Authors: **Group 8**

Eliot Ullmo s221646
Mohamed Charfi
Ahmed Aziz Ben Haj Hmida s221551
Ghalia Bennani s221649
Thibaud Bourgeois s221592

02170 - Database Systems

Group report

AudioAvenue - A music streaming service database

1. Statement of requirements

The music streaming AudioAvenue allows a person to create a user.

Each user has a certain SubscriptionType, a name …. And a Country.

The SubscriptionType represents the different subscription plans offered by the service. This includes information about the plan's name and the price.

The country is represented by a name.

The users are allowed to create and manage playlists.

The playlists that users create to organize and listen to their favorite music. This table could contain information such as the playlist's name, description, and songs

A song has a Name and a genre and is made by an artist.

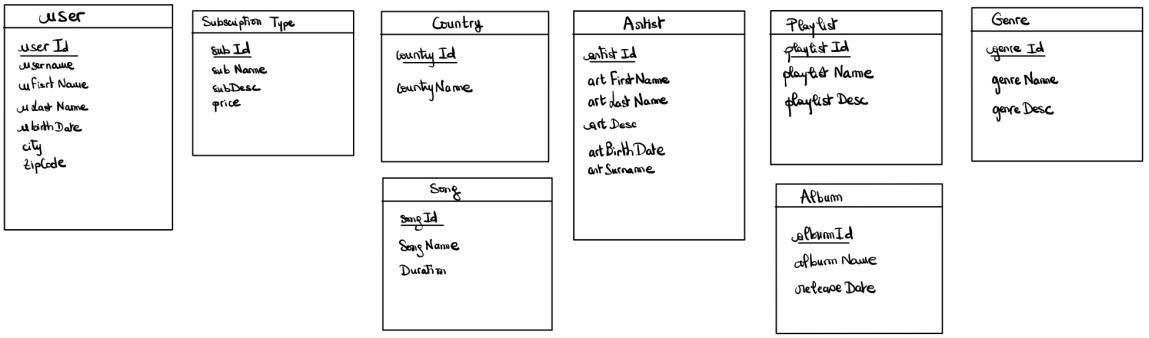
The Artist table likely contains information such as their name, genre, and nationality.

One artist can make an Album.

2. Conceptual design

Entities:

1. User: someone that uses the music platform, it has the following attributes: an id, a username, a first name, a birthdate, the city from where he's from and its zip code.
2. Subscription Type: the type of subscription a user subscribes to. It has the following attributes: an id, a name, a description and a price.
3. Country: the country from which the user comes from, it has the following attributes: an id and a name.
4. Artist: artist that users can listen to on our platform. It has as attribute: an id, a first name, a last name, a description and a birth date.
5. Playlist: music's collection that a user has created. It has attributes: an id, a name and a description.
6. Genre: genre of music users can listen to on our platform. It has the following attributes: an id, a name and a description.
7. Song: song users can listen to on our platform. It has as attribute: an id, a name and a duration.
8. Album: music's collection produced by an artist that users can listen to on our platform. It has as attribute: an id, a name and a release date.

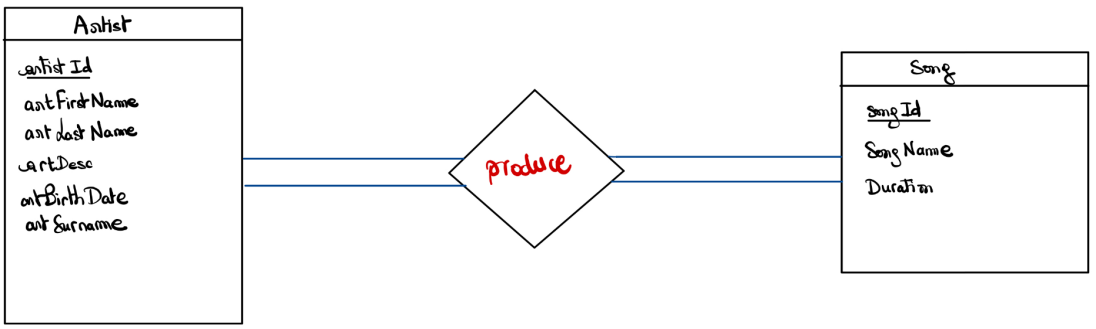


Relationships:

1. The relationship “produce” from artist to song translates the fact that artists produce songs.

Cardinality choice: It’s a **many-to-many relationship from artist to song** since a song can be produced by many artists (featuring) and an artist can produce multiple songs.

