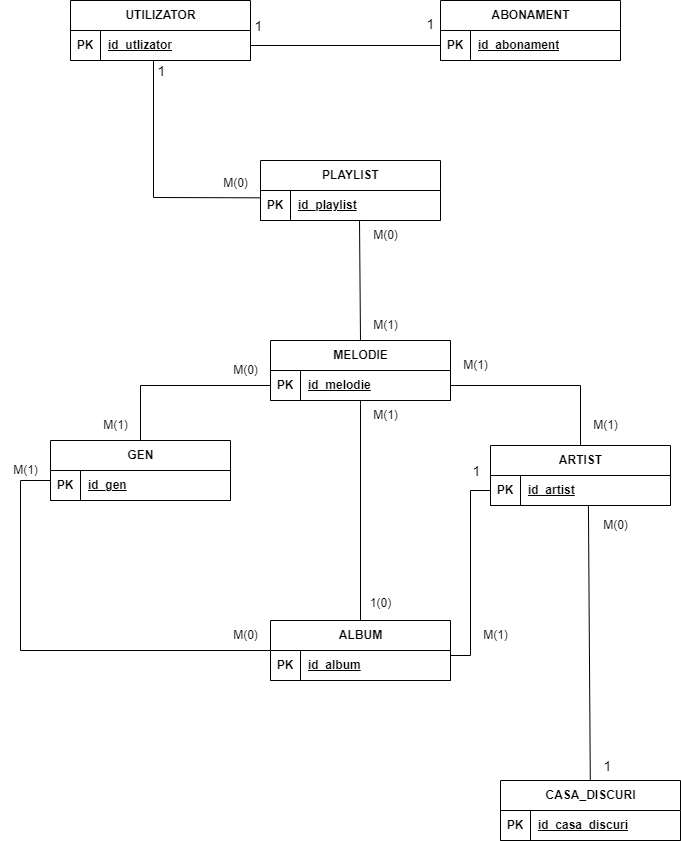
1. **Tabela ARTIST\_MELODIE:**

* (id\_artist, id\_melodie): cele doua coloane formeaza cheia primara a tabelei

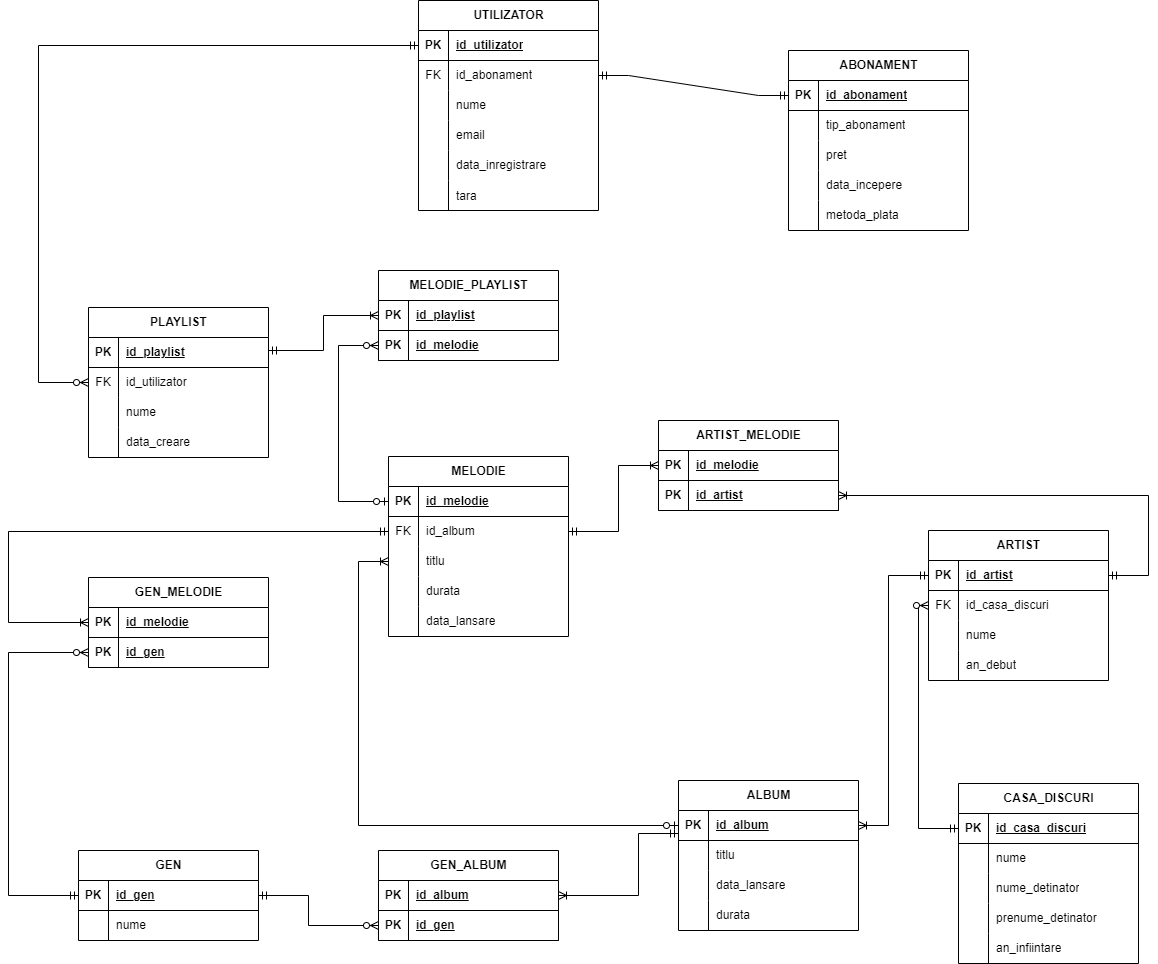
1. **Tabela CASA\_DISCURI:**

* id\_casa\_discuri: identificator unic de tip INT, cheie primara
* nume: numele casei de discuri, de tip VARCHAR(255)
* nume\_detinator: numele de familie al detinatorului casei de discuri, de tip VARCHAR(255)
* prenume\_detinator: prenumele detinatorului casei de discuri, de tip VARCHAR(255)
* an\_infiintare: anul infiintarii casei de discuri, de tip INT

1. **Realizarea diagramei entitate-relatie corespunzatoare descrierii de la punctele 3-5**



1. **Realizarea diagramei conceptuale corespunzatoare diagramei entitate-relatie proiectate la punctul 6. Diagrama conceptuala obtinuta trebuie sa contina minimum 6 tabele (fara considerarea subentitatilor), dintre care cel putin un tabel asociativ**



1. **Enumerarea schemelor relationale corespunzatoare diagramei conceptuale proiectate la punctual 7**

**Schemele relationale:**

* UTILIZATOR(id\_utilizator, id\_abonament, nume, email, data\_inregistrare, tara)
* ABONAMENT(id\_abonament, tip\_abonament, pret, data\_incepere, metoda\_plata)
* PLAYLIST(id\_playlist, id\_utilizator, nume, data\_creare)
* MELODIE\_PLAYLIST(id\_playlist, id\_melodie)
* MELODIE(id\_melodie, id\_artist, titlu, durata, data\_lansare)
* GEN(id\_gen, nume)
* GEN\_MELODIE(id\_gen, id\_melodie)
* GEN\_ALBUM(id\_gen, id\_album)
* ALBUM(id\_album, titlu, durata, data\_lansare)
* ARTIST(id\_artist, id\_casa\_discuri, nume, an\_debut)
* CASA\_DISCURI(id\_casa\_discuri, nume, nume\_detinator, prenume\_detinator, an\_infiintare)

1. **Realizarea normalizarii pana la forma normala 3 (FN1-FN3)**
2. **Exemplu NON-FN1: Tabela MELODIE**

O relatie se afla in FN1 daca fiecarui atribut care o compune ii corespunde o valoare care nu se poate descompune, adica trebuie sa obtinem atribute atomice. Un exemplu care nu se incadreaza in FN1 este tabelul de mai jos, in care observam faptul ca in id\_artist avem multiple valori, asadar facem cate o inserare pentru fiecare valoare posibila a lui id\_artist pentru a obtine FN1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MELODIE | | | | |
| id\_melodie | id\_artist | titlu | durata | data\_lansare |
| 1 | 1 | Vampire | 219 | 30/06/2023 |
| 2 | 2, 4, 5 | Don’t Call Me Angel | 190 | 13/09/2019 |
| 3 | 3 | Sofia | 188 | 26/07/2019 |
| 4 | 5, 6 | Die For You - Remix | 232 | 24/02/2023 |

Transformare in FN1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MELODIE | | | | |
| id\_melodie | id\_artist | titlu | durata | data\_lansare |
| 1 | 1 | Vampire | 219 | 30/06/2023 |
| 2 | 2 | Don’t Call Me Angel | 190 | 13/09/2019 |
| 3 | 3 | Sofia | 188 | 26/07/2019 |
| 4 | 5 | Die For You - Remix | 232 | 24/02/2023 |
| 5 | 4 | Don’t Call Me Angel | 190 | 13/09/2019 |
| 6 | 5 | Don’t Call Me Angel | 190 | 13/09/2019 |
| 7 | 6 | Die For You - Remix | 232 | 24/02/2023 |

1. **Exemplu NON-FN2: Tabela MELODIE**

O relatie se afla in FN2 daca si numai daca aceasta este deja in FN1, iar fiecare atribut care nu este cheie primara este dependent de intreaga cheie primara. Astfel, știm că FN2 presupune să nu existe dependențe funcționale parțiale în cadrul relației.

