***Proiect Sisteme de gestiune a bazelor de date – aplicatie de streaming de muzica tip Spotify***

**Curecheriu Elena**

**Seria 24, grupa 242**

**Anul universitar 2024 – 2025**

**Facultatea de Matematica si Informatica**

**Universitatea din Bucuresti**

Cuprins

[Introducere: 3](#_Toc187420484)

[Cerinta 1: Prezentați pe scurt baza de date (utilitatea ei). 4](#_Toc187420485)

[Cerinta 2: Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română. 5](#_Toc187420486)

[Cerinta 3: Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română. 6](#_Toc187420487)

[Cerinta 4: Implementați în Oracle diagrama conceptuală realizată: definiți toate tabelele, adăugând toate constrângerile de integritate necesare (chei primare, cheile externe etc). 7](#_Toc187420488)

[Cerinta 5: Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă). 24](#_Toc187420489)

[Cerinta 6: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul. 49](#_Toc187420490)

[Cerinta 7: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul. 55](#_Toc187420491)

[Cerinta 8: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate. 61](#_Toc187420492)

[Cerinta 9: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate. 68](#_Toc187420493)

[Cerinta 10: Definiți un trigger de tip LMD la nivel de comandă. Declanșați trigger-ul. 77](#_Toc187420494)

[Cerinta 11: Definiți un trigger de tip LMD la nivel de linie. Declanșați trigger-ul. 80](#_Toc187420495)

[Cerinta 12: Definiți un trigger de tip LDD. Declanșați trigger-ul. 83](#_Toc187420496)

# Introducere:

Tema aleasa pentru proiect reprezinta 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.

Infrastructura utilizata pentru implementarea acestuia este:

* Versiunea SGBD-ului: Oracle Database 21c Express Edition
* RAM alocat: 16 GB
* Sistem de operare: Windows 11
* Utilizare masina virtuala: nu

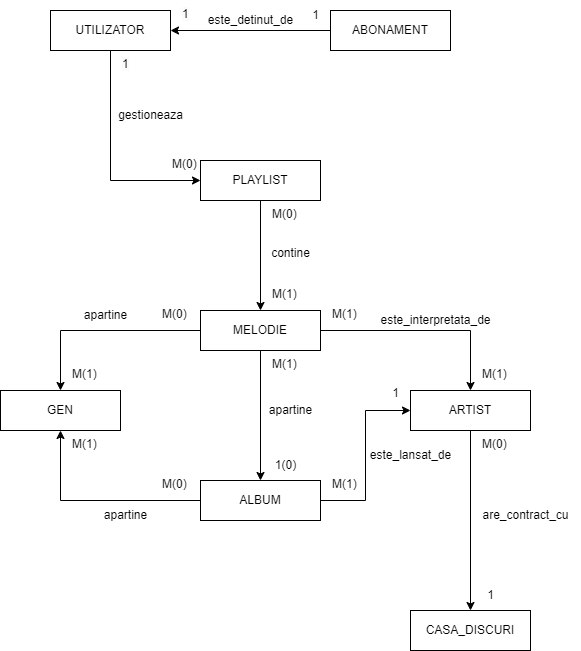
# Cerinta 1: Prezentați pe scurt baza de date (utilitatea ei).

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.

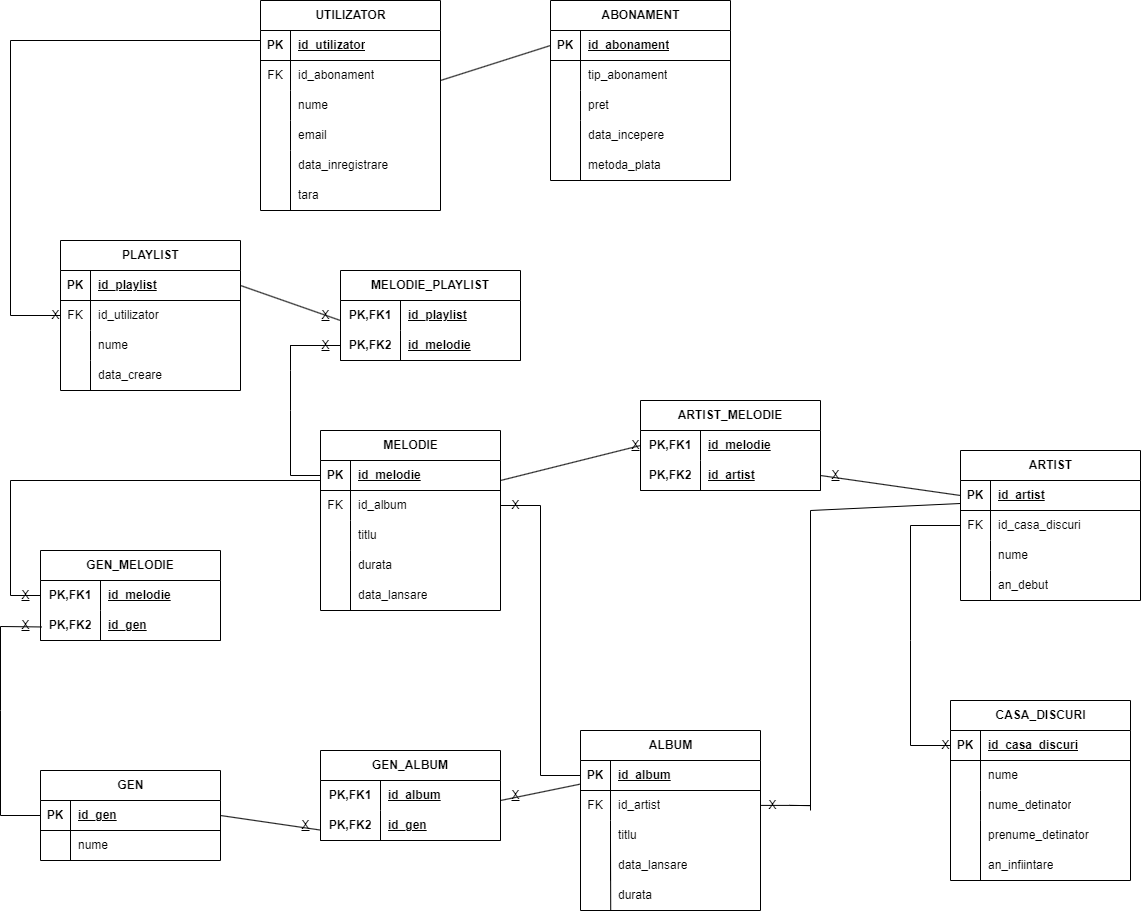
Utilizatorii detin abonamente si pot crea playlist-uri care contin melodii. Melodiile pot apartine unui album, sunt interpretate de catre un artist, se incadreaza intr-un gen muzical, iar artistii au contract cu o casa de discuri.

Ca reguli de functionare, un utilizator poate avea un singur abonament pe platforma, iar respectivul abonament este detinut de catre un singur utilizator. Aplicatia poate avea mai multi utilizatori care gestioneaza playlist-uri. Aceste playlist-uri contin cel putin o melodie, care sunt interpretate de unul sau mai multi artisti, se incadreaza in cel putin un gen muzical si pot apartine unui album, mai multor albume sau niciunui album, putand fi lansata sub forma de single. Albumele apartin unui singur artist si se incadreaza in cel putin un gen muzical, iar fiecare artist este reprezentat de o singura casa de discuri, acestea avand contract cu minimum un artist. In plus, melodiile pot sa apara sau nu in playlist-uri, playlist-urile fiind detinute de catre un unic utilizator.

# Cerinta 2: Realizați diagrama entitate-relație (ERD): entitățile, relațiile și atributele trebuie definite în limba română.



# Cerinta 3: Pornind de la diagrama entitate-relație realizați diagrama conceptuală a modelului propus, integrând toate atributele necesare: entitățile, relațiile și atributele trebuie definite în limba română.



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

CREATE SEQUENCE UTILIZATOR\_SEQ START WITH 1;

CREATE SEQUENCE ABONAMENT\_SEQ START WITH 1;

CREATE SEQUENCE PLAYLIST\_SEQ START WITH 1;

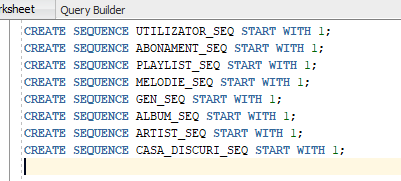
CREATE SEQUENCE MELODIE\_SEQ START WITH 1;

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;

****

--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')),

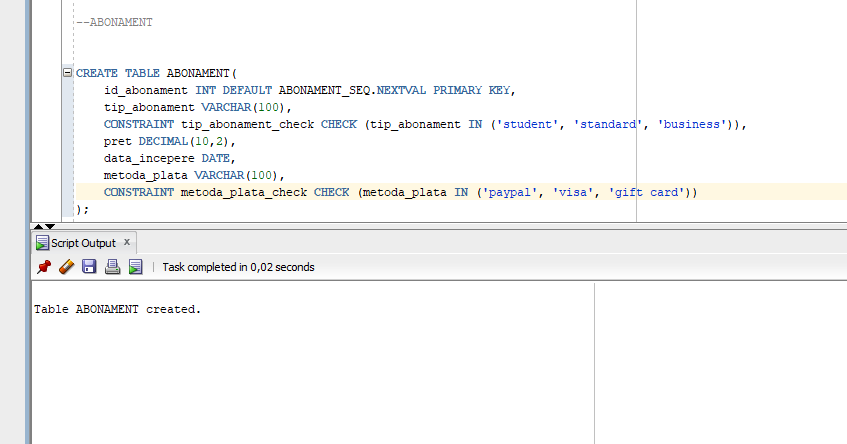
pret DECIMAL(10,2),

data\_incepere DATE,

metoda\_plata VARCHAR(100),

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

);



--UTILIZATOR

CREATE TABLE UTILIZATOR(

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

id\_abonament INT,

nume VARCHAR(255),

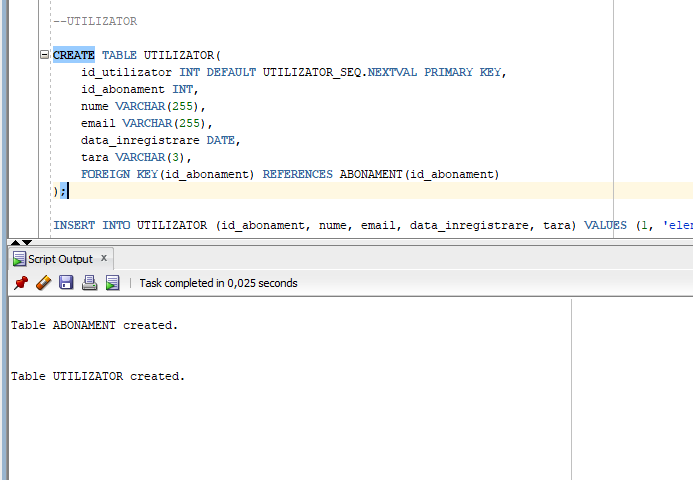
email VARCHAR(255),

data\_inregistrare DATE,

tara VARCHAR(3),

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

);