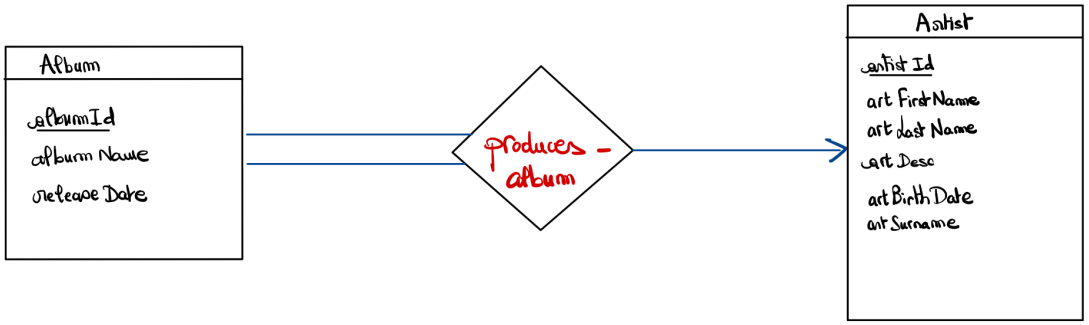
Participation choice: The participation is total in both sides since we assume that if an artist is present in our platform that its songs are too and vice-versa.



2. The relationship “produces_album” from artist to album translates to the fact that artists produce albums.

Cardinality choice: It’s a **one-to-many relationship from artist to album** since an album is produced by exactly one artist but an artist can produce multiple albums.

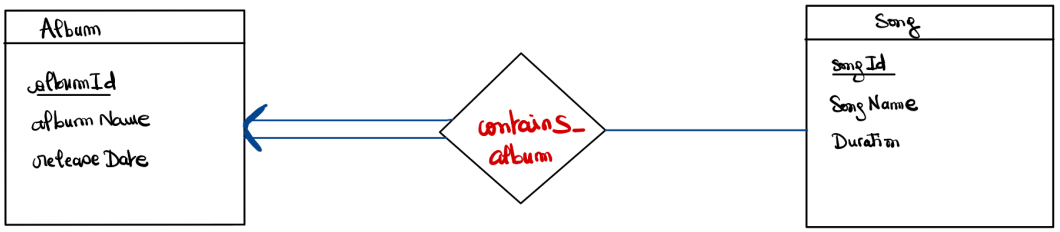
Participation choice: The participation is total in the album side since we assume that if an album is present in our platform, then the artist is also present but an artist can be present in our database without having produced albums.



3. The relationship “contains_album” from album to song translates the fact that songs are contained in an album.

Cardinality choice: It’s a **one-to-many relationship from album to song** since an album contains multiple songs but a song can be part of at most one album.

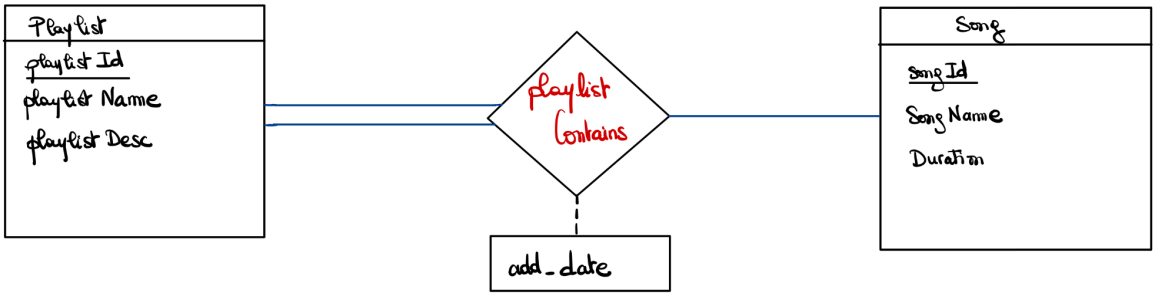
Participation choice: The participation is total in the album side since we assume that if an album is present in our platform, then the song it contains are too but a song can be present in our database without being part of an album.