La tabelul utilizat mai sus, am modificat id\_artist pentru claritate si am adaugat un nou atribut, “an\_debut\_artist”, care nu depinde de cheia primara a entitatii MELODIE, astfel cream o noua entitate ARTIST care cuprinde atributa id\_artist si atributa an\_debut\_artist, astfel realizandu-se FN2.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | MELODIE | | | | |
| id\_melodie | id\_artist | | titlu | durata | data\_lansare | an\_debut\_artist |
| 1 | Olivia Rodrigo | | Vampire | 219 | 30/06/2023 | 2021 |
| 2 | Lana del Rey | | Don’t Call Me Angel | 190 | 13/09/2019 | 2012 |
| 3 | Clairo | | Sofia | 188 | 26/07/2019 | 2018 |
| 4 | Ariana Grande | | Die For You - Remix | 232 | 24/02/2023 | 2013 |
| 5 | Miley Cyrus | | Don’t Call Me Angel | 190 | 13/09/2019 | 2007 |
| 6 | Ariana Grande | | Don’t Call Me Angel | 190 | 13/09/2019 | 2013 |
| 7 | The Weeknd | | Die For You - Remix | 232 | 24/02/2023 | 2010 |

Transformare in FN2:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MELODIE | | | | |
| id\_melodie | id\_artist | titlu | durata | data\_lansare |
| 1 | 1 | Vampire | 219 | 30/06/2023 |
| 2 | 2 | Don’t Call Me Angel | 190 | 13/09/2019 |
| 3 | 3 | Sofia | 188 | 26/07/2019 |
| 4 | 5 | Die For You - Remix | 232 | 24/02/2023 |
| 5 | 4 | Don’t Call Me Angel | 190 | 13/09/2019 |
| 6 | 5 | Don’t Call Me Angel | 190 | 13/09/2019 |
| 7 | 6 | Die For You - Remix | 232 | 24/02/2023 |

|  |  |
| --- | --- |
| ARTIST | |
| id\_artist | an\_debut\_artist |
| 1 | 2021 |
| 2 | 2012 |
| 3 | 2018 |
| 4 | 2007 |
| 5 | 2013 |
| 6 | 2010 |

1. **Exemplu NON-FN3:**

O relație se află în FN3 dacă și numai dacă această relație este deja in FN2 (și implicit în FN1), iar fiecare atribut care nu este cheie primară este dependent de cheia primară. Asadar, nu trebuie sa avem dependente tranzitive. In tabelul de mai jos, id\_melodie, id\_album si titlu\_album au o dependenta tranzitiva, titlu\_album fiind derivat din id\_album.

Se creeaza tabela ALBUM pentru a cuprinde atributele id\_album si titlu\_album si pentru a se realiza FN3.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MELODIE | | | | |
| id\_melodie | titlu\_melodie | durata | data\_lansare | titlu\_album |
| 1 | the boy is mine | 174 | 08/03/2024 | eternal sunshine |
| 2 | Come back to me | 388 | 24/05/2024 | Right Place, Wrong People |
| 3 | Hype Boy | 179 | 01/08/2022 | New Jeans |
| 4 | bye | 164 | 08/03/2024 | eternal sunshine |
| 5 | LOST! | 233 | 24/05/2024 | Right Place, Wrong People |
| 6 | Lolita | 219 | 30/01/2012 | Born To Die |
| 7 | supernatural | 163 | 08/03/2024 | eternal sunshine |
| 8 | Attention | 180 | 01/08/2022 | New Jeans |

Transformare in FN3:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | MELODIE | | | |
| id\_album | id\_melodie | titlu\_melodie | durata | data\_lansare |
| 1 | 1 | the boy is mine | 174 | 08/03/2024 |
| 2 | 2 | Come back to me | 388 | 24/05/2024 |
| 3 | 3 | Hype Boy | 179 | 01/08/2022 |
| 1 | 4 | bye | 164 | 08/03/2024 |
| 2 | 5 | LOST! | 233 | 24/05/2024 |
| 4 | 6 | Lolita | 219 | 30/01/2012 |
| 1 | 7 | supernatural | 163 | 08/03/2024 |
| 3 | 8 | Attention | 180 | 01/08/2022 |

|  |  |
| --- | --- |
| ALBUM | |
| id\_album | titlu\_album |
| 1 | eternal sunshine |
| 2 | Right Place, Wrong People |
| 3 | New Jeans |
| 4 | eternal sunshine |

1. **Crearea unei secvente ce va fi utilizata in inserarea inregistrarilor in tabele (punctul 11)**

CREATE SEQUENCE UTILIZATOR\_SEQ START WITH 1;

CREATE SEQUENCE ABONAMENT\_SEQ START WITH 1;

CREATE SEQUENCE PLAYLIST\_SEQ START WITH 1;

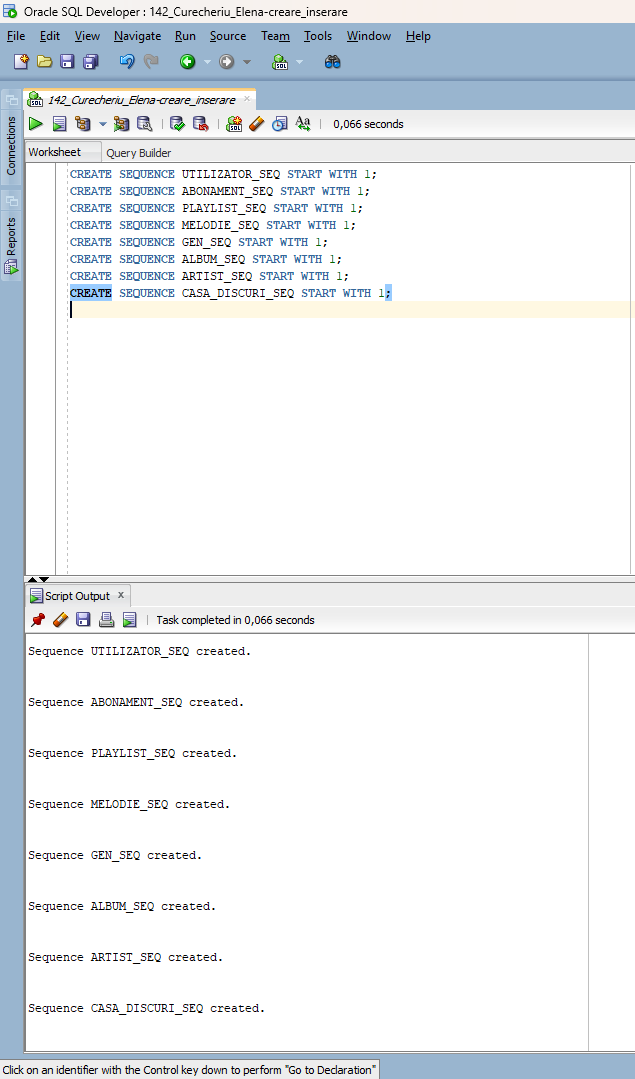
CREATE SEQUENCE MELODIE\_SEQ START WITH 1;

CREATE SEQUENCE GEN\_SEQ START WITH 1;

CREATE SEQUENCE ALBUM\_SEQ START WITH 1;

CREATE SEQUENCE ARTIST\_SEQ START WITH 1;

CREATE SEQUENCE CASA\_DISCURI\_SEQ START WITH 1;

****

1. **Crearea tabelelor in SQL si inserarea de date coerente in fiecare dintre acestea**
2. **Tabela ABONAMENT**

--ABONAMENT

CREATE TABLE ABONAMENT(

id\_abonament INT DEFAULT ABONAMENT\_SEQ.NEXTVAL PRIMARY KEY,

tip\_abonament VARCHAR(100),

CONSTRAINT tip\_abonament\_check CHECK (tip\_abonament IN ('student', 'standard', 'business')),

pret DECIMAL(10,2),

data\_incepere DATE,

metoda\_plata VARCHAR(100),

CONSTRAINT metoda\_plata\_check CHECK (metoda\_plata IN ('paypal', 'visa', 'gift card'))

);

INSERT INTO ABONAMENT (tip\_abonament, pret, data\_incepere, metoda\_plata) VALUES ('standard', 5.99, '12-SEP-2023', 'visa');

