Calcan Elena-Claudia
343C3

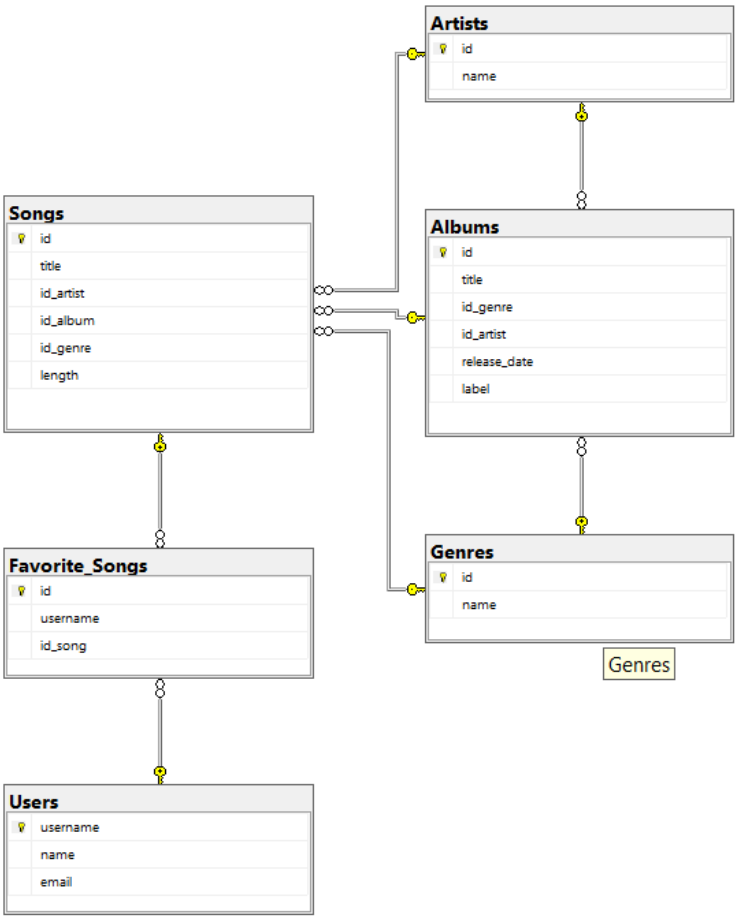
Baza de date cu melodiile preferate a unor users

Synopsys

S-a realizat o baza de date cu melodiile preferate a unor users care folosesc o platforma de streaming audio.

In acest proiect SGBD-ul folosit este Microsoft SQL. Pentru vizualizarea rapoartelor s-a folosit PowerBI.

Diagrama bazei de date



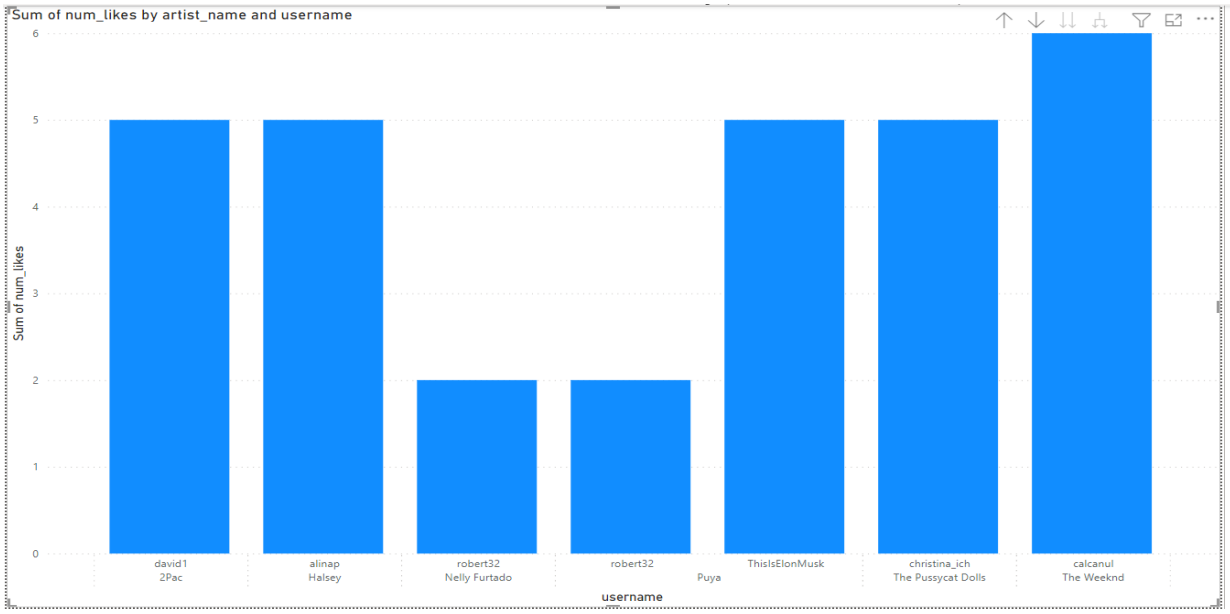
Baza de date este alcatuita din 6 tabele:

1. Users
    - contine informatii despre userii din platforma, cum ar fi: username, numele lor si email
    - constrangeri tabela:
        - primary key pe coloana username
        - unique si not null pe coloana username
2. Artists
    - contine informatii despre artisti, cum ar fi numele lor
    - constrangeri tabela
        - primary key pe coloana id
        - not null pe coloana name
3. Genres
    - contine informatii despre genurile muzicale, cum ar fi numele genului
    - constrangeri tabela
        - primary key pe coloana id
        - unique si not null pe coloana name
4. Albums
    - contine informatii despre albumele artistilor, cum ar fi: titlul albumului, id-ul artistului, id-ul genului muzical caruia ii apartine, data lansarii si label-ul in care a fost inregistrat
    - constrangeri tabela:
        - primary key pe coloana id
        - foreign key si not null pe coloana id_artist ce refera coloana id din tabela Artists
        - foreign key si not null pe coloana id_genre ce refera coloana id din tabela Genres
        - not null pe coloanele title si label
5. Songs
    - contine informatii despre piesele artistilor, cum ar fi: titlul piesei, id-ul artistului, id-ul albumului din car eface parte piesa (acest camp poate fi si null, insemnand ca e single), id-ul genului muzical caruia ii apartine si durata piesei
    - constrangeri tabela:
        - primary key pe coloana id
        - foreign key si not null pe coloana id_artist ce refera coloana id din tabela Artists
        - foreign key pe coloana id_album ce refere coloana id din tabela Albums
        - foreign key si not null pe coloana id_genre ce refera coloana id din tabela Genres
        - not null pe coloanele title si length
6. Favorite_Songs
    - contine asocieri intre users si melodiile lor preferate
    - constrangeri tabela:
        - primary key pe coloana id (de asemenea, aici s-a folosit popretatea identity pentru autoincrementare)
        - foreign key si not null pe coloana username ce refera coloana username din tabela Users
        - foreign key si not null pe coloana id_song ce refera coloana id din tabela Songs