--PLAYLIST

CREATE TABLE PLAYLIST(

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

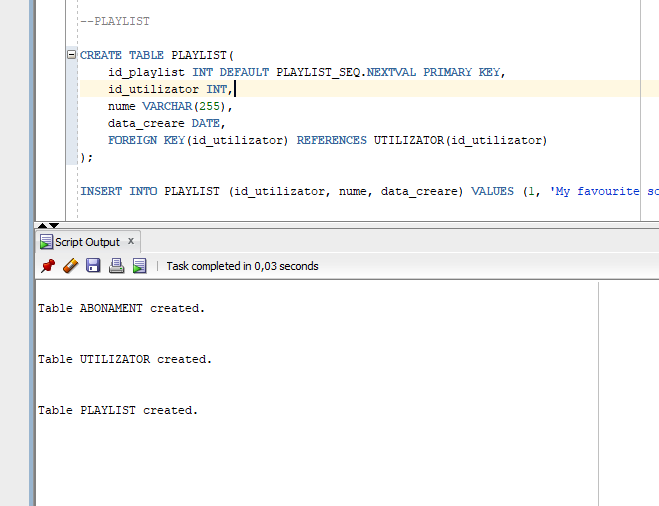
id\_utilizator INT,

nume VARCHAR(255),

data\_creare DATE,

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

);



--CASA DISCURI

CREATE TABLE CASA\_DISCURI(

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

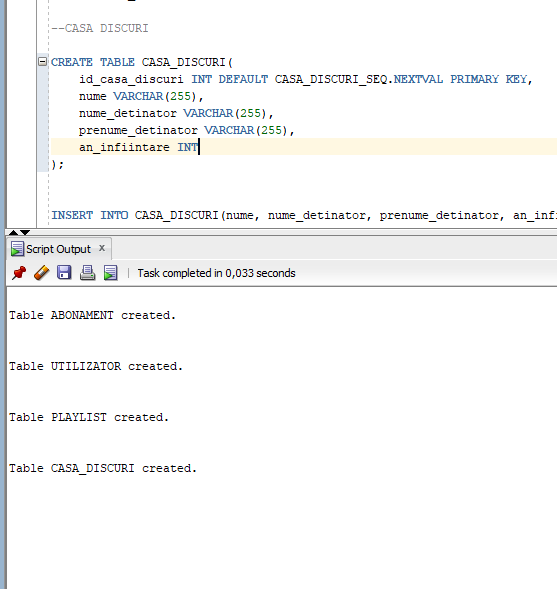
nume VARCHAR(255),

nume\_detinator VARCHAR(255),

prenume\_detinator VARCHAR(255),

an\_infiintare INT

);



--ARTIST

CREATE TABLE ARTIST(

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

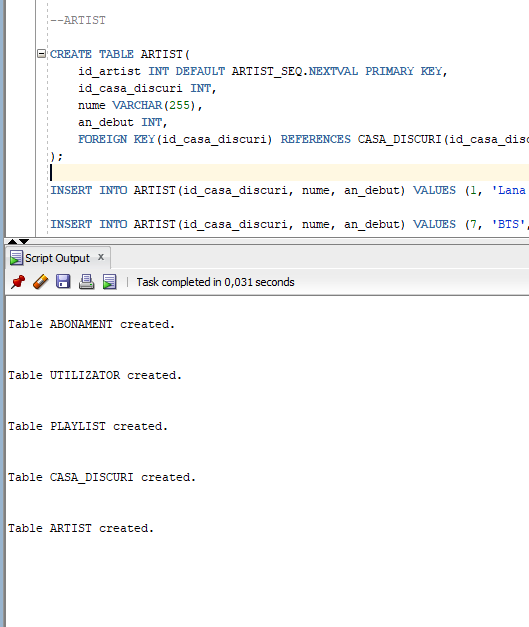
id\_casa\_discuri INT,

nume VARCHAR(255),

an\_debut INT,

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

);



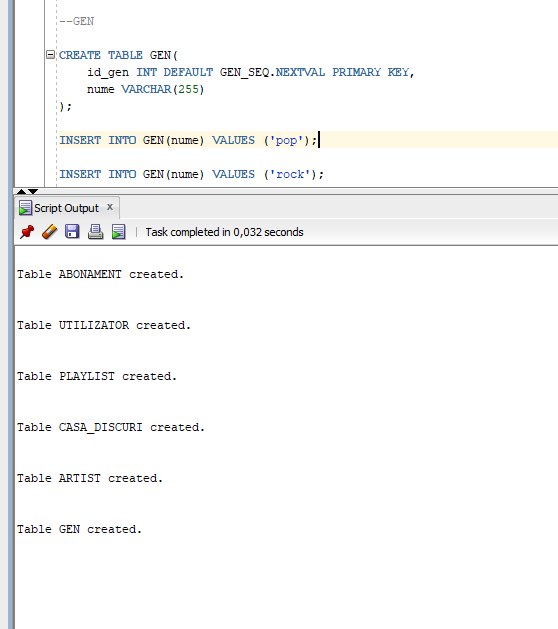
--gen

CREATE TABLE GEN(

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

nume VARCHAR(255)

);



--album

CREATE TABLE ALBUM(

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

id\_artist INT,

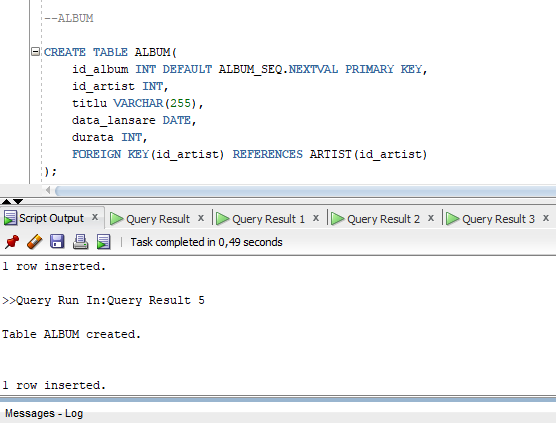
titlu VARCHAR(255),

data\_lansare DATE,

durata INT,

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

);



--MELODIE

CREATE TABLE MELODIE(

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

id\_album INT,

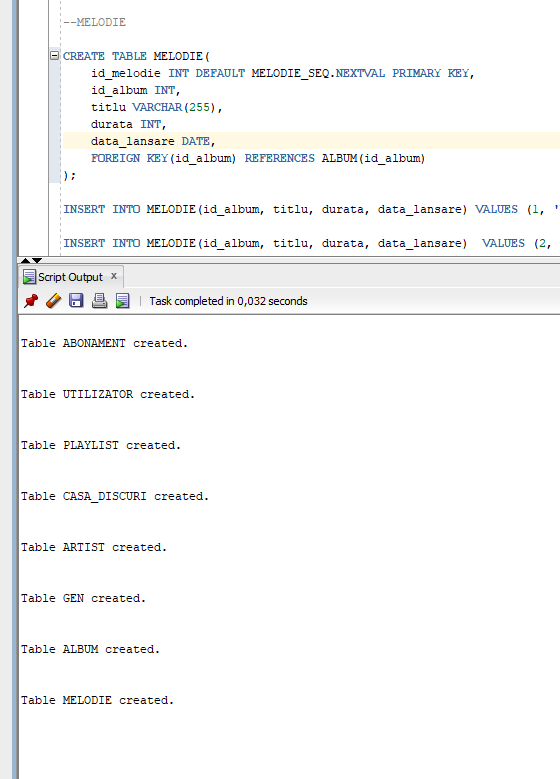
titlu VARCHAR(255),

durata INT,

data\_lansare DATE,

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

);



--tabelele asociative

--gen\_album

CREATE TABLE GEN\_ALBUM(

id\_album INT,

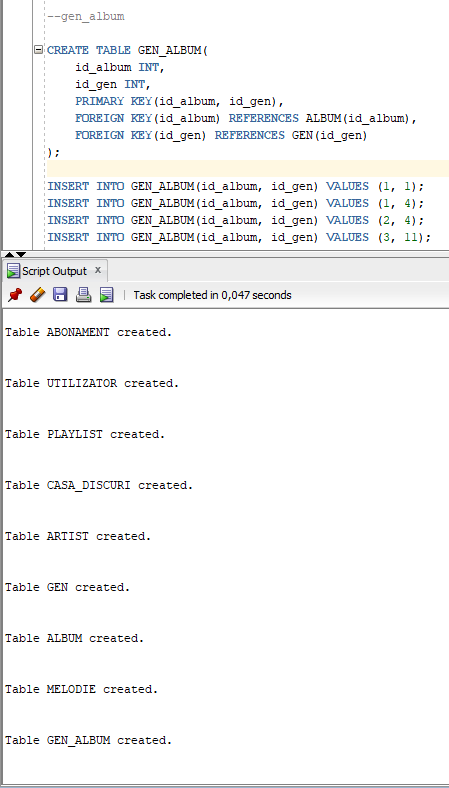
id\_gen INT,

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

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

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

);



CREATE TABLE GEN\_MELODIE(

id\_melodie INT,

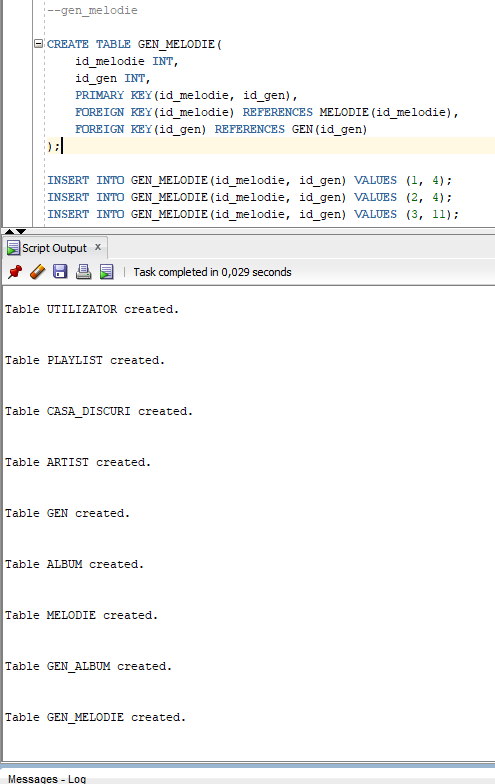
id\_gen INT,

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

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

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

);



--melodie\_playlist

CREATE TABLE MELODIE\_PLAYLIST(

id\_melodie INT,

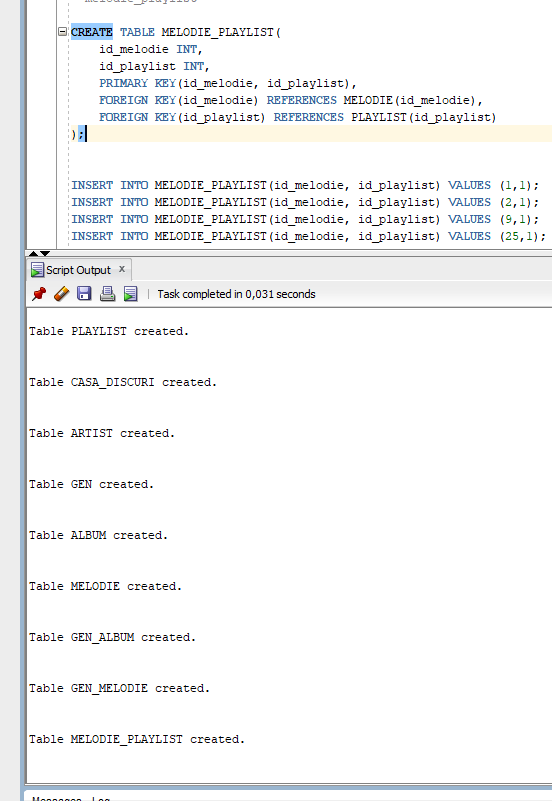
id\_playlist INT,

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

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

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

);



--artist\_melodie

CREATE TABLE ARTIST\_MELODIE(

id\_artist INT,

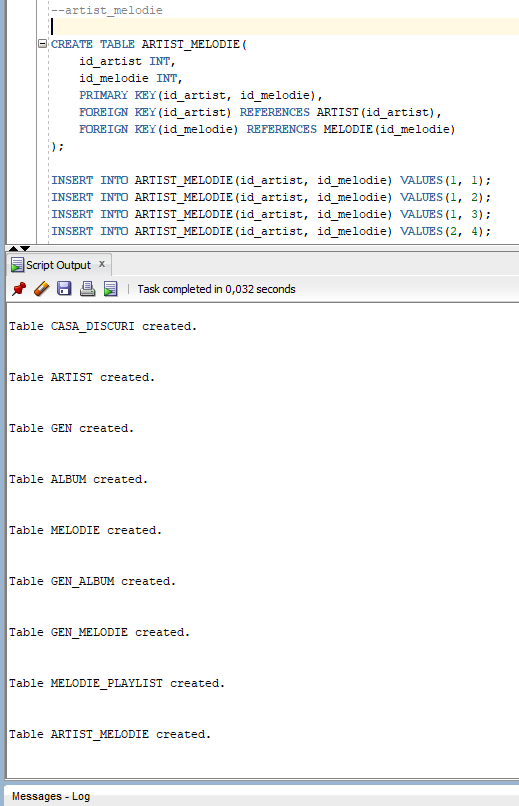
id\_melodie INT,

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

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

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

);



# Cerinta 5: Adăugați informații coerente în tabelele create (minim 5 înregistrări pentru fiecare entitate independentă; minim 10 înregistrări pentru fiecare tabelă asociativă).