INSERT INTO ABONAMENT (tip\_abonament, pret, data\_incepere, metoda\_plata) VALUES ('student', 2.99, '03-NOV-2015', 'paypal');

INSERT INTO ABONAMENT (tip\_abonament, pret, data\_incepere, metoda\_plata) VALUES ('business', 16.20, '14-JUN-2020', 'visa');

INSERT INTO ABONAMENT (tip\_abonament, pret, data\_incepere, metoda\_plata) VALUES ('standard', 5.99, '09-OCT-2019', 'gift card');

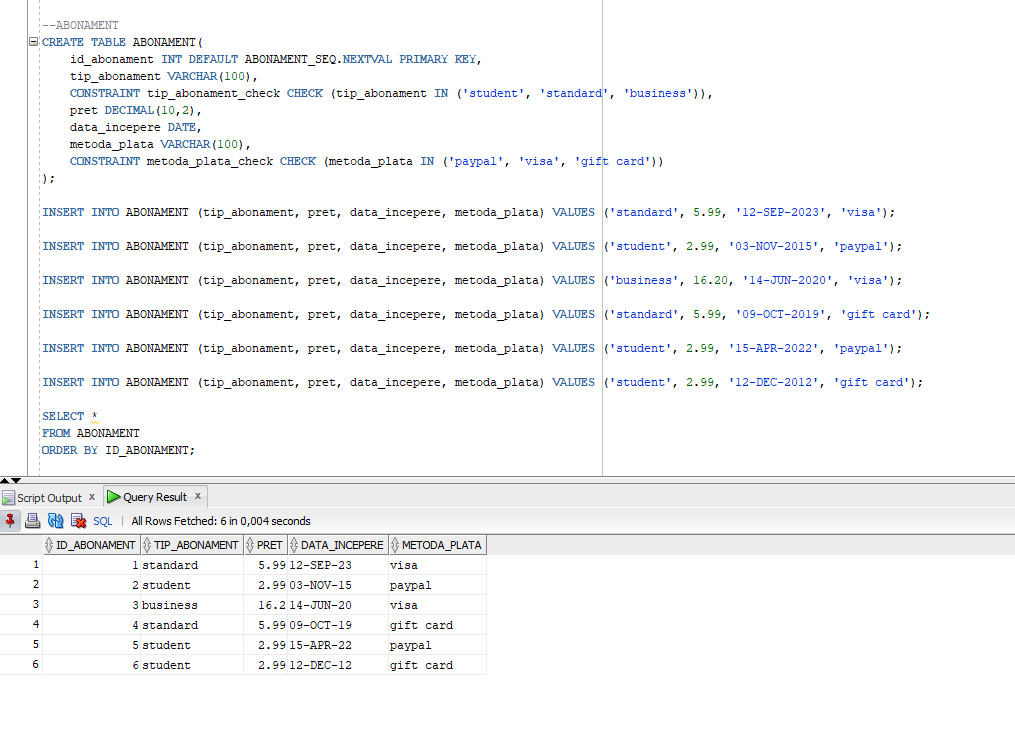
INSERT INTO ABONAMENT (tip\_abonament, pret, data\_incepere, metoda\_plata) VALUES ('student', 2.99, '15-APR-2022', 'paypal');

INSERT INTO ABONAMENT (tip\_abonament, pret, data\_incepere, metoda\_plata) VALUES ('student', 2.99, '12-DEC-2012', 'gift card');

SELECT \*

FROM ABONAMENT

ORDER BY ID\_ABONAMENT;



1. **Tabela UTILIZATOR**

--UTILIZATOR

CREATE TABLE UTILIZATOR(

id\_utilizator INT DEFAULT UTILIZATOR\_SEQ.NEXTVAL PRIMARY KEY,

id\_abonament INT,

nume VARCHAR(255),

email VARCHAR(255),

data\_inregistrare DATE,

tara VARCHAR(3),

FOREIGN KEY(id\_abonament) REFERENCES ABONAMENT(id\_abonament)

);

INSERT INTO UTILIZATOR (id\_abonament, nume, email, data\_inregistrare, tara) VALUES (1, 'elenacurecheriu', 'elena.curecheriu@s.unibuc.ro', '12-SEP-2005', 'RO');

INSERT INTO UTILIZATOR (id\_abonament, nume, email, data\_inregistrare, tara) VALUES (2, 'anastasia', 'anastasiac@yahoo.com', '03-NOV-2008', 'RO');

INSERT INTO UTILIZATOR (id\_abonament, nume, email, data\_inregistrare, tara) VALUES (3, 'pinkfluffyunicorns', 'dancingonrainbows@gmail.com', '21-MAY-2018', 'US');

INSERT INTO UTILIZATOR (id\_abonament, nume, email, data\_inregistrare, tara) VALUES (4, 'costinel777', 'costinel777@hotmail.com', '04-APR-2014', 'JP');

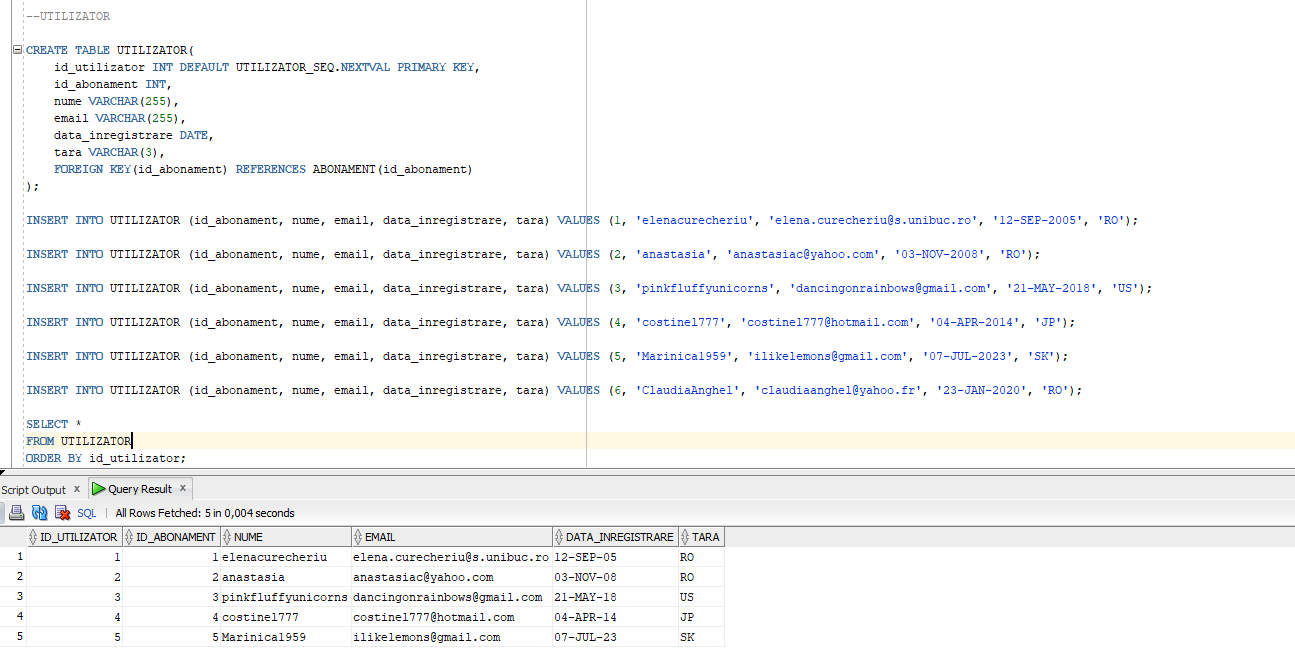
INSERT INTO UTILIZATOR (id\_abonament, nume, email, data\_inregistrare, tara) VALUES (5, 'Marinica1959', 'ilikelemons@gmail.com', '07-JUL-2023', 'SK');

INSERT INTO UTILIZATOR (id\_abonament, nume, email, data\_inregistrare, tara) VALUES (6, 'ClaudiaAnghel', 'claudiaanghel@yahoo.fr', '23-JAN-2020', 'RO');

SELECT \*

FROM UTILIZATOR

ORDER BY id\_utilizator;



1. **Tabela PLAYLIST**

--PLAYLIST

CREATE TABLE PLAYLIST(

id\_playlist INT DEFAULT PLAYLIST\_SEQ.NEXTVAL PRIMARY KEY,

id\_utilizator INT,

nume VARCHAR(255),

data\_creare DATE,

FOREIGN KEY(id\_utilizator) REFERENCES UTILIZATOR(id\_utilizator)

);

INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (1, 'My favourite songs', '07-JUN-2024');

INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (2, 'Chill Vibes', '15-FEB-2023');

INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (3, 'Workout Hits', '23-MAR-2024');

INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (2, 'Road Trip', '08-APR-2022');

INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (3, 'Party Playlist', '11-MAY-2023');

INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (3, 'Relaxing Music', '30-SEP-2021');