4. The relationship “playlistContains” from playlist to song translates the fact that songs are contained in a playlist. It has an attribute add_date which is the date when the author of the playlist added a song to it.

Cardinality choice: It’s a **many-to-many relationship from playlist to song** since a playlist contains multiple songs and a song can be part of multiple playlists.

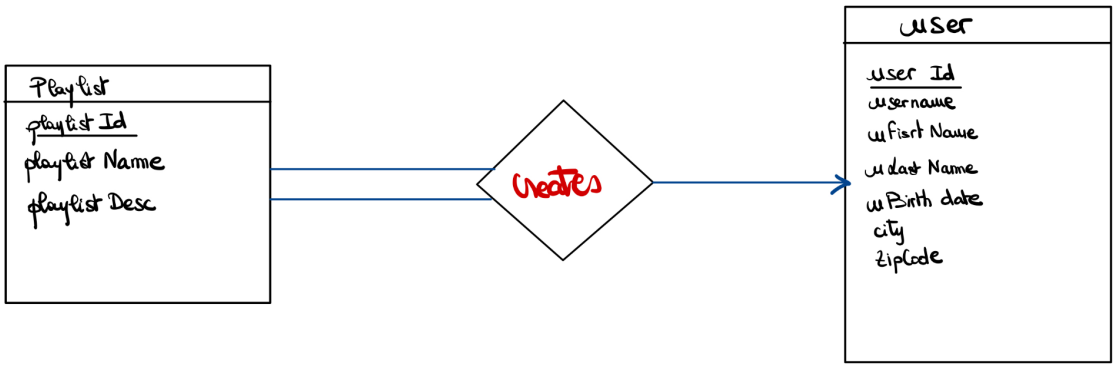
Participation choice: The participation is total in the playlist side since we assume that if a playlist is present in our platform, then the song it contains are too but a song can be present in our database without being part of a playlist.



5. The relationship “creates” from user to playlist translates the fact that a user is a playlist’s author.

Cardinality choice: It’s a **one-to-many relationship from user to playlist** since a user can create multiple playlists but a playlist has exactly one user as author.

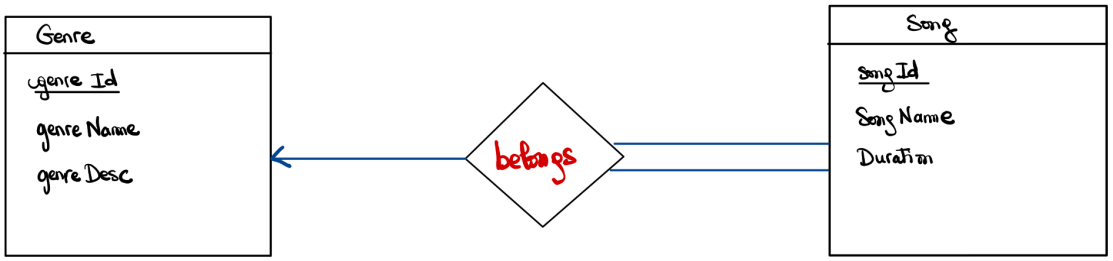
Participation choice: The participation is total in the playlist side since we assume that if a playlist is present in our platform, then the user that it has a profile but a user that uses the platform doesn't systematically create playlists.



6. The relationship “belongs” from song to genre translates to the fact that a song is part of a musical genre.

Cardinality choice: It’s a **many-to-one from song to genre** since multiple songs can be part of the same musical genre but a song cannot have multiple genres.