--ABONAMENT

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');

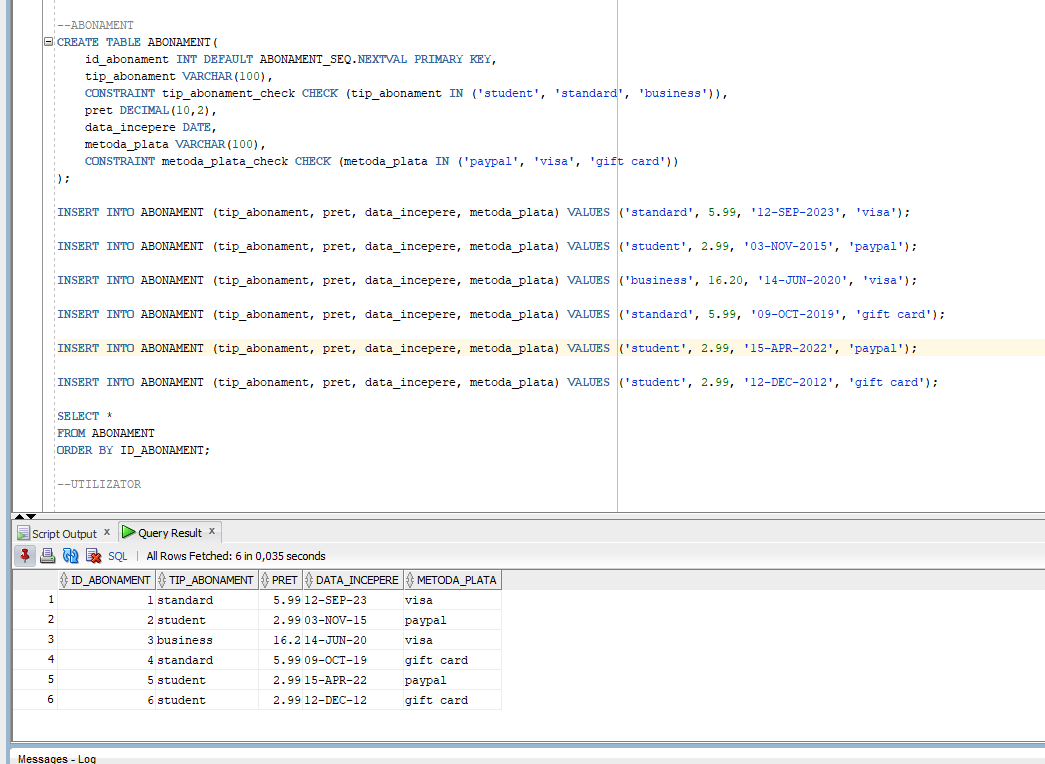
INSERT INTO ABONAMENT (tip\_abonament, pret, data\_incepere, metoda\_plata) VALUES ('student', 2.99, '15-APR-2022', 'paypal');

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;



--UTILIZATOR

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');

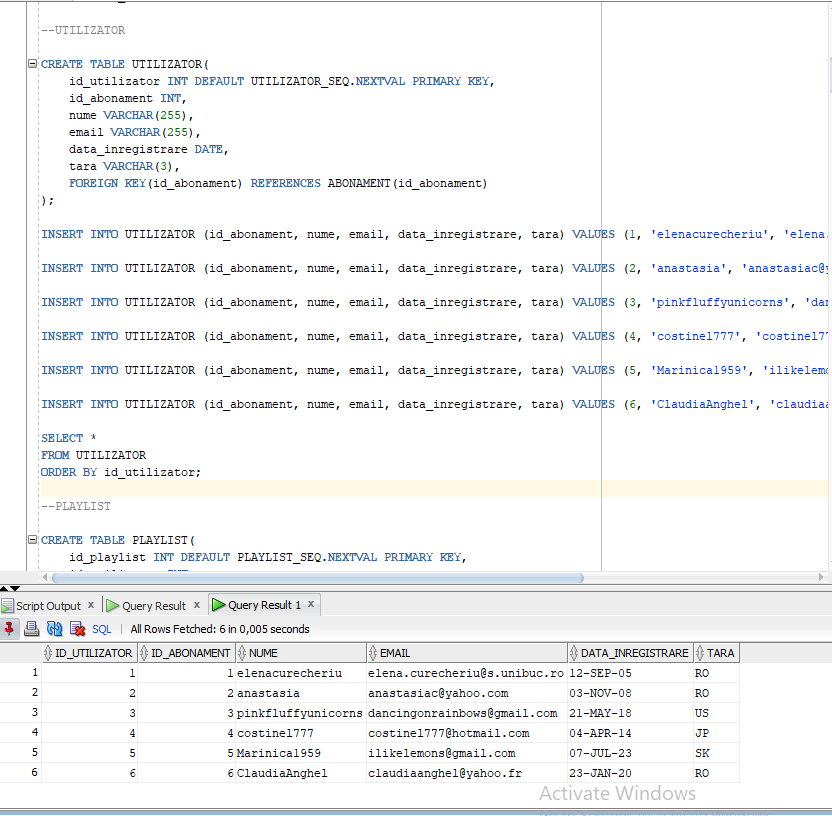
INSERT INTO UTILIZATOR (id\_abonament, nume, email, data\_inregistrare, tara) VALUES (5, 'Marinica1959', 'ilikelemons@gmail.com', '07-JUL-2023', 'SK');

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;



--PLAYLIST

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;

--CASA DISCURI

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);

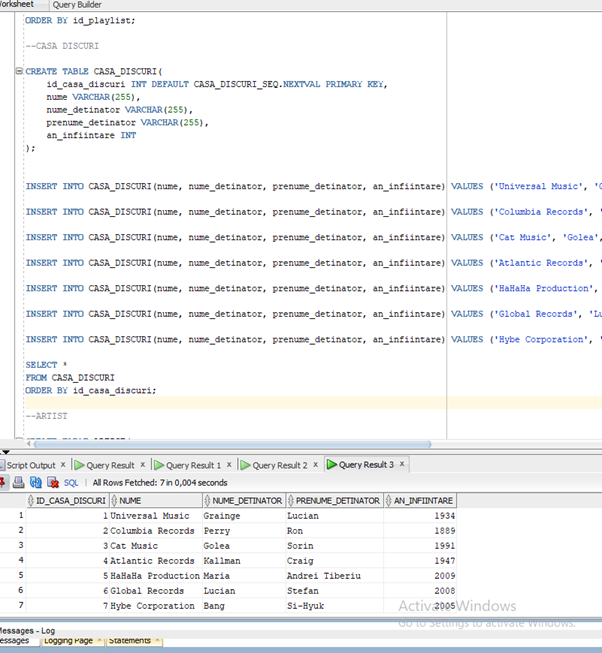
INSERT INTO CASA\_DISCURI(nume, nume\_detinator, prenume\_detinator, an\_infiintare) VALUES ('Global Records', 'Lucian', 'Stefan', 2008);

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;



--ARTIST

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);

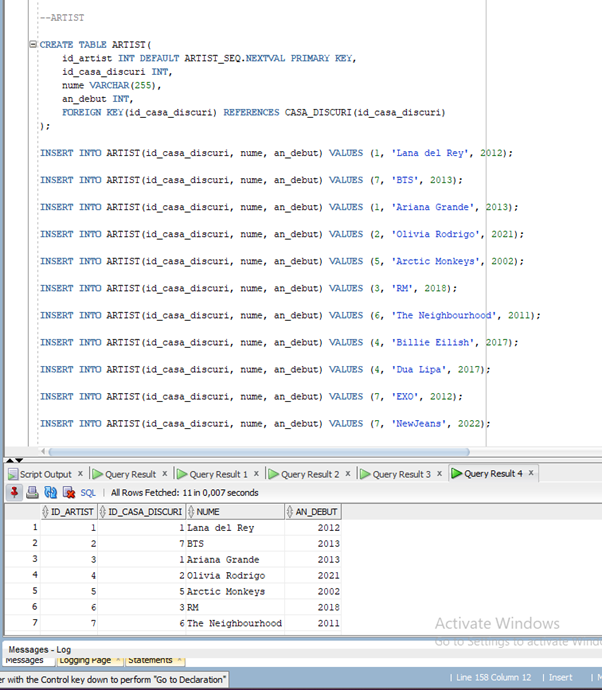
INSERT INTO ARTIST(id\_casa\_discuri, nume, an\_debut) VALUES (7, 'EXO', 2012);

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

SELECT \*

FROM ARTIST

ORDER BY id\_artist;



--GEN

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');

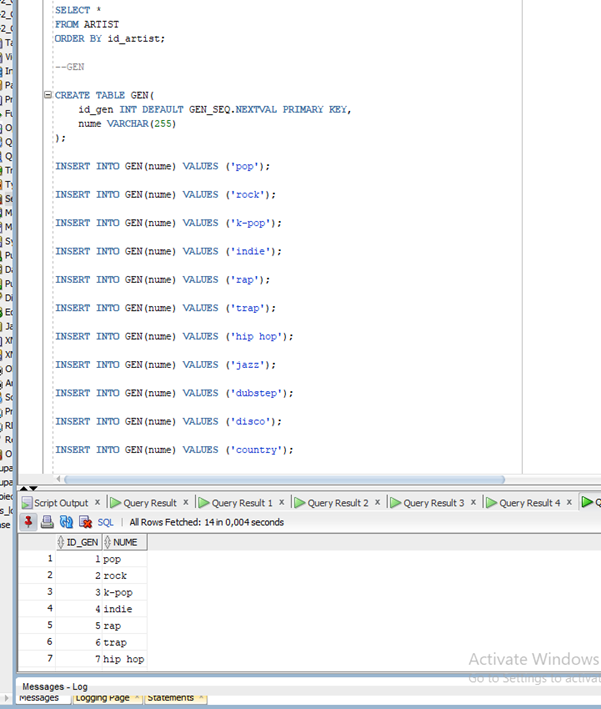
INSERT INTO GEN(nume) VALUES ('metal');