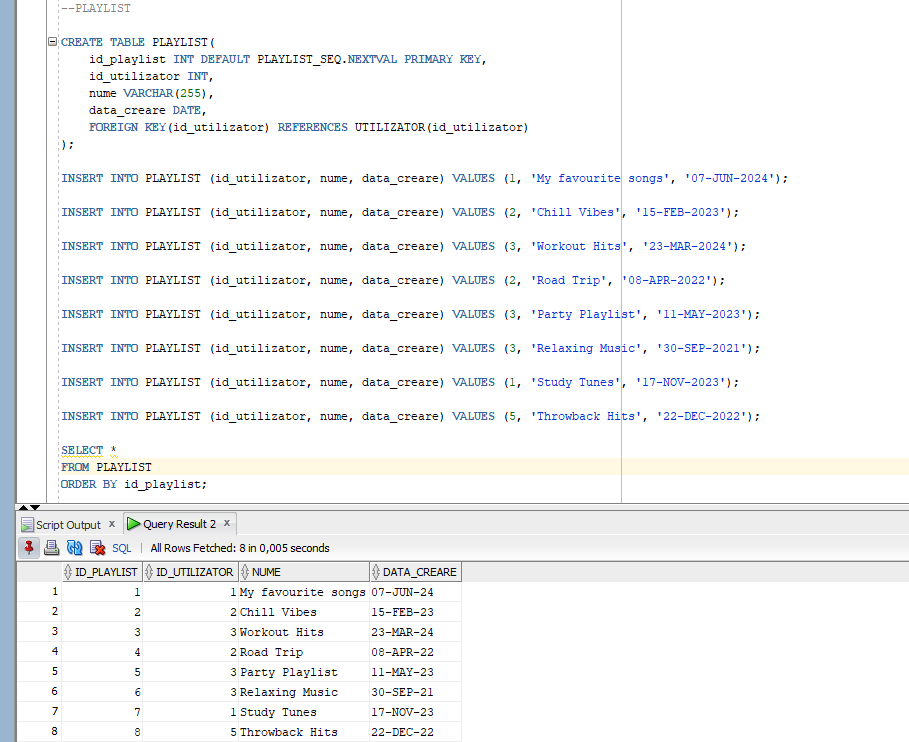
INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (1, 'Study Tunes', '17-NOV-2023');

INSERT INTO PLAYLIST (id\_utilizator, nume, data\_creare) VALUES (5, 'Throwback Hits', '22-DEC-2022');

SELECT \*

FROM PLAYLIST

ORDER BY id\_playlist;



1. **Tabela CASA\_DISCURI**

--CASA DISCURI

CREATE TABLE CASA\_DISCURI(

id\_casa\_discuri INT DEFAULT CASA\_DISCURI\_SEQ.NEXTVAL PRIMARY KEY,

nume VARCHAR(255),

nume\_detinator VARCHAR(255),

prenume\_detinator VARCHAR(255),

an\_infiintare INT

);

INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('Universal Music', 'Grainge', 'Lucian', 1934);

INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('Columbia Records', 'Perry', 'Ron', 1889);

INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('Cat Music', 'Golea', 'Sorin', 1991);

INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('Atlantic Records', 'Kallman', 'Craig', 1947);

INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('HaHaHa Production', 'Maria', 'Andrei Tiberiu', 2009);

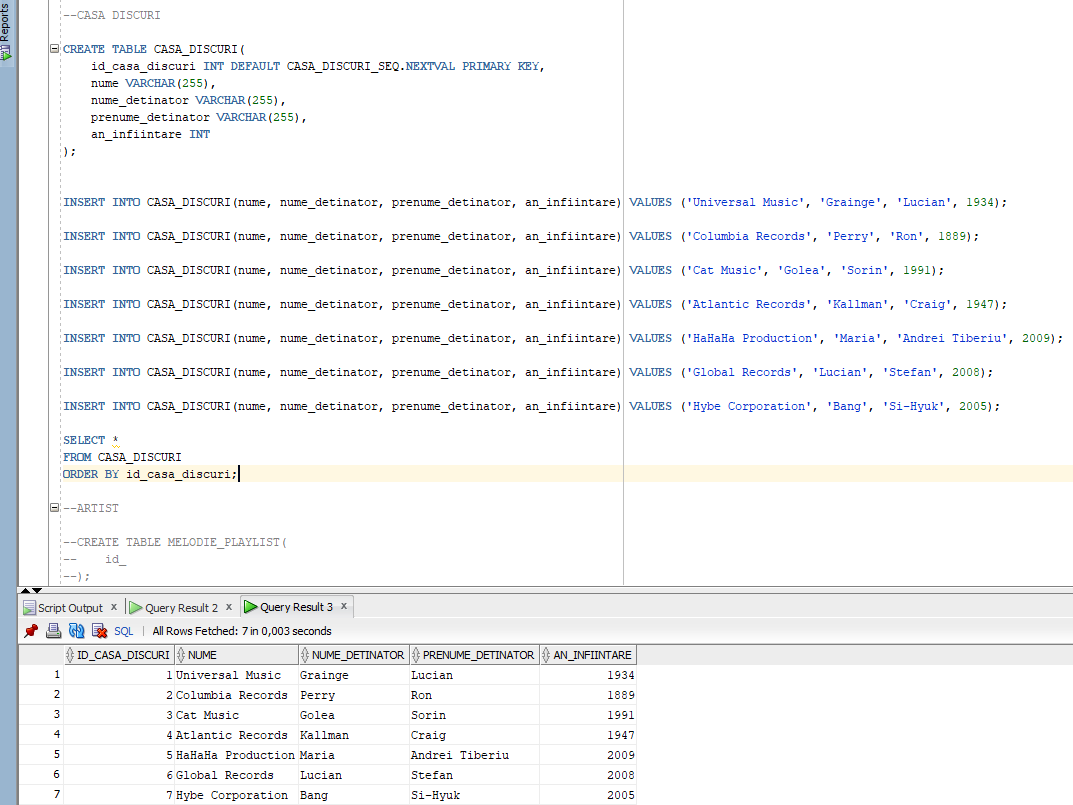
INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('Global Records', 'Lucian', 'Stefan', 2008);

INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('Hybe Corporation', 'Bang', 'Si-Hyuk', 2005);

SELECT \*

FROM CASA\_DISCURI

ORDER BY id\_casa\_discuri;



1. **Tabela ARTIST**

CREATE TABLE ARTIST(

id\_artist INT DEFAULT ARTIST\_SEQ.NEXTVAL PRIMARY KEY,

id\_casa\_discuri INT,

nume VARCHAR(255),

an\_debut INT,

FOREIGN KEY(id\_casa\_discuri) REFERENCES CASA\_DISCURI(id\_casa\_discuri)

);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (1, 'Lana del Rey', 2012);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (7, 'BTS', 2013);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (1, 'Ariana Grande', 2013);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (2, 'Olivia Rodrigo', 2021);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (5, 'Arctic Monkeys', 2002);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (3, 'RM', 2018);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (6, 'The Neighbourhood', 2011);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (4, 'Billie Eilish', 2017);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (4, 'Dua Lipa', 2017);

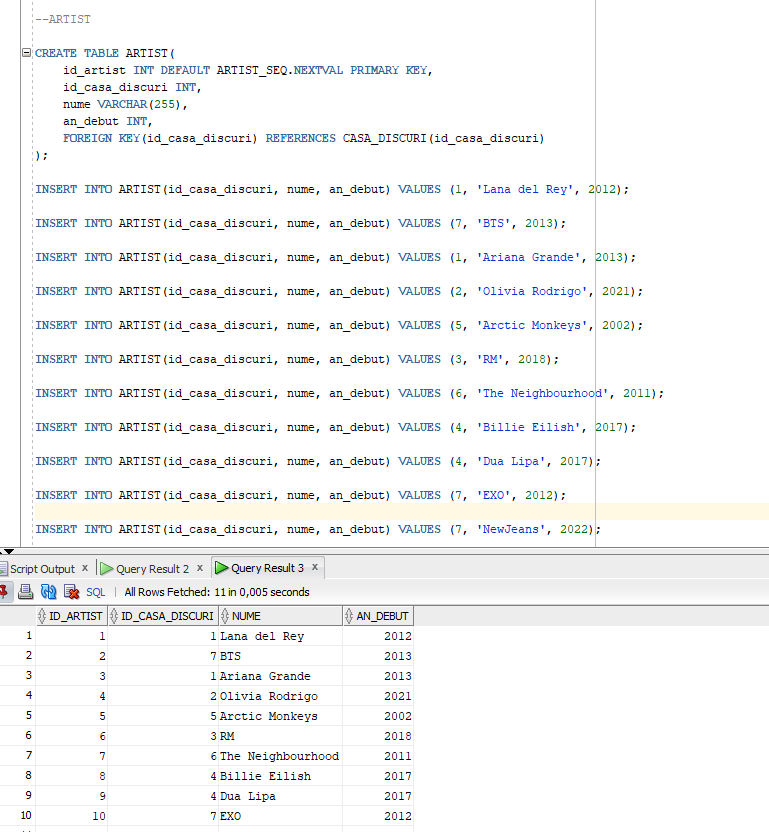
INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (7, 'EXO', 2012);

INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (7, 'NewJeans', 2022);

SELECT \*

FROM ARTIST

ORDER BY id\_artist;



1. **Tabela GEN**

--GEN

CREATE TABLE GEN(

id\_gen INT DEFAULT GEN\_SEQ.NEXTVAL PRIMARY KEY,

nume VARCHAR(255)

);

INSERT INTO GEN(nume) VALUES ('pop');

INSERT INTO GEN(nume) VALUES ('rock');

INSERT INTO GEN(nume) VALUES ('k-pop');

INSERT INTO GEN(nume) VALUES ('indie');

INSERT INTO GEN(nume) VALUES ('rap');

INSERT INTO GEN(nume) VALUES ('trap');

INSERT INTO GEN(nume) VALUES ('hip hop');

INSERT INTO GEN(nume) VALUES ('jazz');

INSERT INTO GEN(nume) VALUES ('dubstep');

INSERT INTO GEN(nume) VALUES ('disco');

INSERT INTO GEN(nume) VALUES ('country');

INSERT INTO GEN(nume) VALUES ('house');

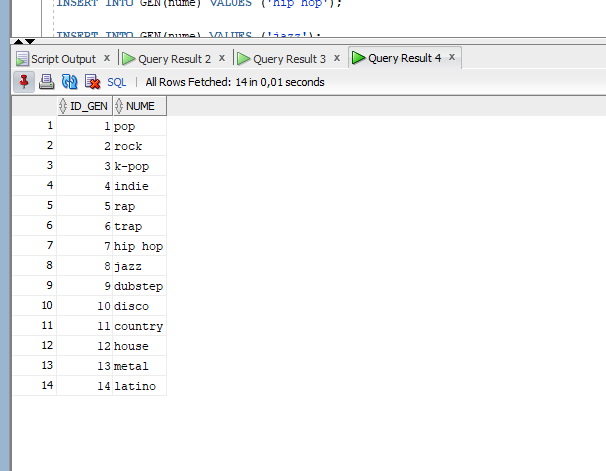
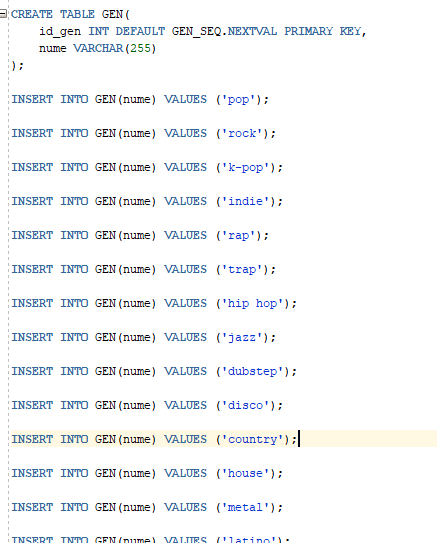
INSERT INTO GEN(nume) VALUES ('metal');

INSERT INTO GEN(nume) VALUES ('latino');

SELECT \*

FROM GEN

ORDER BY id\_gen;



1. **Tabela ALBUM**

CREATE TABLE ALBUM(

id\_album INT DEFAULT ALBUM\_SEQ.NEXTVAL PRIMARY KEY,

titlu VARCHAR(255),

data\_lansare DATE,

durata INT

);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Born To Die', '27-JAN-2012', 2968);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Ultraviolence', '13-JUN-2014', 2637);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Chemtrails Over The Country Club', '19-MAR-2021', 2453);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Love Yourself: Tear', '25-MAY-2018', 2968);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Wings', '09-OCT-2016', 2013);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('eternal sunshine', '08-MAR-2024', 2324);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Dangerous Woman', '20-MAY-2016', 2338);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Sour', '21-MAY-2021', 1968);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Favourite Worst Nightmare', '23-APR-2007', 2238);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('mono.', '23-OCT-2018', 2448);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Indigo', '02-DEC-2022', 1908);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Wiped Out!', '30-OCT-2015', 2718);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('I Love You.', '22-APR-2013', 2687);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('dont smile at me', '11-AUG-2017', 2328);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Happier Than Ever', '30-JUL-2021', 1923);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Future Nostalgia', '27-MAR-2020', 2024);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('The War', '18-JUL-2017', 2783);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('EXODUS', '30-MAR-2015', 2777);

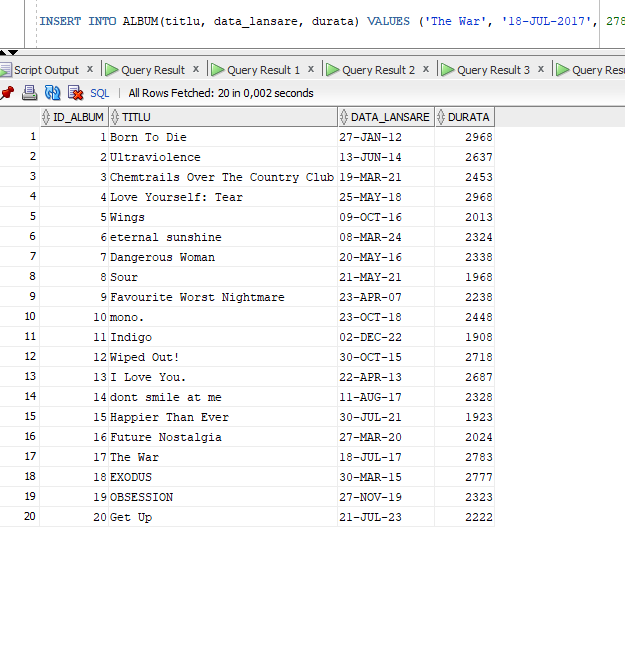
INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('OBSESSION', '27-NOV-2019', 2323);

INSERT INTO ALBUM(titlu, data\_lansare, durata) VALUES ('Get Up', '21-JUL-2023', 2222);

SELECT \*

FROM ALBUM

ORDER BY id\_album;



1. **Tabela MELODIE**

--MELODIE

--MELODIE

CREATE TABLE MELODIE(

id\_melodie INT DEFAULT MELODIE\_SEQ.NEXTVAL PRIMARY KEY,

id\_album INT,

titlu VARCHAR(255),

durata INT,

data\_lansare DATE,

FOREIGN KEY(id\_album) REFERENCES ALBUM(id\_album)

);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (1, 'Video Games', 295, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (2, 'Shades Of Cool', 243, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (3, 'Chemtrails Over The Country Club', 233, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (4, 'FAKE LOVE', 287, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (5, 'Blood Sweat and Tears', 213, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (6, 'bye', 233, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (7, 'Into You', 277, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (8, 'traitor', 260, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (8, 'drivers license', 188, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (9, '505', 200, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (10, 'forever rain', 205, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (10, 'moonchild', 160, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (11, 'Wild Flower', 200, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (12, 'Daddy Issues', 295, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (13, 'Sweater Weather', 242, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (14, 'copycat', 234, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (15, 'Happier Than Ever', 234, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (16, 'Levitating', 199, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (17, 'Ko Ko Bop', 180, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (17, 'Going Crazy', 300, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (18, 'Call Me Baby', 233, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (19, 'OBSESSION', 176, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (20, 'ETA', 245, NULL);

INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (20, 'Get Up', 119, NULL);

SELECT \*

FROM MELODIE

ORDER BY id\_melodie;



1. **Tabela GEN\_ALBUM**

CREATE TABLE GEN\_ALBUM(

id\_album INT,

id\_gen INT,

PRIMARY KEY(id\_album, id\_gen),

FOREIGN KEY(id\_album) REFERENCES ALBUM(id\_album),

FOREIGN KEY(id\_gen) REFERENCES GEN(id\_gen)

);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (1, 1);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (1, 4);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (2, 4);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (3, 11);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (4, 3);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (5, 3);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (6, 1);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (7, 1);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (8, 1);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (8, 2);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (9, 2);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (10, 3);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (10, 4);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (11, 1);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (11, 3);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (11, 4);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (12, 2);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (13, 2);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (14, 4);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (15, 4);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (16, 10);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (16, 12);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (17, 3);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (18, 3);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (19, 3);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (20, 3);