Rapoarte

1. Artistul preferat al fiecarui user, in functie de numarul de piese apreciate

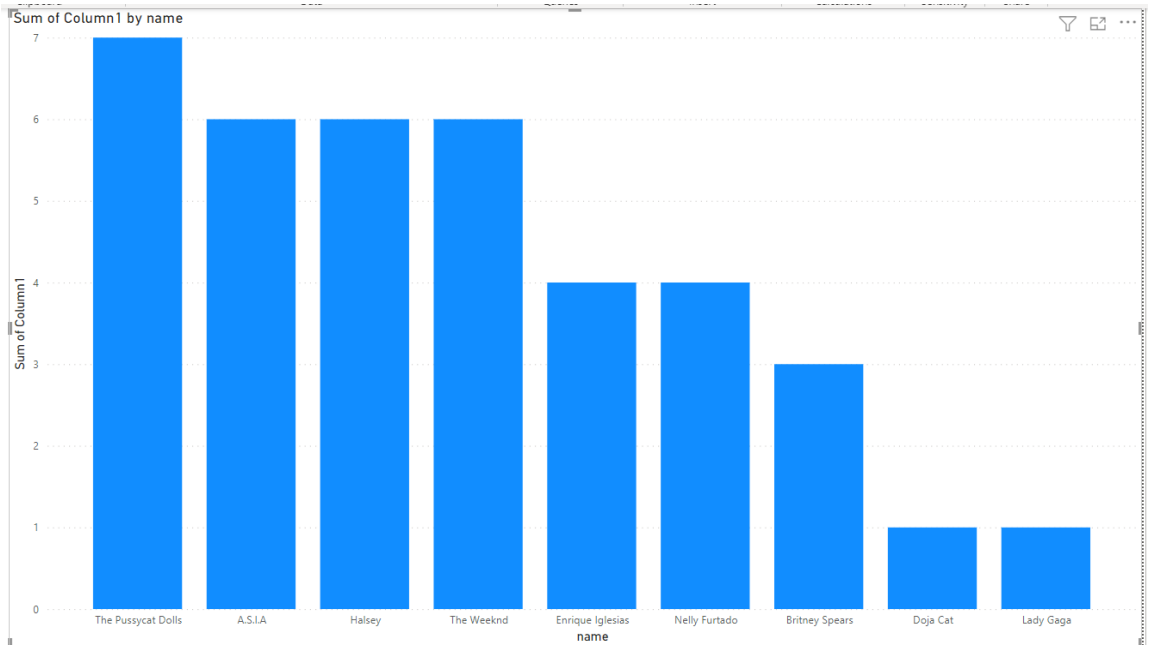
    - pentru fiecare user s-a calculat numarul de piese preferate de la fiecare artist ascultat de el
    - pentru asta s-au facut 2 join-uri; un join intre tabelele Favorite_Songs si Songs si al doilea intre Songs si Artists
    - rezultatul obtinut este stocat intr-o variabila de tip table, pentru o lizibilitate a query-urilor
    - rezultatul final este obtinut prin corelarea dintre tabela obtinuta anterior si o tabela temporara creata in subclauza FROM, care contine cel mai mare numar de piese preferate de la un artist, acesta fiind preferat-ul userului
    - **Fisier sursa:** selectUsersMostFavoritedArtist.sql
    - **Nume procedura:** usersMostFavoritedArtist