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

SELECT \*

FROM GEN

ORDER BY id\_gen;



--ALBUM

CREATE TABLE ALBUM(

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

id\_artist INT,

titlu VARCHAR(255),

data\_lansare DATE,

durata INT,

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

);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (1, 'Born To Die', '27-JAN-2012', 2968);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (1, 'Ultraviolence', '13-JUN-2014', 2637);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (1, 'Chemtrails Over The Country Club', '19-MAR-2021', 2453);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (2, 'Love Yourself: Tear', '25-MAY-2018', 2968);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (2, 'Wings', '09-OCT-2016', 2013);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (3, 'eternal sunshine', '08-MAR-2024', 2324);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (3, 'Dangerous Woman', '20-MAY-2016', 2338);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (4, 'Sour', '21-MAY-2021', 1968);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (5, 'Favourite Worst Nightmare', '23-APR-2007', 2238);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (6, 'mono.', '23-OCT-2018', 2448);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (6, 'Indigo', '02-DEC-2022', 1908);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (7, 'Wiped Out!', '30-OCT-2015', 2718);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (7, 'I Love You.', '22-APR-2013', 2687);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (8, 'dont smile at me', '11-AUG-2017', 2328);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (8, 'Happier Than Ever', '30-JUL-2021', 1923);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (9, 'Future Nostalgia', '27-MAR-2020', 2024);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (10, 'The War', '18-JUL-2017', 2783);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (10, 'EXODUS', '30-MAR-2015', 2777);

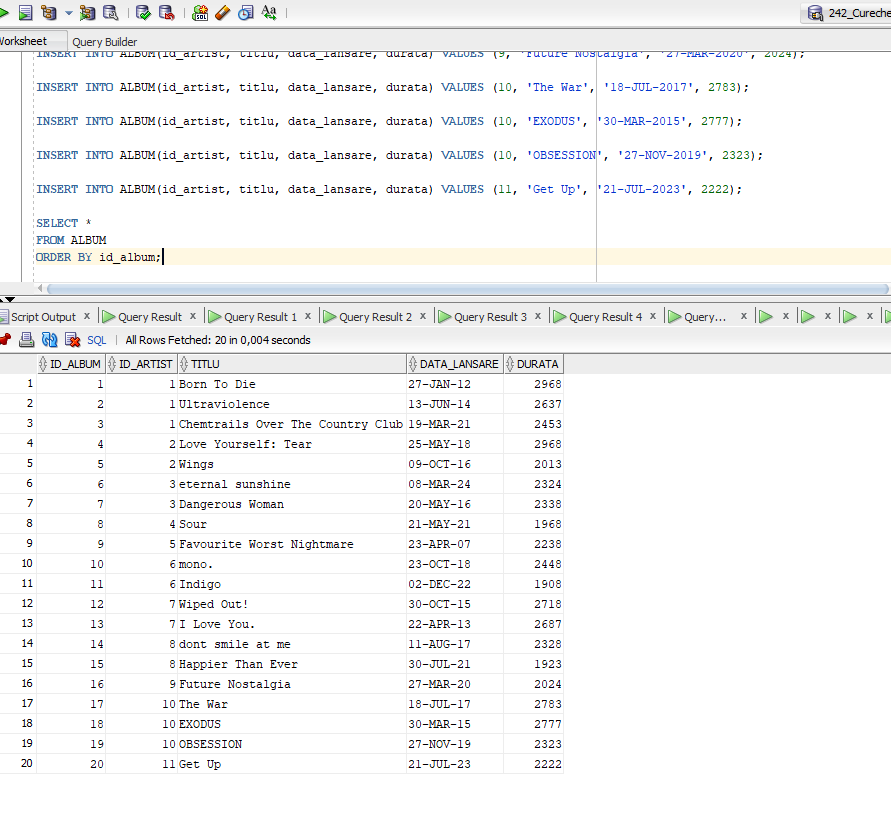
INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (10, 'OBSESSION', '27-NOV-2019', 2323);

INSERT INTO ALBUM(id\_artist, titlu, data\_lansare, durata) VALUES (11, 'Get Up', '21-JUL-2023', 2222);

SELECT \*

FROM ALBUM

ORDER BY id\_album;



--MELODIE

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);

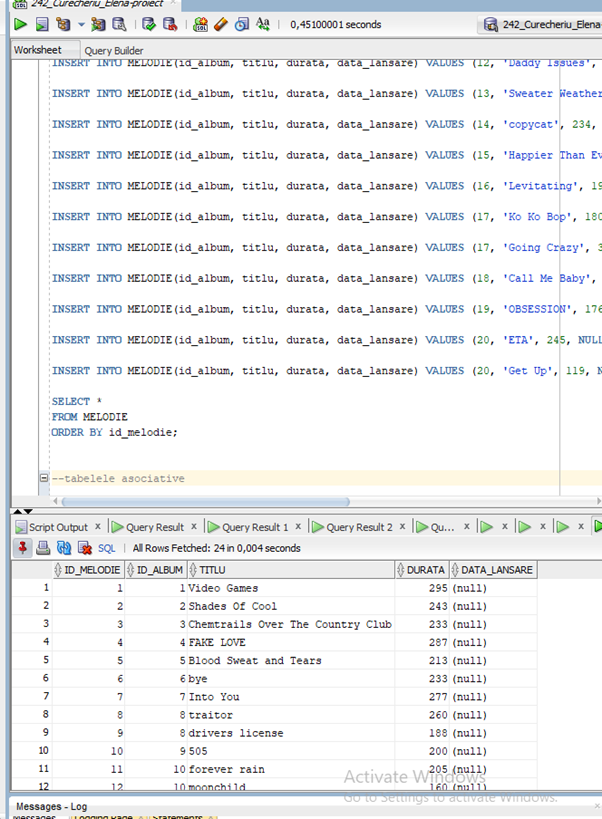
INSERT INTO MELODIE(id\_album, titlu, durata, data\_lansare) VALUES (20, 'ETA', 245, NULL);

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

SELECT \*

FROM MELODIE

ORDER BY id\_melodie;



--GEN\_ALBUM

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);

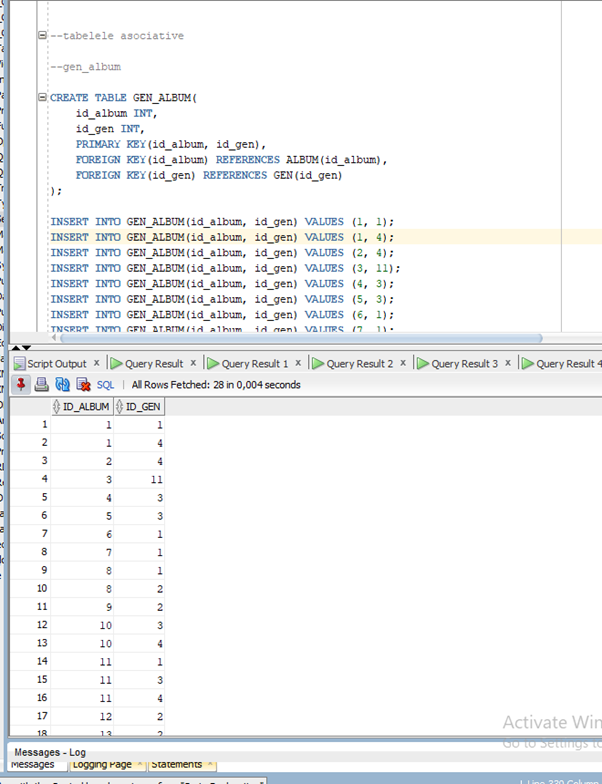
INSERT INTO GEN\_ALBUM(id\_album, id\_gen) VALUES (20, 4);

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

SELECT \*

FROM GEN\_ALBUM

ORDER BY id\_album;



--GEN\_MELODIE

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);

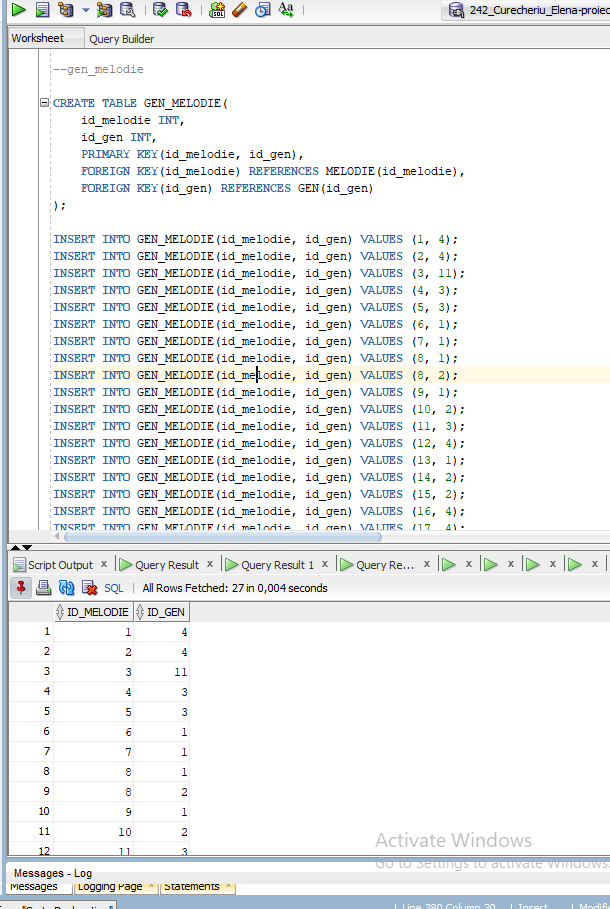
INSERT INTO GEN\_MELODIE(id\_melodie, id\_gen) VALUES (24, 4);

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

SELECT \*

FROM GEN\_MELODIE

ORDER BY id\_melodie;



--MELODIE\_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 (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);

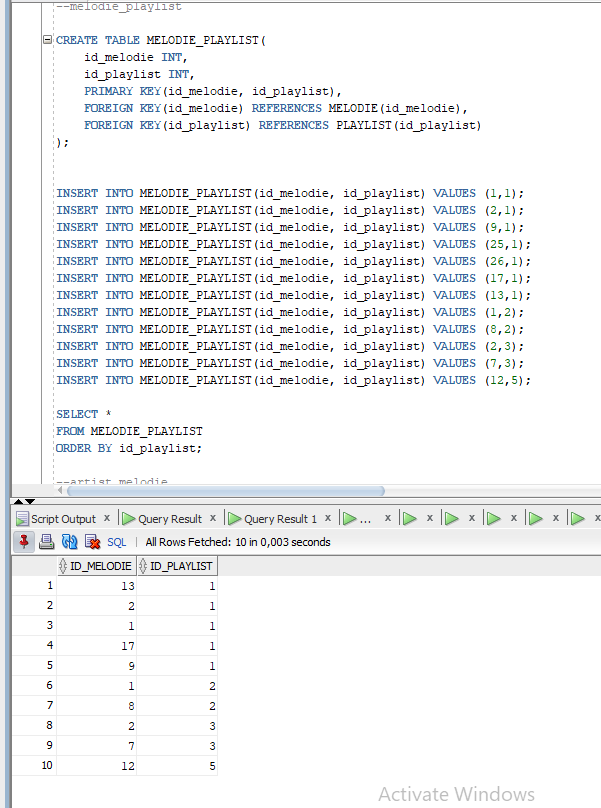
INSERT INTO MELODIE\_PLAYLIST(id\_melodie, id\_playlist) VALUES (7,3);

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

SELECT \*

FROM MELODIE\_PLAYLIST

ORDER BY id\_playlist;