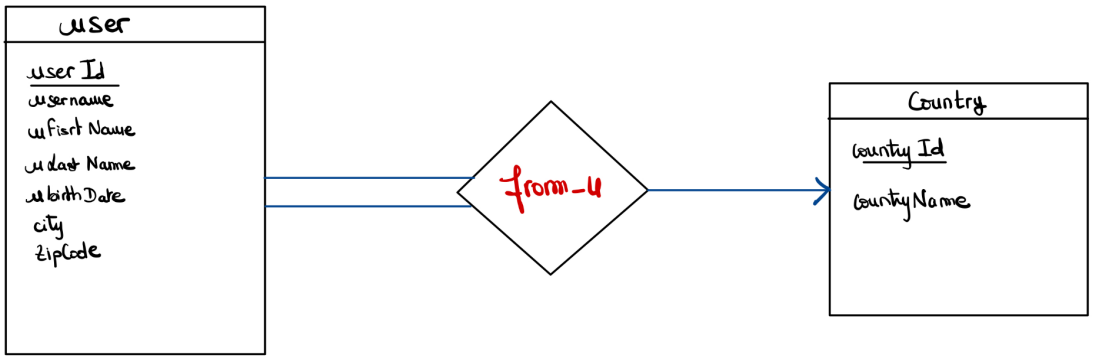
Participation choice: The participation is total in the song part side since each song has a genre but there can be genres in our database that do not have songs.



7. The relationship “from_u” from user to country translates the fact that a user is from a country.

Cardinality choice: It’s a **many-to-one relationship from user to country** since a user can fill in exactly one country in the database but a country can have multiple users associated with it.

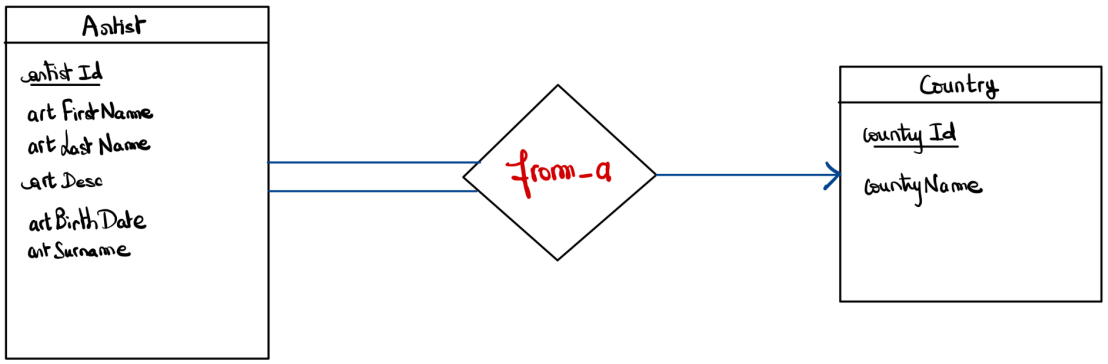
Participation choice: The participation is total on the user side since users must fill in their home country so each user has a country but not all countries have users associated with it.



8. The relationship “from_a” from artist to country translates the fact that an artist is from a country.

Cardinality choice: It’s a **many-to-one relationship from artist to country** since an artist can be from exactly one country in the database but a country can have multiple artists associated with it.