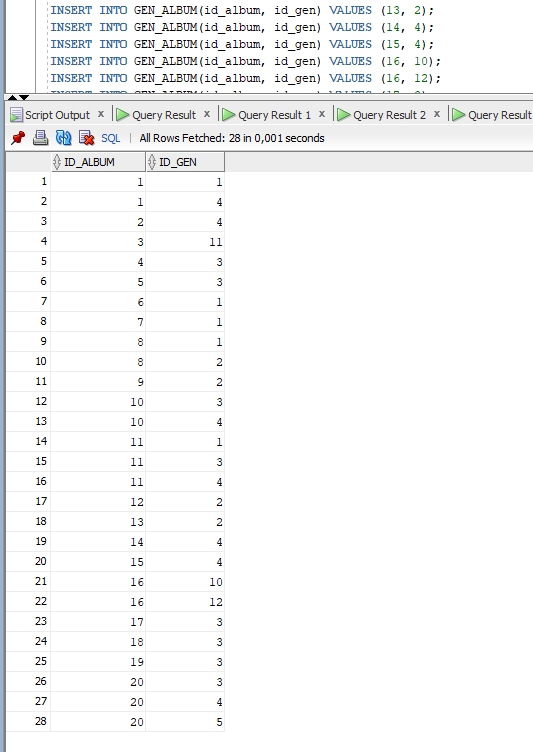
INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (20, 4);

INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (20, 5);

SELECT \*

FROM GEN\_ALBUM

ORDER BY id\_album;



1. **Tabela GEN\_MELODIE**

CREATE TABLE GEN\_MELODIE(

id\_melodie INT,

id\_gen INT,

PRIMARY KEY(id\_melodie, id\_gen),

FOREIGN KEY(id\_melodie) REFERENCES MELODIE(id\_melodie),

FOREIGN KEY(id\_gen) REFERENCES GEN(id\_gen)

);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (1, 4);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (2, 4);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (3, 11);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (4, 3);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (5, 3);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (6, 1);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (7, 1);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (8, 1);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (8, 2);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (9, 1);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (10, 2);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (11, 3);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (12, 4);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (13, 1);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (14, 2);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (15, 2);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (16, 4);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (17, 4);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (18, 10);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (19, 3);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (20, 3);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (21, 3);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (22, 3);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (23, 5);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (24, 5);

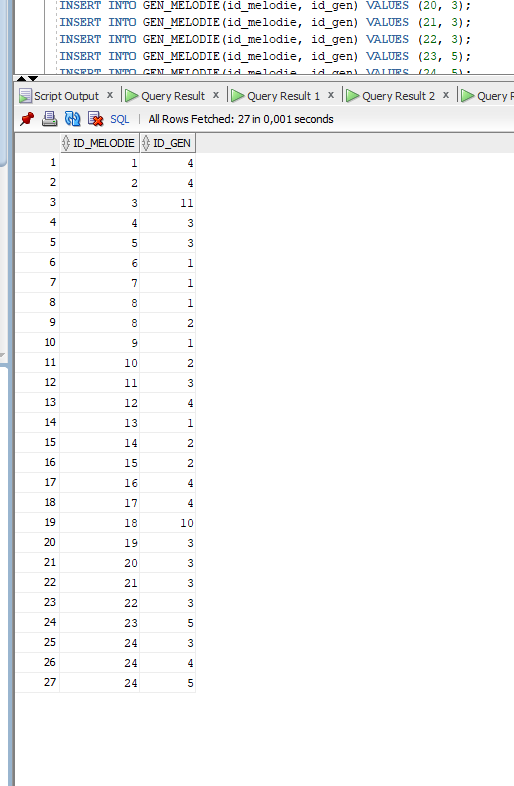
INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (24, 4);

INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (24, 3);

SELECT \*

FROM GEN\_MELODIE

ORDER BY id\_melodie;



1. **Tabela MELODIE\_PLAYLIST**

--melodie\_playlist

CREATE TABLE MELODIE\_PLAYLIST(

id\_melodie INT,

id\_playlist INT,

PRIMARY KEY(id\_melodie, id\_playlist),

FOREIGN KEY(id\_melodie) REFERENCES MELODIE(id\_melodie),

FOREIGN KEY(id\_playlist) REFERENCES PLAYLIST(id\_playlist)

);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (1,1);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (2,1);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (9,1);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (25,1);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (26,1);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (17,1);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (13,1);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (1,2);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (8,2);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (2,3);

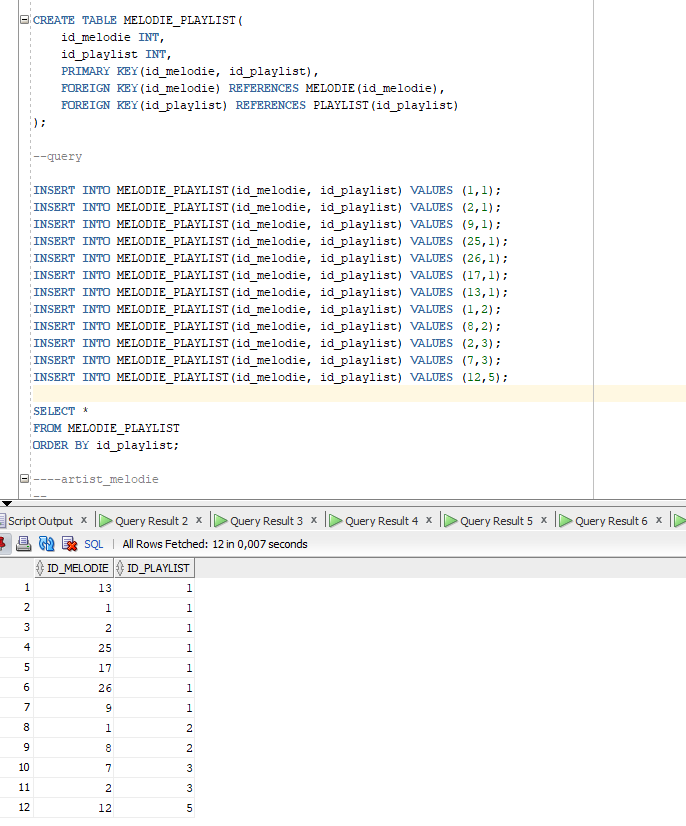
INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (7,3);

INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (12,5);

SELECT \*

FROM MELODIE\_PLAYLIST

ORDER BY id\_playlist;



1. **Tabela ARTIST\_MELODIE**

--artist\_melodie

CREATE TABLE ARTIST\_MELODIE(

id\_artist INT,

id\_melodie INT,

PRIMARY KEY(id\_artist, id\_melodie),

FOREIGN KEY(id\_artist) REFERENCES ARTIST(id\_artist),

FOREIGN KEY(id\_melodie) REFERENCES MELODIE(id\_melodie)

);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(1, 1);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(1, 2);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(1, 3);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(2, 4);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(2, 5);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(2, 11);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(2, 12);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(2, 13);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(3, 6);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(3, 7);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(4, 8);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(4, 9);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(5, 10);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(6, 11);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(6, 12);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(6, 13);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(7, 14);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(7, 15);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(8, 16);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(8, 17);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(9, 18);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(10, 19);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(10, 20);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(10, 21);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(10, 22);

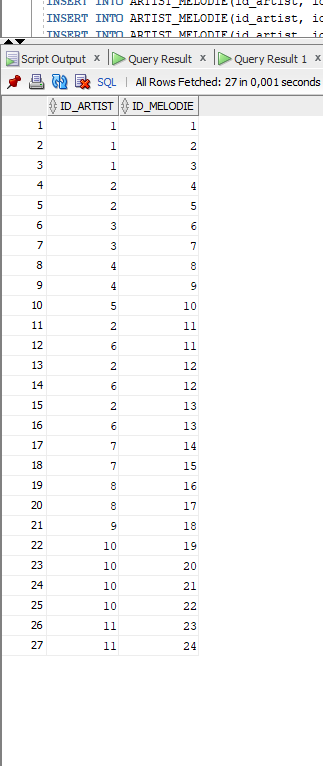
INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(11, 23);

INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(11, 24);

SELECT \*

FROM ARTIST\_MELODIE

ORDER BY id\_melodie;

****

1. **Formulati in limbaj natural si implementati 5 cereri SQL complexe ce vor utiliza, in ansamblul lor, urmatoarele elemente:**
   1. **subcereri sincronizate în care intervin cel puțin 3 tabele**
   2. **subcereri nesincronizate în clauza FROM**
   3. **grupări de date, funcții grup, filtrare la nivel de grupuri cu subcereri nesincronizate (în clauza de HAVING) în care intervin cel puțin 3 tabele (in cadrul aceleiași cereri)**
   4. **ordonări și utilizarea funcțiilor NVL și DECODE (în cadrul aceleiași cereri)**
   5. **utilizarea a cel puțin 2 funcții pe șiruri de caractere, 2 funcții pe date calendaristice, a cel puțin unei expresii CASE**
   6. **utilizarea a cel puțin 1 bloc de cerere (clauza WITH)**
2. Sa se gaseasca utilizatorii cu un nume de utilizator mai lung de 6 caractere, care s-au inregistrat pe platforma dupa anul 2000 si au pretul abonamentului mai mic decat media tuturor abonamentelor utilizatorilor. Se va afisa numele de utilizator concatenat cu tara de origine (intr-o singura coloana), data inregistrarii, tipul de abonament si pretul acestuia.