--ARTIST\_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);

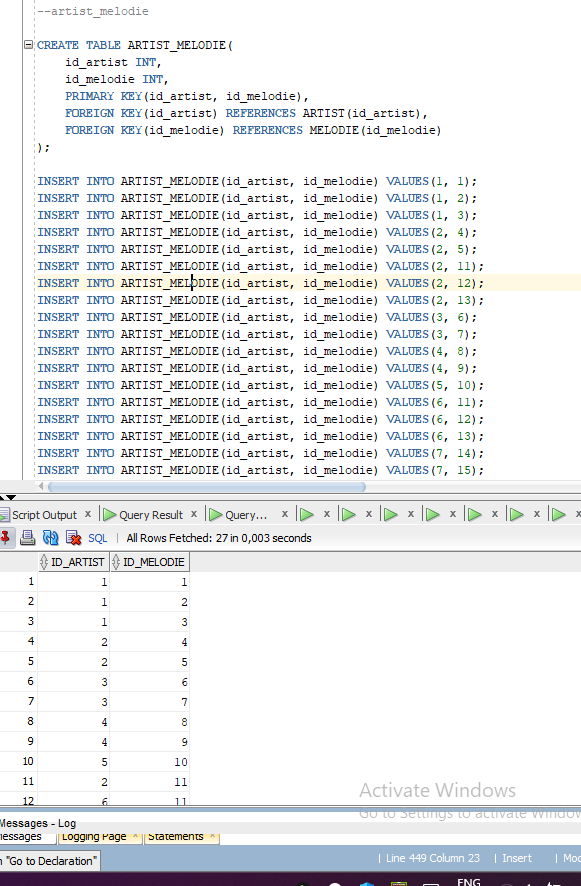
INSERT INTO ARTIST\_MELODIE(id\_artist, id\_melodie) VALUES(11, 23);

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

SELECT \*

FROM ARTIST\_MELODIE

ORDER BY id\_melodie;



# Cerinta 6: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

Pentru un utilizator dat, sa se afiseze tipul de abonament pe care il are, playlist-urile sale, iar pentru fiecare playlist cate un mesaj specific: daca nu are nicio melodie, sa se afiseze „Nu avem melodii in acest playlist”, in caz contrar sa se afiseze numarul de melodii din playlist, titlul fiecarei melodii si durata acesteia.

-- Cerinta 6:

-- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram

-- stocat independent care să utilizeze toate cele 3 tipuri de colecții studiate. Apelați subprogramul.

-- Pentru un utilizator dat, sa se afiseze tipul de abonament pe care il are,

-- playlist-urile sale, iar pentru fiecare playlist cate un mesaj specific: daca nu are nicio melodie,

-- sa se afiseze „Nu avem melodii in acest playlist”, in caz contrar sa se afiseze numarul de melodii din playlist,

-- titlul fiecarei melodii si durata acesteia.

create or replace procedure cerinta\_6 (

v\_utilizator in utilizator.nume%type

)

is

v\_tip\_abonament abonament.tip\_abonament%type;

--tablou indexat care retine numele playlist-urilor

type playlisturi is table of playlist.nume%type index by pls\_integer;

v\_playlisturi playlisturi;

--tablou imbricat care retine numele melodiilor

type melodii is table of melodie.titlu%type;

v\_melodii melodii;

--varray care reprezinta un vector caracteristic care indica daca un

--playlist contine sau nu melodii

type v\_caracteristic is varray(256) of boolean;

are\_melodii v\_caracteristic := v\_caracteristic();

aux number;

begin

dbms\_output.put\_line('Nume utilizator: ' || v\_utilizator);

select tip\_abonament

into v\_tip\_abonament

from utilizator

join abonament on utilizator.id\_abonament = abonament.id\_abonament

where utilizator.nume = v\_utilizator;

if(v\_tip\_abonament is not null) then

dbms\_output.put\_line('Tip abonament: ' || v\_tip\_abonament);

else

dbms\_output.put\_line('Utilizatorul nu exista.');

return;

end if;

select nume

bulk collect into v\_playlisturi

from playlist

where id\_utilizator = (select id\_utilizator

from utilizator

where nume = v\_utilizator);

are\_melodii.extend(v\_playlisturi.count);

for i in 1..v\_playlisturi.count loop --initializam vectorul caracteristic

are\_melodii(i) := false;

end loop;

select titlu

bulk collect into v\_melodii

from melodie

join melodie\_playlist on melodie.id\_melodie = melodie\_playlist.id\_melodie

where id\_playlist in (select id\_playlist

from playlist

where id\_utilizator = (select id\_utilizator

from utilizator

where nume = v\_utilizator));

for i in 1..v\_melodii.count loop

for j in 1..v\_playlisturi.count loop

select count(\*)

into aux

from melodie\_playlist

where id\_melodie = (select id\_melodie

from melodie

where titlu = v\_melodii(i))

and id\_playlist = (select id\_playlist

from playlist

where nume = v\_playlisturi(j));

if aux > 0 then

are\_melodii(j) := true;

end if;

end loop;

end loop;

for i in 1..v\_playlisturi.count loop

dbms\_output.put\_line('Playlist: ' || v\_playlisturi(i));

if are\_melodii(i) = false then

dbms\_output.put\_line('Nu avem melodii in acest playlist.');

else

dbms\_output.put\_line('Numar melodii: ' || v\_melodii.count);

for j in 1..v\_melodii.count loop

dbms\_output.put\_line('Melodie: ' || v\_melodii(j));

select durata

into aux

from melodie

where titlu = v\_melodii(j);

dbms\_output.put\_line('Durata: ' || floor(aux/60) || 'm ' || aux mod 60 || 's');

end loop;

end if;

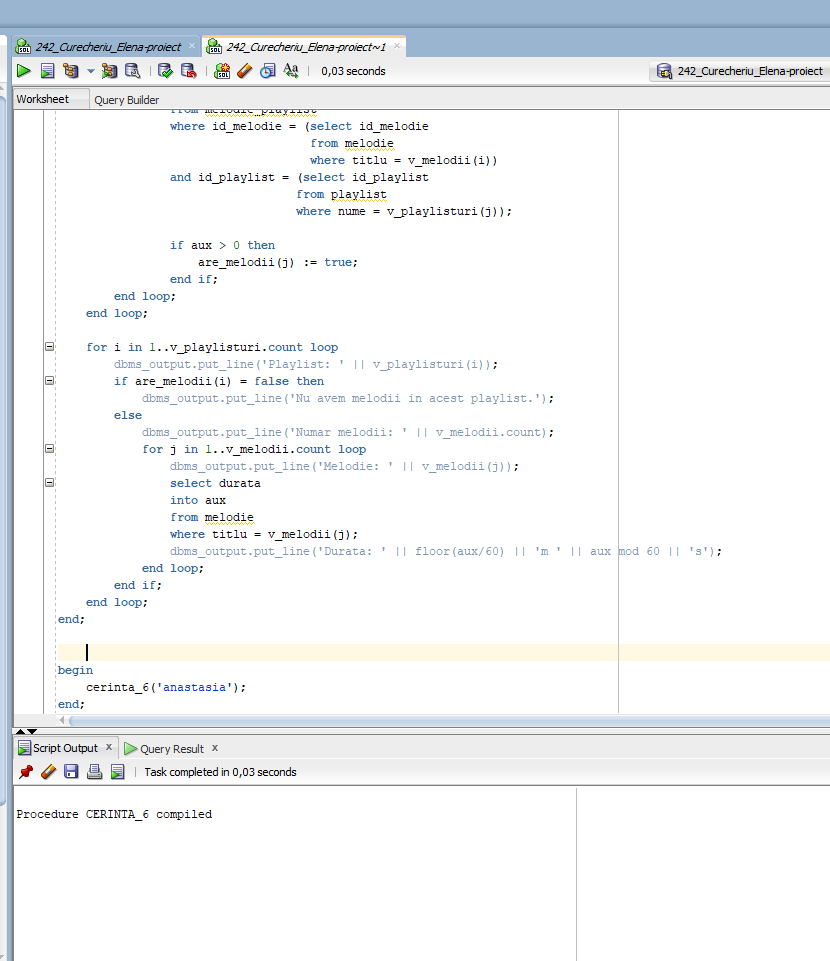
end loop;

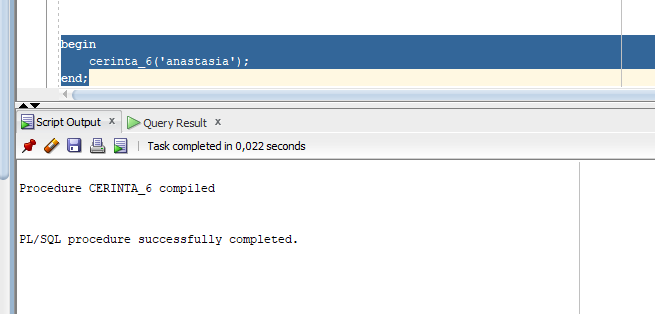
end;

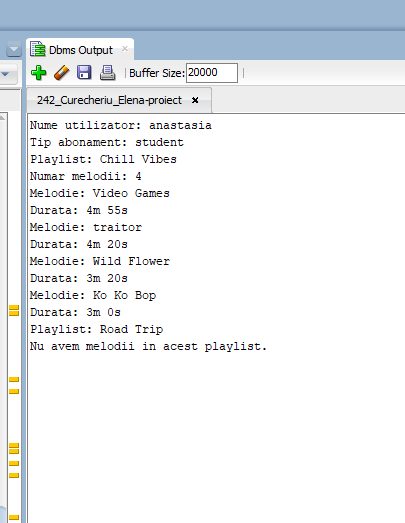
begin

cerinta\_6('anastasia');

end;







# Cerinta 7: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

Sa se gaseasca casa de discuri cu cei mai putini artisti si sa se afiseze artistii care au debutat dupa anul 2000. Daca exista mai multe case de discuri care indeplinesc conditia din cerinta, se va lua in considerare cea care are anul de infiintare mai apropiat de anul curent.

-- Cerinta 7

-- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram

-- stocat independent care să utilizeze 2 tipuri diferite de cursoare studiate, unul dintre

-- acestea fiind cursor parametrizat, dependent de celălalt cursor. Apelați subprogramul.

-- Sa se gaseasca casa de discuri cu cei mai putini artisti si sa se afiseze artistii care au debutat dupa anul 2000.