sql scripts > selectUsersMostFavoritedArtist.sql > PROCEDURE usersMostFavoritedArtist

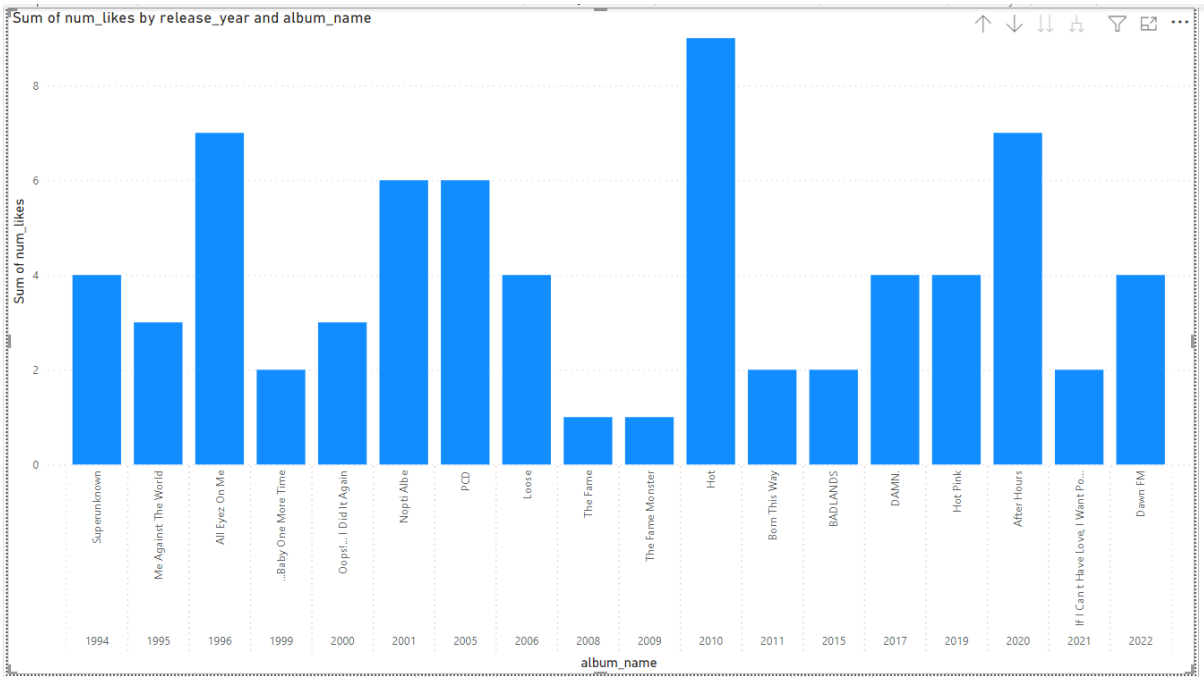
```sql
CREATE PROCEDURE usersMostFavoritedArtist
AS

DECLARE @FavoriteArtists TABLE (
    username varchar(30),
    artist_name varchar(50),
    num_likes int
)

INSERT INTO @FavoriteArtists (username, artist_name, num_likes)
SELECT f.username, a.name, COUNT(s.id) num_likes
FROM Favorite_Songs f
JOIN Songs s on s.id = f.id_song
JOIN Artists a on a.id = s.id_artist
GROUP BY f.username, a.name;

SELECT t.username, t.artist_name, t.num_likes
FROM (SELECT username,  MAX(num_likes) max_likes FROM @FavoriteArtists GROUP BY username) b
JOIN @FavoriteArtists t on t.username = b.username AND t.num_likes = b.max_likes
ORDER BY t.num_likes desc;

RETURN;
```