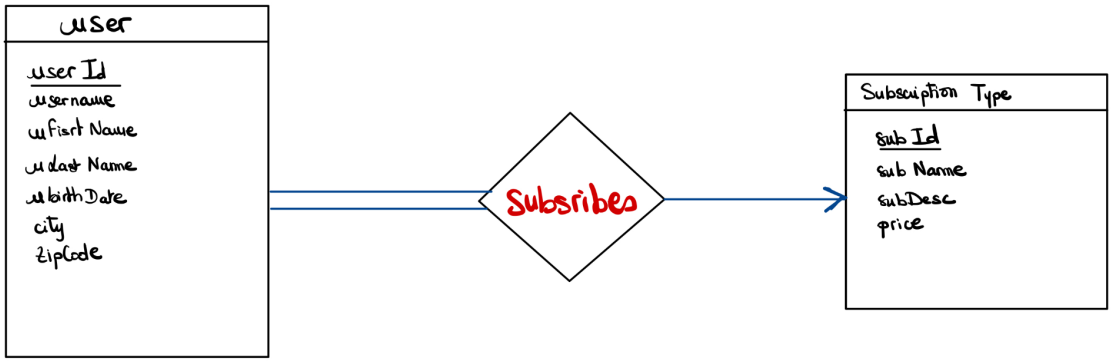
Participation choice: The participation is total on the artist side since artists must fill in their home country so each artist has a country but not all countries have artists associated with it.



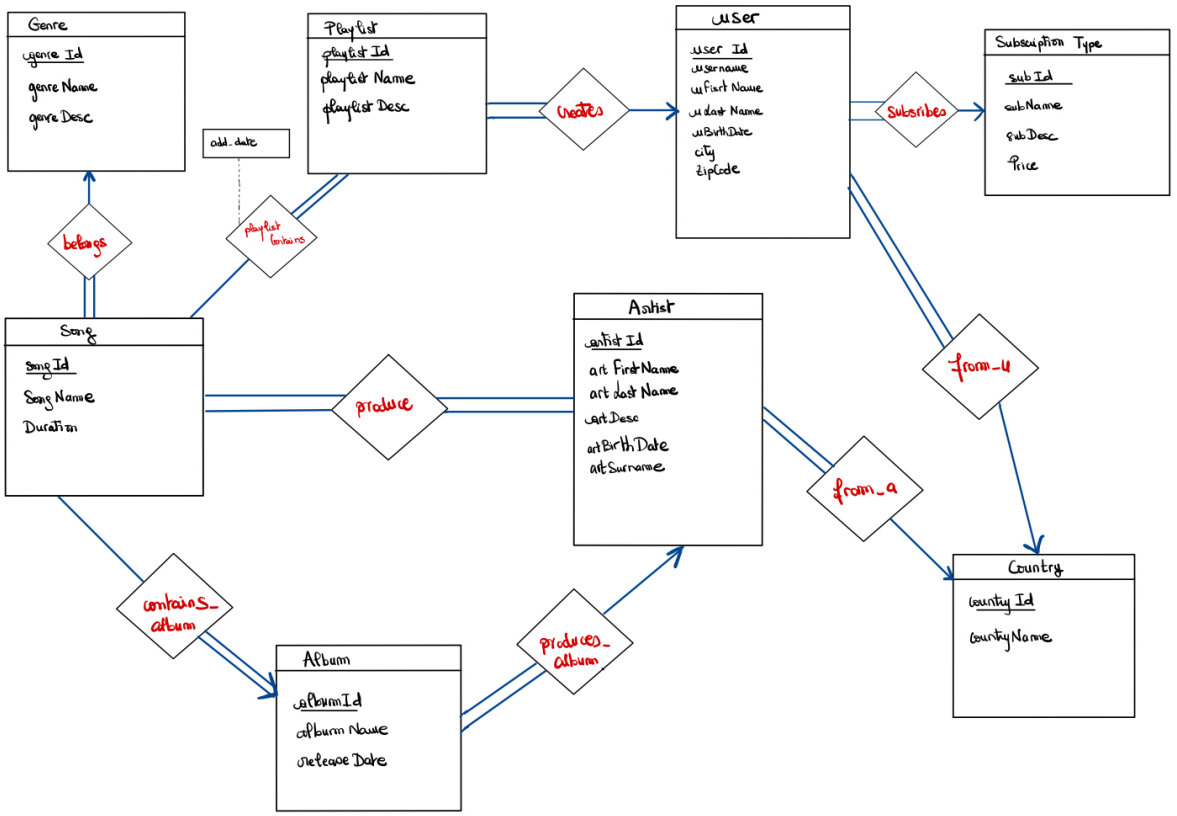
9. The relationship "subscribes" from user to subscriptionType translates the fact that a user needs to subscribe to one subscription type when creating their profile.