-- Daca exista mai multe case de discuri care indeplinesc conditia din cerinta,

-- se va lua in considerare cea care are anul de infiintare mai apropiat de anul curent.

create or replace procedure cerinta\_7 as

cursor c1 is

select c.id\_casa\_discuri, c.nume, c.an\_infiintare, count(a.id\_artist) as nr\_artisti

from casa\_discuri c

join artist a on c.id\_casa\_discuri = a.id\_casa\_discuri

group by c.id\_casa\_discuri, c.nume, c.an\_infiintare

order by nr\_artisti, c.an\_infiintare desc;

cursor c2(param casa\_discuri.id\_casa\_discuri%type) is

select nume, an\_debut

from artist a

where a.id\_casa\_discuri = param and a.an\_debut > 2000;

v\_id\_casa\_discuri casa\_discuri.id\_casa\_discuri%type;

v\_nume\_casa\_discuri casa\_discuri.nume%type;

v\_an\_infiintare casa\_discuri.an\_infiintare%type;

v\_nume\_artist artist.nume%type;

v\_an\_debut artist.an\_debut%type;

v\_nr\_artisti number;

begin

open c1;

fetch c1 into v\_id\_casa\_discuri, v\_nume\_casa\_discuri, v\_an\_infiintare, v\_nr\_artisti;

close c1;

dbms\_output.put\_line('Casa de discuri cu cei mai putini artisti: ' || v\_nume\_casa\_discuri);

open c2(v\_id\_casa\_discuri);

loop

fetch c2 into v\_nume\_artist, v\_an\_debut;

exit when c2%notfound;

if (v\_an\_debut is null) then

dbms\_output.put\_line('Nu exista artisti care sa fi debutat dupa anul 2000.');

else

dbms\_output.put\_line('Artist: ' || v\_nume\_artist || '; An debut: ' || v\_an\_debut);

end if;

end loop;

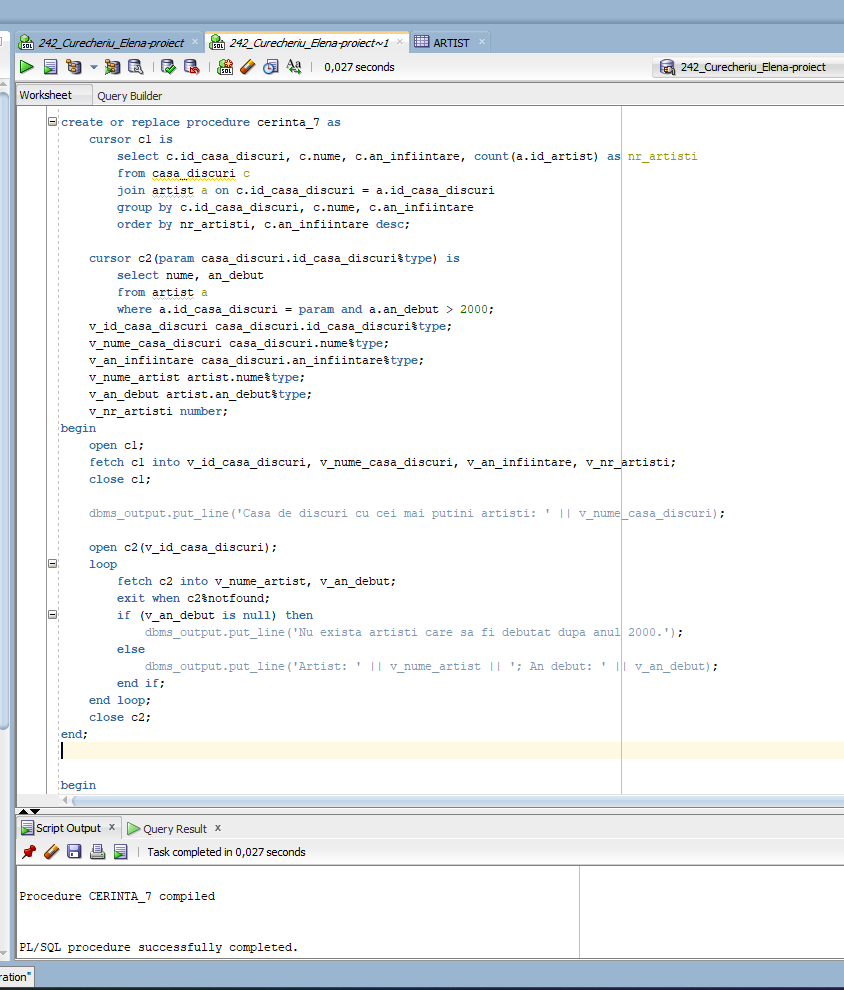
close c2;

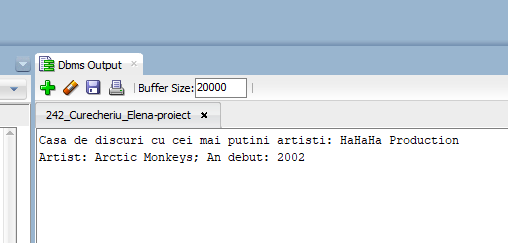
end;

begin

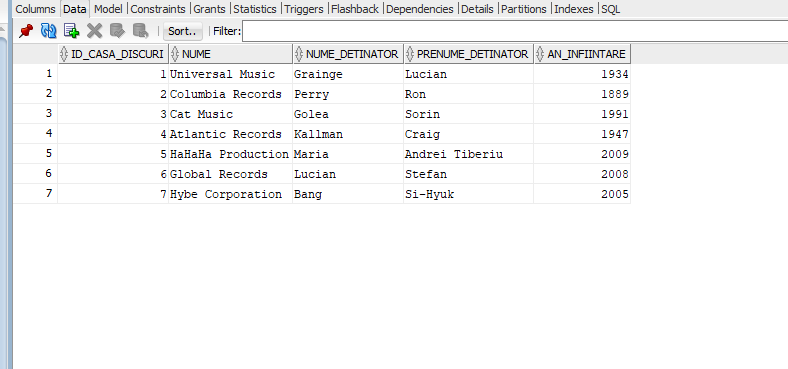
cerinta\_7;

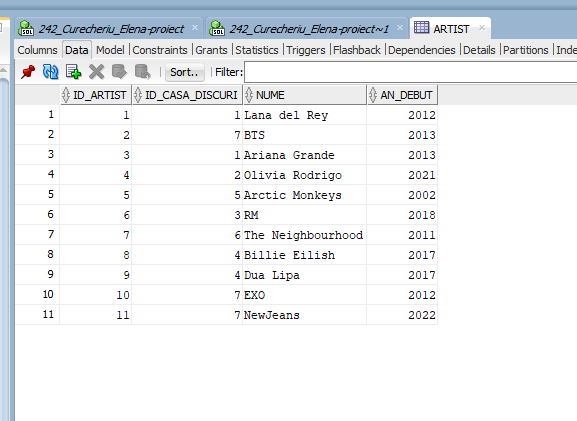
end;





Explicatii:





Exista doua case de discuri cu numarul minim de artisti, dar HaHaHa Production a fost infiintata dupa Hybe Corporation, deci vom afisa artistii care au debutat dupa anul 2000 care au contract cu HaHaHa Production.

# Cerinta 8: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create. Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS. Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

Pentru un utilizator dat, sa se ia playlist-ul cu durata cea mai mare si sa se afiseze cate melodii are. Sa se trateze cazurile in care utilizatorul nu exista in baza de date, utilizatorul nu are playlist-uri, respectiv are doar playlist-uri fara melodii.

-- Cerinta 8

-- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de

-- tip funcție care să utilizeze într-o singură comandă SQL 3 dintre tabelele create.

-- Tratați toate excepțiile care pot apărea, incluzând excepțiile predefinite NO\_DATA\_FOUND și TOO\_MANY\_ROWS.

-- Apelați subprogramul astfel încât să evidențiați toate cazurile tratate.

-- Pentru un utilizator dat, sa se ia playlist-ul cu durata cea mai mare si sa se afiseze cate melodii are.

-- Sa se trateze cazurile in care utilizatorul nu are playlist-uri, respectiv are doar playlist-uri fara melodii.

create or replace function cerinta\_8(param in utilizator.nume%type) return number is

v\_nr\_melodii number;

v\_nr\_playlisturi number;

no\_user\_found exception;

no\_playlist\_found exception;

no\_song\_found exception;

cursor c1 is

select nume from utilizator;

v\_nume utilizator.nume%type;

utilizator\_gasit boolean := false;

begin

open c1;

loop

fetch c1 into v\_nume;

exit when c1%notfound;

if v\_nume = param then

utilizator\_gasit := true;

exit;

end if;

end loop;

close c1;

if utilizator\_gasit = false then

raise no\_user\_found;

end if;

select count(p.id\_playlist) into v\_nr\_playlisturi

from utilizator u

join playlist p on u.id\_utilizator = p.id\_utilizator

where u.nume = param;

if v\_nr\_playlisturi = 0 then

raise no\_playlist\_found;

end if;

select count(mp.id\_melodie) into v\_nr\_melodii

from utilizator u

join playlist p on u.id\_utilizator = p.id\_utilizator

join melodie\_playlist mp on p.id\_playlist = mp.id\_playlist

join melodie m on mp.id\_melodie = m.id\_melodie

where u.nume = param

and p.id\_playlist = (

select id\_playlist

from (

select mp.id\_playlist, sum(m.durata) as durata\_totala

from playlist p1

join melodie\_playlist mp on p1.id\_playlist = mp.id\_playlist

join melodie m on mp.id\_melodie = m.id\_melodie

group by mp.id\_playlist

order by durata\_totala desc

fetch first 1 rows only

)

);

if v\_nr\_melodii = 0 then

raise no\_song\_found;

end if;

return v\_nr\_melodii;

exception

when no\_data\_found then

dbms\_output.put\_line('Nu s-a gasit nicio inregistrare');

return 0;

when too\_many\_rows then

dbms\_output.put\_line('Prea multe inregistrari');

return 0;

when no\_user\_found then

dbms\_output.put\_line('Utilizatorul nu exista');

return -1;

when no\_playlist\_found then

dbms\_output.put\_line('Utilizatorul nu are playlist-uri');

return 0;

when no\_song\_found then

dbms\_output.put\_line('Utilizatorul are playlist-uri, dar nu are melodii');

return 0;

end;

--cazul functional

begin

dbms\_output.put\_line(cerinta\_8('elenacurecheriu'));

end;

--cazul "utilizator inexistent"

begin

dbms\_output.put\_line(cerinta\_8('utilizatorinexistent'));

end;

--cazul "niciun playlist"

begin

dbms\_output.put\_line(cerinta\_8('costinel777'));