Elemente utilizate:

* Clauza WITH
* Functii grup (AVG)
* Functii pe siruri de caractere
* Functii pe date calendaristice
* Subcerere nesincronizata in clauza FROM (din blocul de cerere WITH)

--cerinta1

WITH AVG\_PRICE AS (

SELECT AVG(pret) AS avg\_pret

FROM ABONAMENT

),

TABEL AS (

SELECT

UT.NUME AS NUME\_UTILIZATOR,

UT.DATA\_INREGISTRARE AS DATA\_INREGISTRARE,

AB.TIP\_ABONAMENT AS TIP\_ABONAMENT,

AB.PRET AS PRET,

CONCAT(UT.NUME, UT.TARA) AS NUME\_TARA

FROM

UTILIZATOR UT

JOIN

ABONAMENT AB ON UT.ID\_ABONAMENT = AB.ID\_ABONAMENT

WHERE

TO\_NUMBER(TO\_CHAR(UT.DATA\_INREGISTRARE, 'YYYY')) > 2000

AND LENGTH(UT.NUME) > 6

)

SELECT

T.NUME\_TARA,

T.DATA\_INREGISTRARE,

T.TIP\_ABONAMENT,

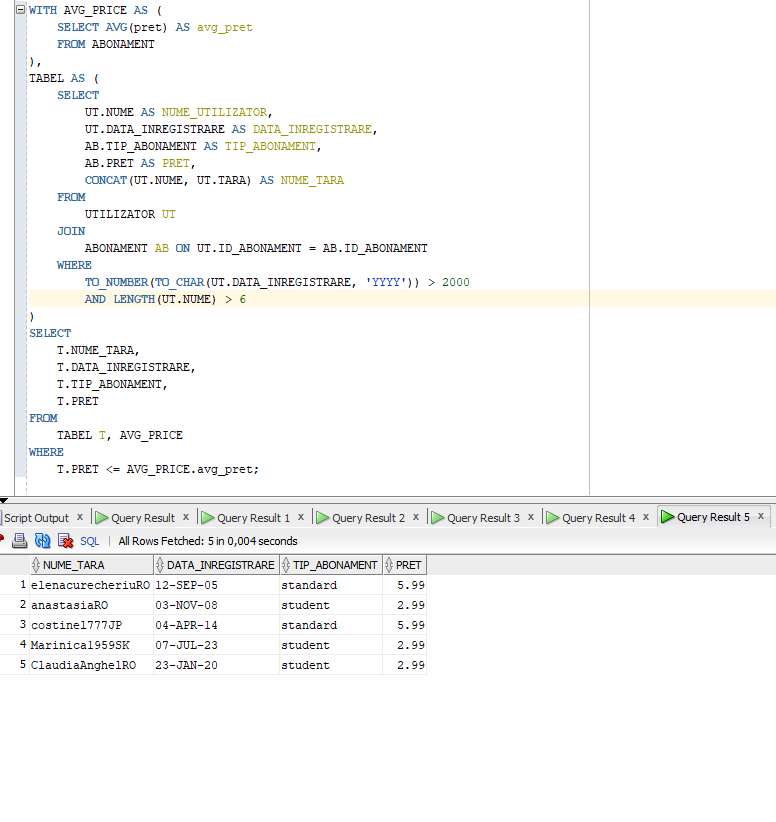
T.PRET

FROM

TABEL T, AVG\_PRICE

WHERE

T.PRET <= AVG\_PRICE.avg\_pret;



1. Selectati primele 7 melodii ca durata (in ordine descrescatoare), afisandu-se titlul acesteia, durata in secunde, titlul albumului in care se regaseste si luna lansarii albumului.

Elemente utilizate:

- subcerere nesincronizata in clauza FROM

- functii pe date calendaristice

- utilizarea unei expresii CASE

- ordonari

--cerinta 2

SELECT \*

FROM (

SELECT

MEL.TITLU AS TITLU\_MELODIE, MEL.DURATA AS DURATA, ALB.TITLU AS TITLU\_ALBUM,

(SELECT

(CASE

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '01' THEN 'Ianuarie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '02' THEN 'Februarie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '03' THEN 'Martie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '04' THEN 'Aprilie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '05' THEN 'Mai'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '06' THEN 'Iunie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '07' THEN 'Iulie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '08' THEN 'August'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '09' THEN 'Septembrie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '10' THEN 'Octombrie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '11' THEN 'Noiembrie'

WHEN TO\_CHAR(ALB.DATA\_LANSARE, 'MM') = '12' THEN 'Decembrie'

END

)

FROM ALBUM ALB

WHERE ALB.ID\_ALBUM = MEL.ID\_ALBUM) AS "LUNA\_LANSARE"

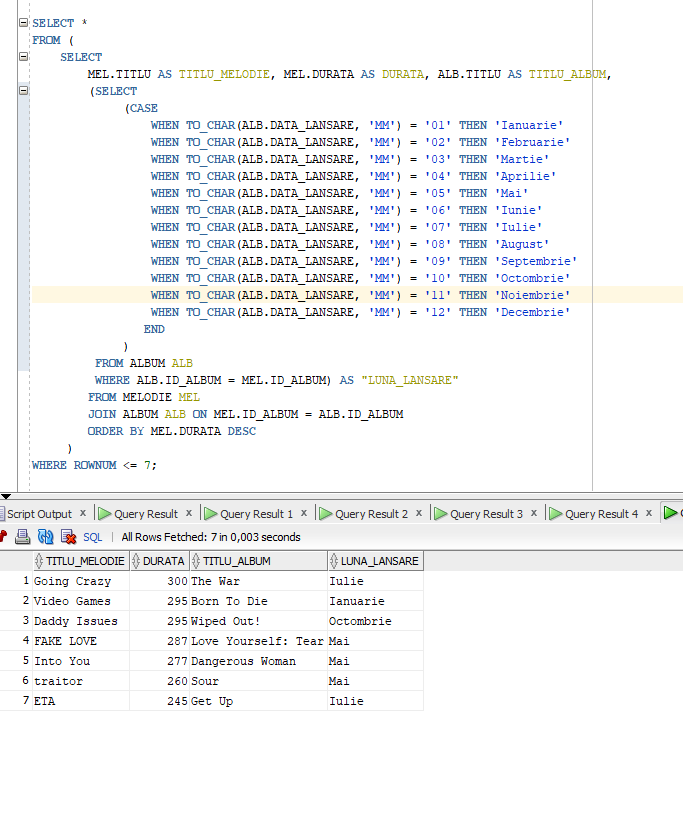
FROM MELODIE MEL

JOIN ALBUM ALB ON MEL.ID\_ALBUM = ALB.ID\_ALBUM

ORDER BY MEL.DURATA DESC

)

WHERE ROWNUM <= 7;



1. Pentru fiecare playlist care contine cel putin o melodie, afisati numarul de luni care au trecut de la data crearii playlistului. In plus, sa se afiseze numele playlist-ului, id-ul sau, data crearii, numarul de melodii si utilizatorul care detine playlist-ul.

Elemente utilizate:

* Functii pe date calendaristice
* Functia NVL
* Grupari de date, filtrare la nivel de grupuri cu subcereri nesincronizate in care intervin cel putin 3 tabele
* Functii grup (COUNT)

--cerinta 3

SELECT DISTINCT

PL.NUME AS NUME\_PLAYLIST,

PL.ID\_PLAYLIST AS ID\_PLAYLIST,

PL.DATA\_CREARE AS DATA\_CREARE,

MONTHS\_BETWEEN(SYSDATE, PL.DATA\_CREARE) AS NR\_LUNI,

NVL(COUNT(PL.ID\_PLAYLIST), 0) AS NR\_MELODII,

(

SELECT DISTINCT UT.NUME AS "NUME\_UTILIZATOR"

FROM UTILIZATOR UT

WHERE UT.ID\_UTILIZATOR = PL.ID\_UTILIZATOR

),

(

SELECT DISTINCT NVL(COUNT(MP.ID\_PLAYLIST), 0) AS "NR\_MELODII"