Cardinality choice: It's a **many-to-one relationship from user to subscriptionType** since multiple users can have the same subscription type but a user must have exactly one subscription type.

Participation choice: The participation is total on the user side since users must systematically choose their subscription type when creating their account so each user has one but not all subscription types are necessarily chosen.



E-R Diagram:



3. Logical Design

Relation Schemas :

1. **Genre(genreId, genreName, genreDesc)**
2. **Song(songId, songName, Duration, genreId, albumId)** foreign key (genreId) references Genre(genreId), foreign key (albumId) references Album(albumId)

   → The foreign key genreId is built from the many to one relation “belongs” from song to genre.

   → The foreign key albumId is built from the one to many relation “contains_album” from album to song.
3. **Album(albumId, albumName, releaseDate, artistId)** foreign key (artistId) references Artist(artistId)

   → The foreign key artistId is built from the one to many relation “produces_album” from artist to album.

4. **Artist(<u>artistId</u>, artFirstName, artLastName, artDesc, artBirthDate, artSurname, artCountryId)** foreign key (countryId) references Country(countryId)

   → The foreign key countryId is built from the many to one relation “from_a” from artist to country.

5. **Playlist(<u>playlistId</u>, playlistName, playlistDesc, userId)** foreign key (userId) references User(userId)

   → The foreign key is built from the one to many relation “creates” from user to playlist.

6. **User(<u>userId</u>, username, uFirstName, uLastName, uBirthdate, city, zipCode, countryId, subId)** foreign key (countryId) references Country(countryId), foreign key (subId) references SubscriptionType(subId)

   → The foreign key countryId is built from the many to one relation “from_u” from user to country.

   → The foreign key subId is built from the many to one relation “subscribes” from user to subscription type.