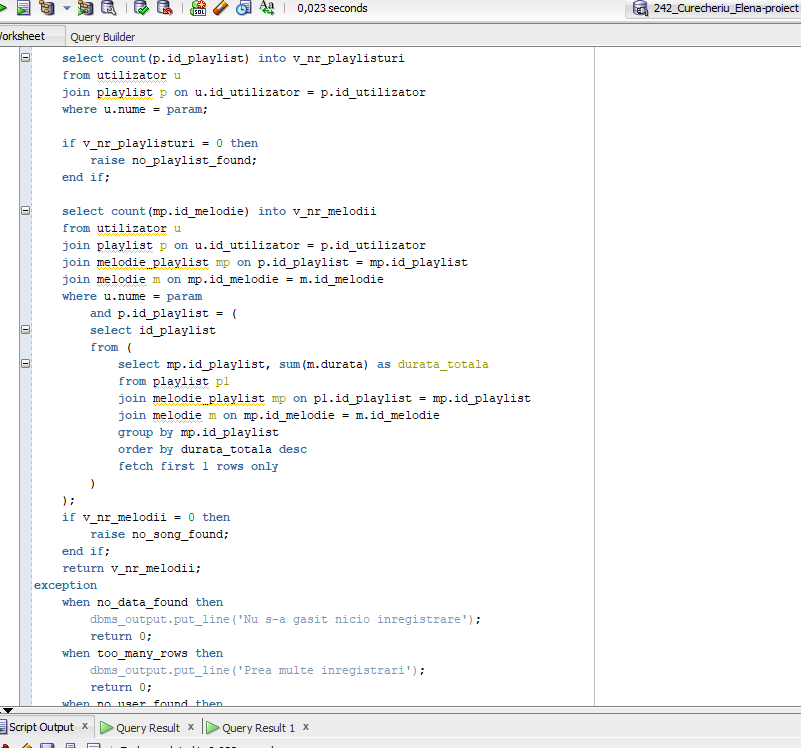
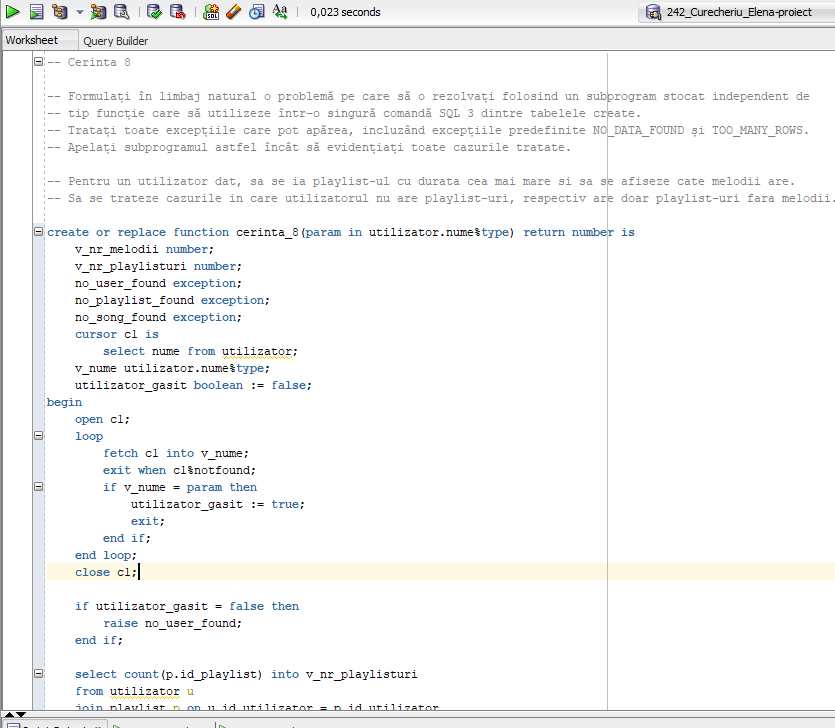
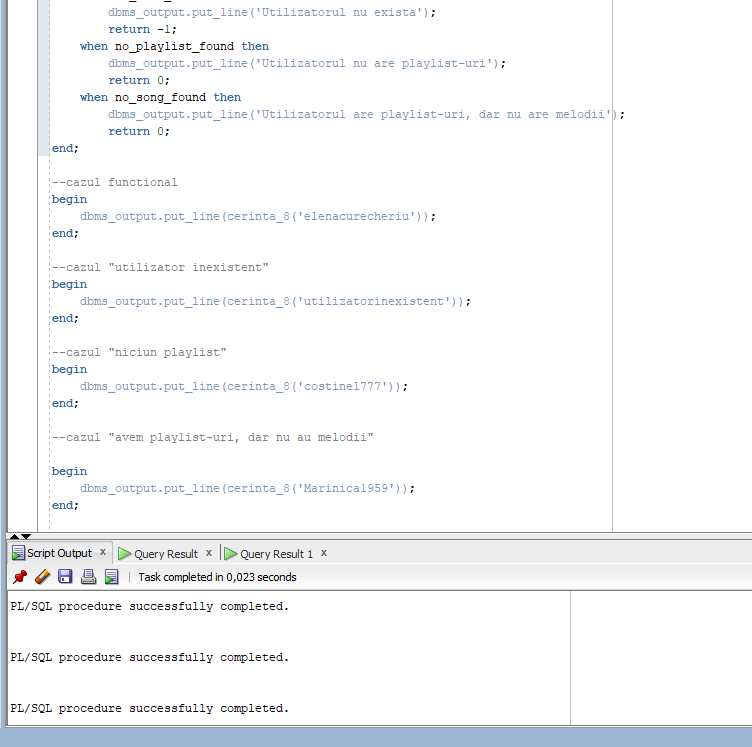
end;

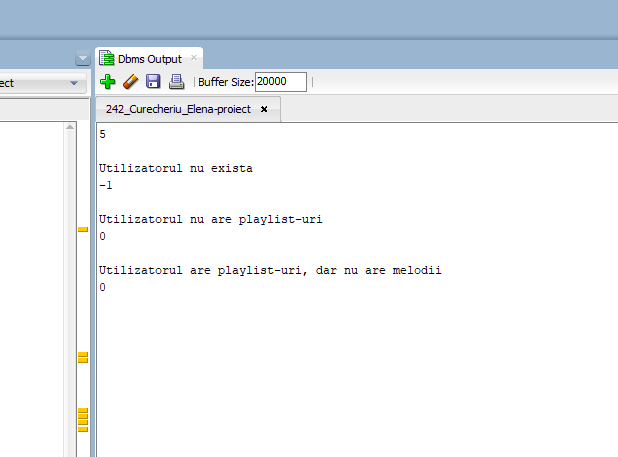
--cazul "avem playlist-uri, dar nu au melodii"

begin

dbms\_output.put\_line(cerinta\_8('Marinica1959'));

end;



# Cerinta 9: Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create. Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem. Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

Pentru un playlist dat, sa se afiseze toate genurile melodiilor care se afla in playlist si sa se contorizeze frecventa acestora. In plus, sa se elimine din playlist melodiile care apartin mai multor genuri muzicale.

-- Cerinta 9

-- Formulați în limbaj natural o problemă pe care să o rezolvați folosind un subprogram stocat independent de tip

-- procedură care să aibă minim 2 parametri și să utilizeze într-o singură comandă SQL 5 dintre tabelele create.

-- Definiți minim 2 excepții proprii, altele decât cele predefinite la nivel de sistem.

-- Apelați subprogramul astfel încât să evidențiați toate cazurile definite și tratate.

-- Pentru un playlist dat, sa se afiseze toate genurile melodiilor care se afla

-- in playlist si sa se contorizeze frecventa acestora. In plus,

-- sa se elimine din playlist melodiile care apartin mai multor genuri muzicale.

create or replace procedure cerinta\_9 (param in playlist.nume%type) is

no\_playlist\_found exception;

empty\_playlist exception;

songs\_not\_in\_multiple\_genres exception;

v\_id\_playlist playlist.id\_playlist%type;

v\_nr\_melodii number;

v\_nr\_melodii\_gen number;

cursor c1 is

select id\_playlist

from playlist

where param = playlist.nume;

cursor c2 is

select count(\*) as nr\_melodii

from melodie\_playlist

where v\_id\_playlist = id\_playlist;

cursor c3 is

select count(\*)

from

(

select mp.id\_melodie

from melodie\_playlist mp

join gen\_melodie gm on mp.id\_melodie = gm.id\_melodie

where v\_id\_playlist = mp.id\_playlist

group by mp.id\_melodie

having count(gm.id\_gen) > 1

);

type t\_melodii\_eliminate is table of varchar2(256);

melodii\_eliminate t\_melodii\_eliminate;

begin

open c1;

fetch c1 into v\_id\_playlist;

if c1%notfound then

raise no\_playlist\_found;

elsif c1%rowcount > 1 then

RAISE\_APPLICATION\_ERROR(-20001, 'TOO\_MANY\_ROWS: Exista mai multe playlist-uri cu acelasi nume.');

end if;

close c1;

open c2;

fetch c2 into v\_nr\_melodii;

if v\_nr\_melodii = 0 then

raise empty\_playlist;

end if;

close c2;

open c3;

fetch c3 into v\_nr\_melodii\_gen;

if v\_nr\_melodii\_gen = 0 then

raise songs\_not\_in\_multiple\_genres;

end if;

close c3;

select distinct m.titlu

bulk collect into melodii\_eliminate

from melodie m

join melodie\_playlist mp on m.id\_melodie = mp.id\_melodie

join gen\_melodie gm on m.id\_melodie = gm.id\_melodie

join gen g on gm.id\_gen = g.id\_gen

join playlist p on mp.id\_playlist = p.id\_playlist

where mp.id\_melodie in(

select mp.id\_melodie

from melodie\_playlist mp

join gen\_melodie gm on mp.id\_melodie = gm.id\_melodie

where mp.id\_playlist = v\_id\_playlist

group by mp.id\_melodie

having count(distinct gm.id\_gen) > 1

) and mp.id\_playlist = v\_id\_playlist;

delete from melodie\_playlist

where id\_melodie in(

select mp.id\_melodie

from melodie\_playlist mp

join gen\_melodie gm on mp.id\_melodie = gm.id\_melodie

where mp.id\_playlist = v\_id\_playlist

group by mp.id\_melodie

having count(distinct gm.id\_gen) > 1

) and id\_playlist = v\_id\_playlist;

dbms\_output.put\_line('Melodiile eliminate din playlist sunt:');

for i in 1..melodii\_eliminate.count loop

dbms\_output.put\_line(melodii\_eliminate(i));

end loop;

dbms\_output.put\_line('Genurile si frecventa melodiilor din playlist:');

for c4 in(

select g.nume, count(mp.id\_melodie) as cnt

from melodie\_playlist mp

join gen\_melodie gm on mp.id\_melodie = gm.id\_melodie

join gen g on gm.id\_gen = g.id\_gen

where mp.id\_playlist = v\_id\_playlist

group by g.nume

order by cnt desc

)

loop

dbms\_output.put\_line('Gen: ' || c4.nume || ', numar de aparitii: ' || c4.cnt);

end loop;

dbms\_output.put\_line('Melodiile ramase in playlist:');

for c5 in(

select m.titlu, g.nume

from melodie\_playlist mp

join melodie m on mp.id\_melodie = m.id\_melodie

join gen\_melodie gm on m.id\_melodie = gm.id\_melodie

join gen g on gm.id\_gen = g.id\_gen

where mp.id\_playlist = v\_id\_playlist

)

loop

dbms\_output.put\_line('Melodie: ' || c5.titlu || ', gen: ' || c5.nume);

end loop;

exception

when no\_playlist\_found then

dbms\_output.put\_line('Playlistul nu exista');

when empty\_playlist then

dbms\_output.put\_line('Playlistul este gol');

when songs\_not\_in\_multiple\_genres then

dbms\_output.put\_line('Nu exista melodii in playlist care apartin mai multor genuri muzicale');

end;

--cazul 'playlist inexistent'

begin

cerinta\_9('Playlist inexistent');

end;

--cazul 'playlist gol'

begin

cerinta\_9('Road Trip');

end;

--cazul 'nu avem melodii care sa se afle in mai multe genuri muzicale'

begin

cerinta\_9('My favourite songs');