2. Top cei mai ascultati artisti din cel mai popular gen muzical
    - se determina mai intai cel mai ascultat gen muzical de catre users; pentru fiecare gen muzical se numara cate piese sunt ascultate/preferate de catre users si se alege genul cu cele mai multe piese preferate
    - pentru asta s-a facut si un join intre tabelele Favorite_Songs si Songs
    - rezultatul final este numarul de piese preferate a unui artist de catre users, care sa apartina genului respectiv; pentru asta s-au facut 2 join-uri: unul intre tabelele Favorite_Songs si Songs si celalalt intre Songs si Artists
    - **Fisier sursa:** selectTopArtistsListenedByGenre.sql
    - **Nume procedura:** topListenedArtistsByGenre



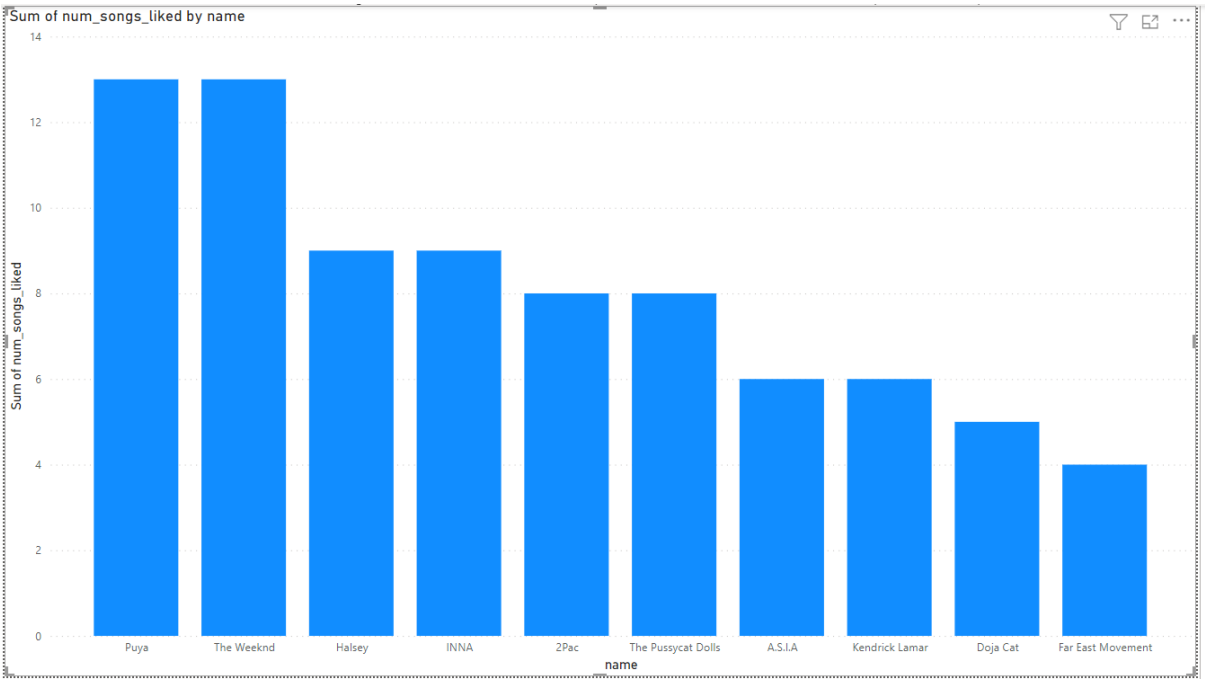

```
sql scripts > selectTopArtistsListenedByGenre.sql > PROCEDURE topListenedArtistsByGenre
1   CREATE PROCEDURE topListenedArtistsByGenre
2   AS
3
4   DECLARE @mostListenedGenre INT
5
6   SELECT TOP(1) @mostListenedGenre=s.id_genre
7   FROM Favorite_Songs f
8   JOIN Songs s ON f.id_song = s.id
9   GROUP BY s.id_genre
10  ORDER BY COUNT(f.id_song) DESC;
11
12  SELECT a.name, COUNT(s.id_artist)
13  FROM Favorite_Songs f
14  JOIN Songs s ON s.id = f.id_song
15  JOIN Artists a ON a.id = s.id_artist
16  WHERE s.id_genre = @mostListenedGenre
17  GROUP BY a.name
18  ORDER BY COUNT(s.id_artist) DESC;
19
20  RETURN;
21
```

3. Cele mai populare albume de-a lungul anilor
    - s-a calculat numarul de piese de pe fiecare album care se gaseste in preferintele userilor, in functie de titlul albumului si anul lansarii
    - pentru asta s-a avut nevoie de 2 join-uri; un join intre tabelele Favorite_Songs si Songs si un altul intre Songs si Albums
    - rezultatul obtinut s-a stocat intr-o variabil de tip tabel pentru a lucra mai usor cu acesta
    - rezultatul final este obtinut prin corelarea dintre tabela obtinuta anterior si o tabela temporara creata in subclauza FROM, care contine albumele cu cele mai preferete piese ale userilor din fiecare an
    - **Fisier sursa:** selectPopularAlbumsThroughoutYears.sql
    - **Nume procedura:** popularAlbumsThroughoutYears




```sql
sql scripts > selectPopularAlbumsThroughoutYears.sql > PROCEDURE popularAlbumsThroughoutYears
CREATE PROCEDURE popularAlbumsThroughoutYears
AS

DECLARE @AlbumsPopularity TABLE (
    album_name VARCHAR(50),
    release_year INT,
    num_likes INT
)

INSERT INTO @AlbumsPopularity(album_name, release_year, num_likes)
SELECT a1.title AS album_name, YEAR(a1.releASe_date) AS release_year, COUNT(f.id_song) num_likes
FROM Favorite_Songs f
JOIN Songs s ON s.id = f.id_song
JOIN Albums a1 ON a1.id = s.id_album
GROUP BY a1.title, YEAR(a1.releASe_date);

SELECT album_name, release_year, num_likes
FROM (SELECT t.release_year AS yr, MAX(t.num_likes) AS max_like FROM @AlbumsPopularity t GROUP BY t.release_year) b
JOIN @AlbumsPopularity on b.yr = releASe_year AND num_likes = b.max_like
ORDER BY release_year;

RETURN;
```