FROM MELODIE\_PLAYLIST MP

WHERE MP.ID\_PLAYLIST = PL.ID\_PLAYLIST

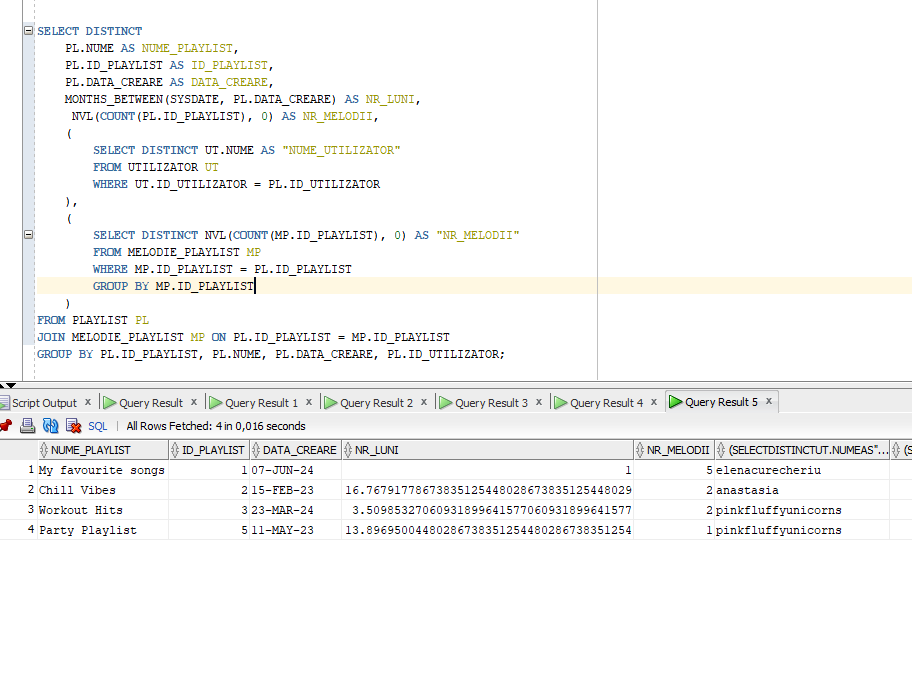
GROUP BY MP.ID\_PLAYLIST

)

FROM PLAYLIST PL

JOIN MELODIE\_PLAYLIST MP ON PL.ID\_PLAYLIST = MP.ID\_PLAYLIST

GROUP BY PL.ID\_PLAYLIST, PL.NUME, PL.DATA\_CREARE, PL.ID\_UTILIZATOR;



1. Pentru fiecare gen care are melodii inregistrate in baza de date, sa se afiseze numele, numarul de melodii si un mesaj specific.

Elemente utilizate:

* Ordonari, utilizarea functiilor NVL si DECODE in cadrul aceleiasi cereri
* Functii grup
* Clauza HAVING
* Functii pe siruri de caractere

SELECT

G.NUME AS NUME\_GEN,

COUNT(GM.ID\_MELODIE) AS NR\_MELODII,

DECODE(

NVL(COUNT(GM.ID\_MELODIE), 0),

0, 'Nu exista melodii in acest gen',

1, 'Exista o melodie in acest gen',

'Exista ' || COUNT(GM.ID\_MELODIE) || ' melodii in acest gen'

) AS SONGS\_EXIST

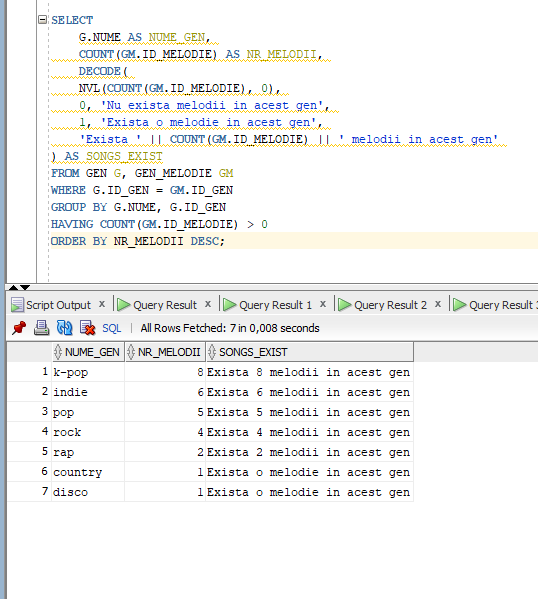
FROM GEN G, GEN\_MELODIE GM

WHERE G.ID\_GEN = GM.ID\_GEN

GROUP BY G.NUME, G.ID\_GEN

HAVING COUNT(GM.ID\_MELODIE) > 0

ORDER BY NR\_MELODII DESC;



1. Afisati numele artistilor care au debutat intre anii 2000 si 2004, au cel putin doua melodii si apartin unei case de discuri infiintata dupa anul 1950

Elemente utilizate:

* Subcerere sincronizata in care intervin cel putin 3 tabele
* Clauza WITH
* Group
* Subcerere nesincronizata in clauza FROM (din WITH)
* Functii pe siruri de caractere

WITH TABEL AS ( --WITH, SUBCERERE NESINCRONIZATA

SELECT DISTINCT

ART.NUME AS NUME\_ARTIST,

ART.AN\_DEBUT AS AN\_DEBUT

FROM ARTIST ART

JOIN ARTIST\_MELODIE AM on ART.id\_artist = AM.id\_artist

JOIN CASA\_DISCURI CD on ART.id\_casa\_discuri = CD.id\_casa\_discuri

WHERE ART.NUME LIKE '%A%' OR ART.NUME LIKE '%a%'

GROUP BY ART.NUME, ART.AN\_DEBUT

)

SELECT

T.NUME\_ARTIST, T.AN\_DEBUT

FROM TABEL T

WHERE

T.AN\_DEBUT BETWEEN 2000 AND 2024

AND ( --CERERI SINCRONIZATE

SELECT COUNT(AM.ID\_MELODIE)

FROM ARTIST\_MELODIE AM

WHERE AM.ID\_ARTIST = (

SELECT ART.ID\_ARTIST

FROM ARTIST ART

WHERE ART.NUME = T.NUME\_ARTIST

)

) >= 2

AND

T.NUME\_ARTIST IN (

SELECT ART.NUME

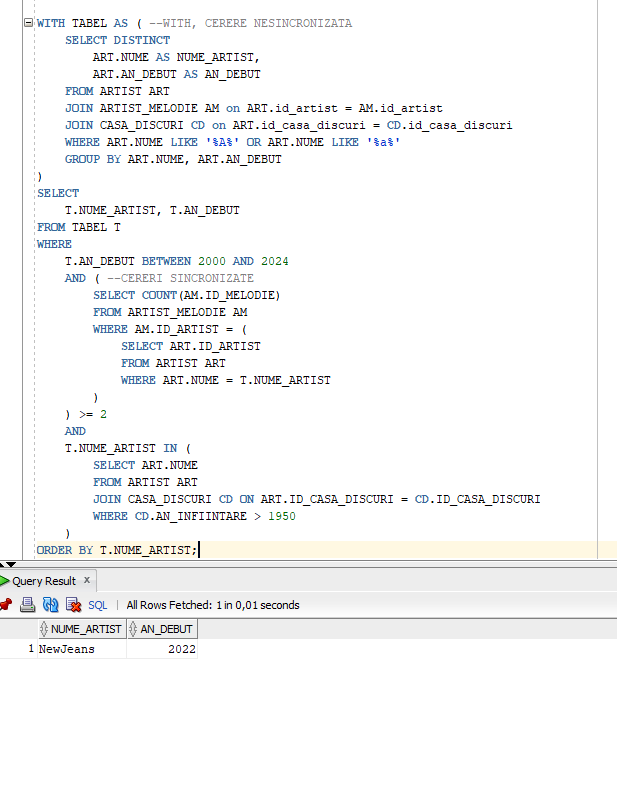
FROM ARTIST ART

JOIN CASA\_DISCURI CD ON ART.ID\_CASA\_DISCURI = CD.ID\_CASA\_DISCURI

WHERE CD.AN\_INFIINTARE > 1950

)

ORDER BY T.NUME\_ARTIST;



1. **Implementarea a 3 operații de actualizare și de suprimare a datelor utilizând subcereri.**
2. Actualizarea coloanei "pret" din tabela ABONAMENT pentru abonamentele utilizatorilor cu conturile create inainte de 2020.

UPDATE ABONAMENT

SET pret = pret \* 1.1 --pretul creste cu 10%

WHERE id\_abonament IN (

SELECT id\_abonament

FROM UTILIZATOR

WHERE EXTRACT(YEAR FROM data\_inregistrare) < 2020

);

SELECT \*

FROM ABONAMENT

ORDER BY id\_abonament;

1. Stergerea inregistrarilor din tabela "UTILIZATOR" pentru utilizatorii are nu detin niciun playlist.

DELETE FROM UTILIZATOR

WHERE id\_utilizator NOT IN (

SELECT DISTINCT id\_utilizator

FROM PLAYLIST

);

1. Actualizarea coloanei "nume" din tabela CASA\_DISCURI pentru fiecare detinator de casa de discuri ai carui prenume incepe cu litera "L" si au minimum 2 artisti contractati.

UPDATE CASA\_DISCURI

SET nume = CONCAT(nume, ' special')

WHERE id\_casa\_discuri IN (

SELECT CD.id\_casa\_discuri

FROM CASA\_DISCURI CD

JOIN ARTIST ART ON CD.id\_casa\_discuri = ART.id\_casa\_discuri

WHERE CD.prenume\_detinator LIKE 'L%'

GROUP BY CD.id\_casa\_discuri

HAVING COUNT(ART.id\_artist) >= 2

);

SELECT \*

FROM CASA\_DISCURI

ORDER BY id\_casa\_discuri;