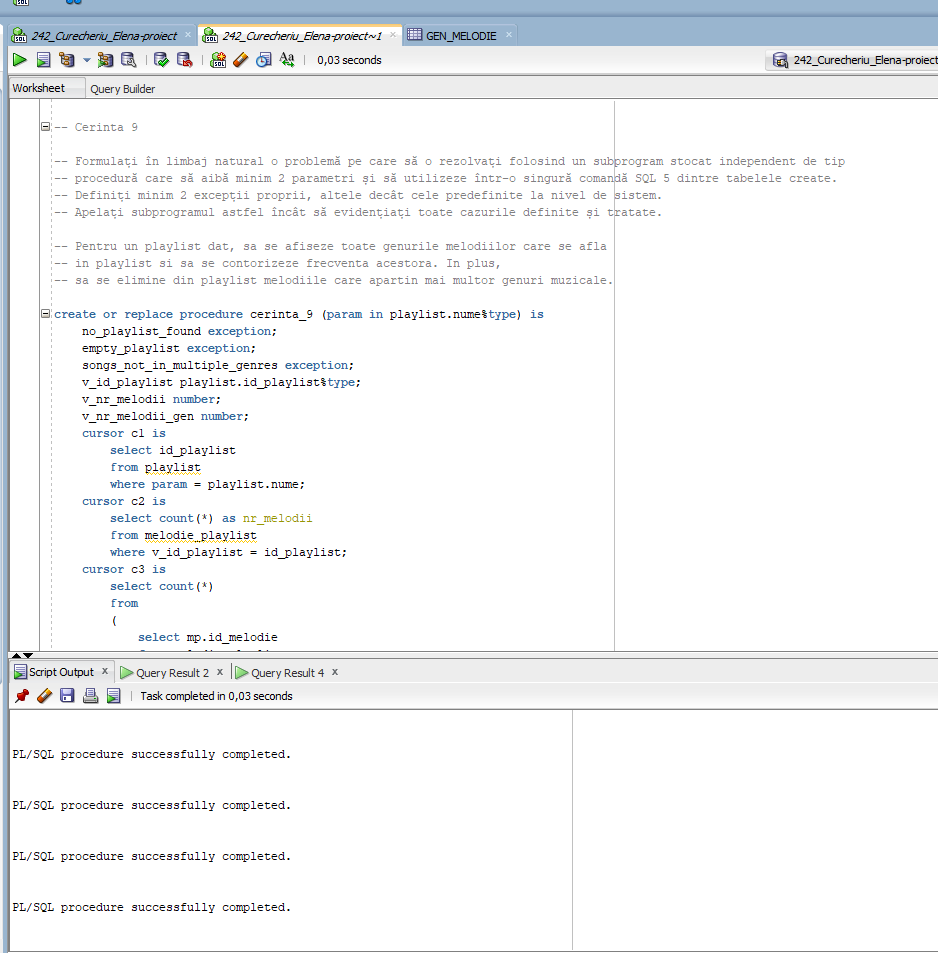
end;

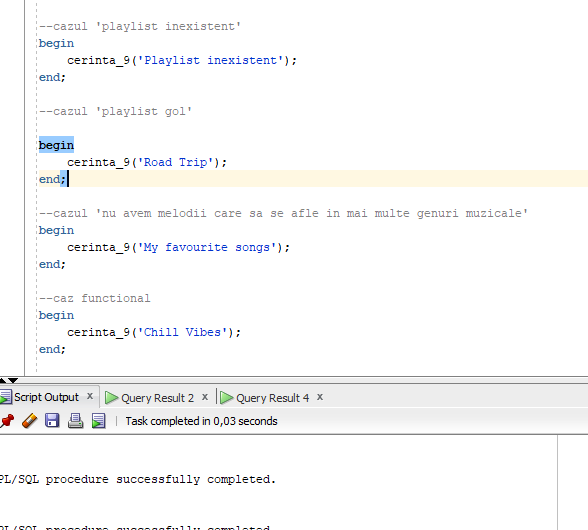
--caz functional

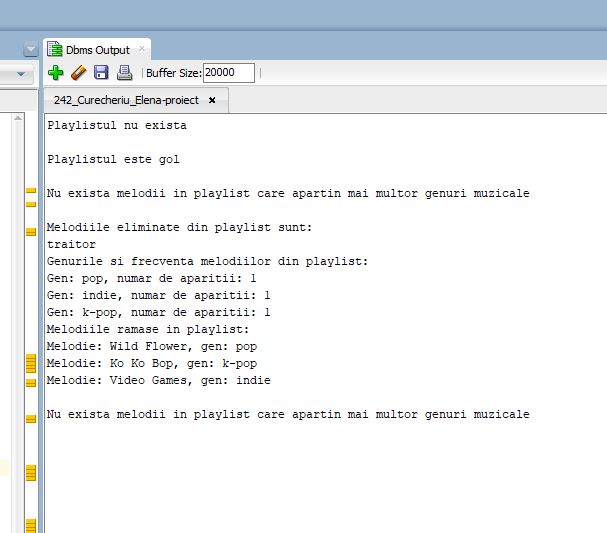
begin

cerinta\_9('Chill Vibes');

end;







--am apelat inca o data cazul favorabil pentru a arata ca melodia a fost eliminata din playlist

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

Trigger care nu permite sa avem in baza de date un album cu data de lansare dupa data curenta.

--Cerinta 10

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

--Trigger care nu permite sa avem in baza de date un album cu data de lansare dupa data curenta

create or replace trigger cerinta\_10

after insert or update on album

declare

cnt number;

begin

select count(\*) into cnt

from album

where data\_lansare > sysdate;

if cnt > 0 then

raise\_application\_error(-20001, 'Nu putem avea un album lansat in viitor!');

end if;

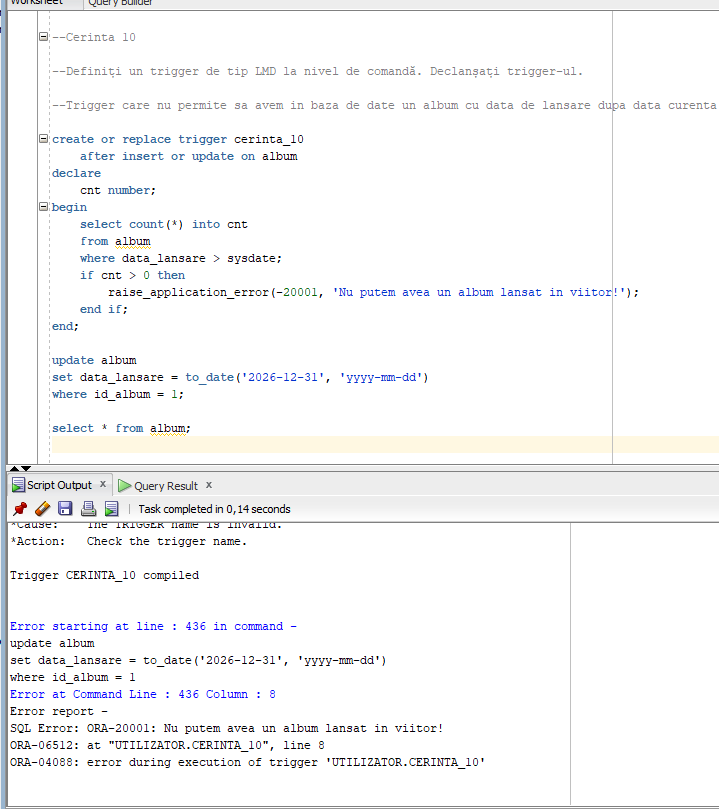
end;

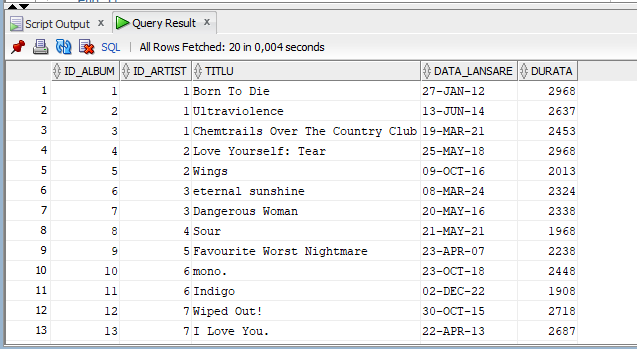
update album

set data\_lansare = to\_date('2026-12-31', 'yyyy-mm-dd')

where id\_album = 1;

select \* from album;

******

******

--albumul cu id-ul 1 nu a fost modificat

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

Trigger care nu ii permite unui utilizator sa aiba mai mult de 5 melodii intr-un playlist:

--Cerinta 11

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

--Trigger care nu ii permite unui utilizator sa aiba mai mult de 5 melodii intr-un playlist

create or replace trigger cerinta\_11

before insert on melodie\_playlist

for each row

declare

nr\_melodii number;

begin

select count(\*) into nr\_melodii

from melodie\_playlist

where id\_playlist = :new.id\_playlist;

if nr\_melodii >= 5 then

raise\_application\_error(-20001, 'Nu puteti adauga mai mult de 5 melodii in playlist');

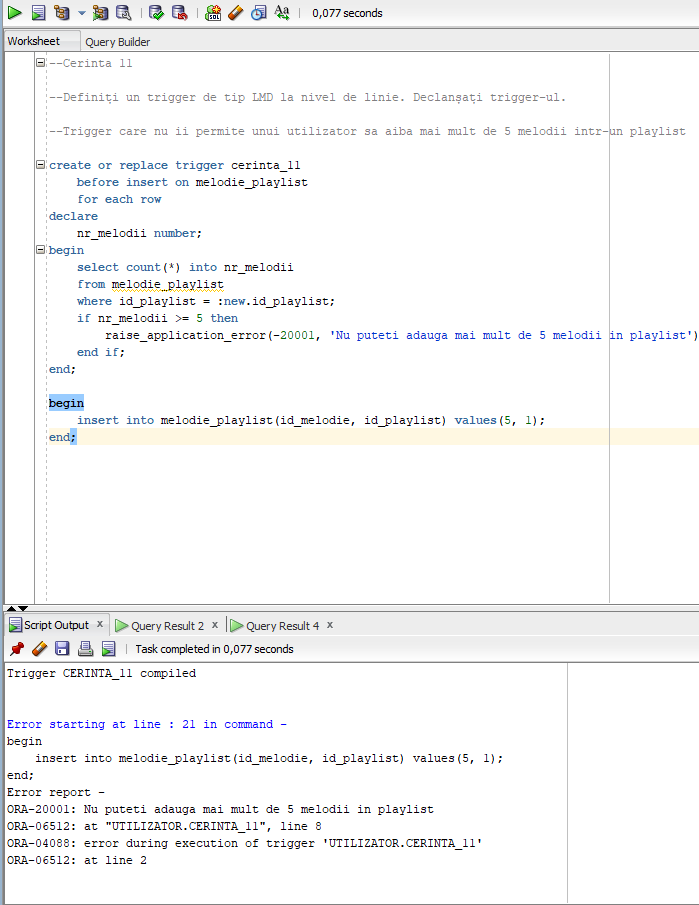
end if;

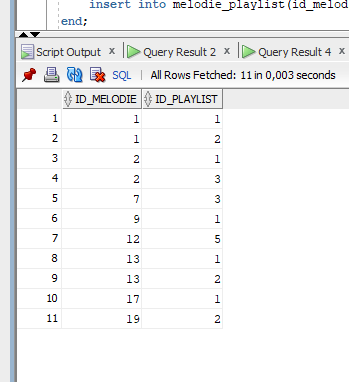
end;

begin

insert into melodie\_playlist(id\_melodie, id\_playlist) values(5, 1);

end;





--playlist-ul cu id-ul 1 deja are asociate 5 melodii, asadar nu o mai putem adauga pe a 6-a

# Cerinta 12: Definiți un trigger de tip LDD. Declanșați trigger-ul.

Trigger care memoreaza actiunile de tip CREATE, ALTER, DROP efectuate asupra bazei de date

--Cerinta 12

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

--Trigger care memoreaza actiunile de tip CREATE, ALTER, DROP efectuate asupra bazei de date

create table audit\_user(

nume\_bd varchar(256),

utilizator varchar(256),

eveniment varchar(256),

nume\_obiect varchar(256),

tip\_obiect varchar(256),

data\_actiune date

);

create or replace trigger cerinta\_12

after create or alter or drop on schema

begin

insert into audit\_user values (sys.database\_name, sys.login\_user, sys.sysevent, sys.dictionary\_obj\_name, sys.dictionary\_obj\_type, sysdate);

end;

create table concert(

artist varchar(256),

locatie varchar(256),

data date,

nr\_locuri number

);

alter table concert

add (pret\_bilet number);

alter table concert

drop column nr\_locuri;

drop table concert;

select \* from audit\_user;