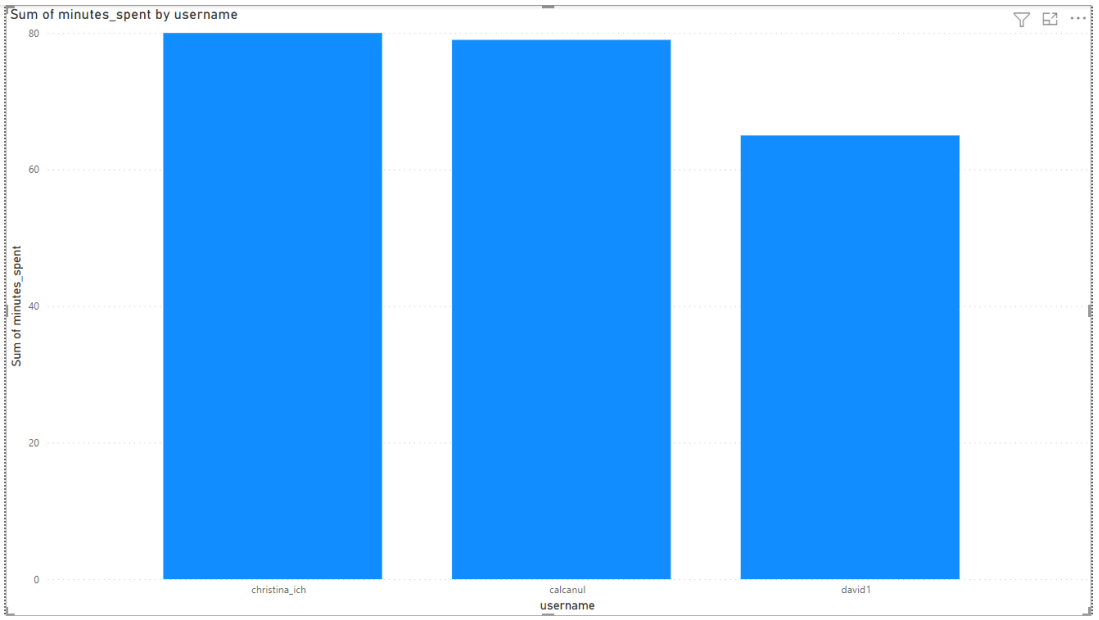
4. Top 10 artisti preferati
    - s-a calculat numarul de piese ale artistilor care se gasesc in preferintele userilor
    - pentru asta a fost nevoie de 2 join-uri: un join intre tabelele Favorite_Songs si Songs si un altul intre tabelele Songs si Artists
    - **Fisier sursa:** selectMostListenedArtist.sql
    - **Nume procedura:** mostListenedArtist




```sql
sql scripts > selectMostListenedArtist.sql > PROCEDURE mostListenedArtist
CREATE PROCEDURE mostListenedArtist
AS

SELECT TOP(10) COUNT(s.id) num_songs_liked, a.name
FROM Favorite_Songs f
JOIN Songs s ON s.id = f.id_song
JOIN Artists a ON s.id_artist = a.id
GROUP BY a.name
ORDER BY COUNT(s.id) desc;

RETURN;
```

5. Top ascultatori

- s-a facut un top al userilor care asculta muzica (in functie de minutele pieselor pe care le are la favorite)
- atat media media cat si suma minutelor a unui user s-a calculat facut join intre Favorite_Songs si Songs
- **Fisier sursa:** selectTopListeners.sql
- **Nume procedura:** listenersAboveAgvTime




```
sql scripts > selectTopListeners.sql > PROCEDURE listenersAboveAgvTime
1  CREATE PROCEDURE listenersAboveAgvTime
2  AS
3      SELECT f.username, SUM(DATEDIFF(MINUTE, '00:00:00', s.length)) minutes_spent
4      FROM Favorite_Songs f
5      JOIN Songs s ON f.id_song = s.id
6      GROUP BY f.username
7      HAVING SUM(DATEDIFF(MINUTE, '00:00:00', s.length)) > (SELECT sum(DATEDIFF(MINUTE, '00:00:00', s1.length)) / count(distinct f1.username)
8                                                            FROM Favorite_Songs f1
9                                                            JOIN Songs s1 ON s1.id = f1.id_song)
10     ORDER BY 2 DESC;
11
12 RETURN;
13
```