7. **Country(<u>countryId</u>, countryName)**

8. **SubscriptionType(<u>subId</u>, subName, subDesc, Price)**

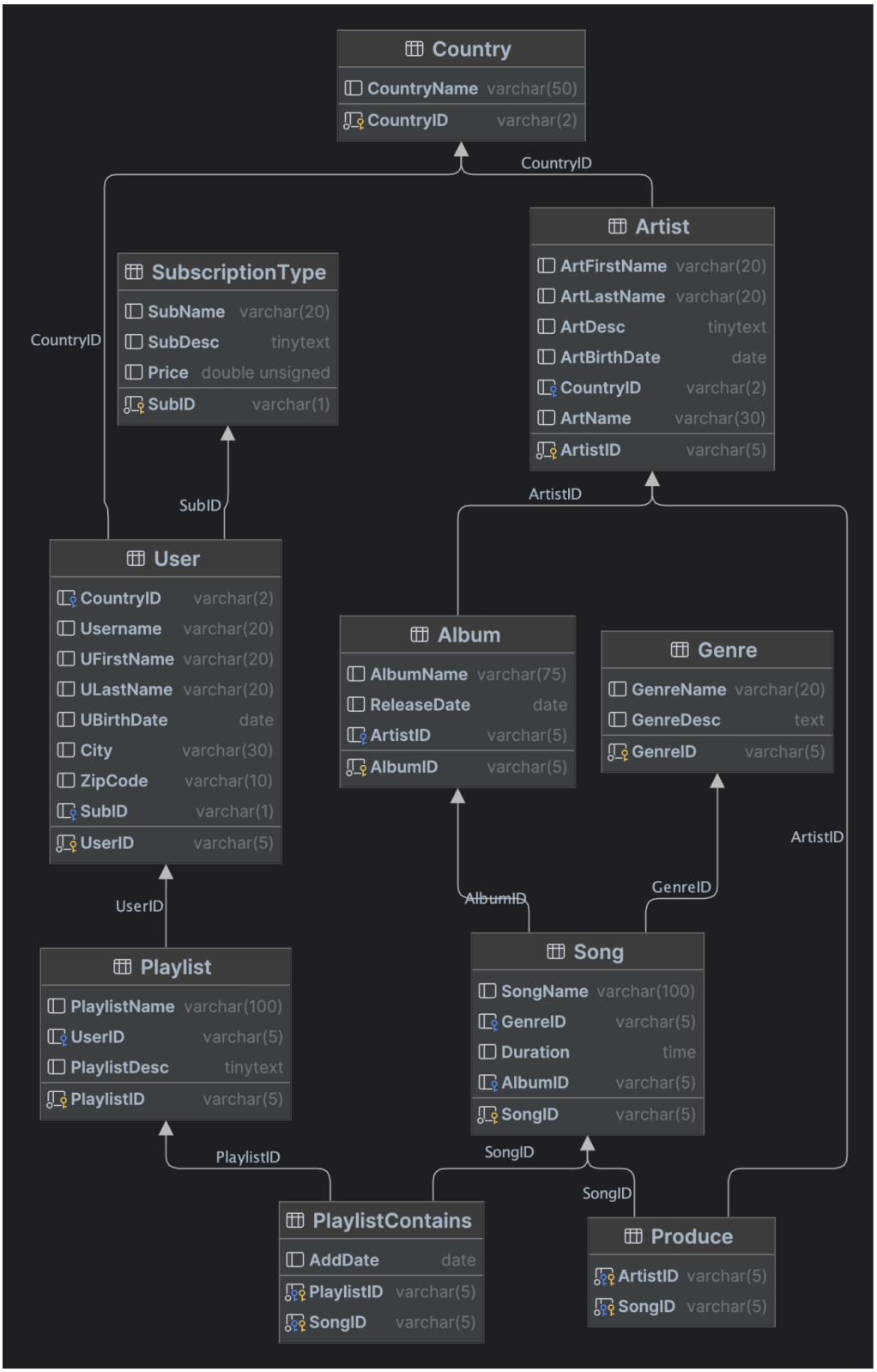
9. **PlaylistContains(<u>playlistId</u>, <u>songId</u>, add_date)** foreign key (playlistId) references Playlist(playlistId), foreign key (songId) references Song(songId)

   → This relation schema is born from the many to many relation ‘playlistContains’ between Playlist and song and the attribute add_date comes from the attribute of the relationship.

10. **Produce(<u>songId</u>, <u>artistId</u>)** foreign key (songId) references Song(songId), foreign key artistId references Artist(artistId)

    → This relation schema is born from the many to many relation ‘produce’ between song and artist.

DataBase Schema Diagram :



4. Implementation

```
CREATE TABLE SubscriptionType
(
   SubID   VARCHAR(1),
   SubName VARCHAR(20),
   SubDesc TINYTEXT,
   Price   DOUBLE UNSIGNED,
   PRIMARY KEY (SubID)
);

CREATE TABLE Country
(
   CountryID   VARCHAR(2),
   CountryName VARCHAR(50),
   PRIMARY KEY (CountryID)
);

CREATE TABLE User
(
   UserID     VARCHAR(5),
   CountryID  VARCHAR(2),
   Username   VARCHAR(20),
   UFirstName VARCHAR(20),
   ULastName  VARCHAR(20),
   UBirthDate DATE,
   City       VARCHAR(30),
   ZipCode    VARCHAR(10),
   SubID      VARCHAR(1),
   PRIMARY KEY (UserID),
   FOREIGN KEY (CountryID) REFERENCES Country (CountryID) ON DELETE SET NULL,
   FOREIGN KEY (SubID) REFERENCES SubscriptionType (SubID) ON DELETE SET NULL
);

CREATE TABLE Artist
(
   ArtistID     VARCHAR(5),
   ArtFirstName VARCHAR(20),
   ArtLastName  VARCHAR(20),
   ArtDesc      TINYTEXT,
   ArtBirthDate DATE,
   CountryID    VARCHAR(2),
   ArtName   VARCHAR(30),
   PRIMARY KEY (ArtistID),
   FOREIGN KEY (CountryID) REFERENCES Country (CountryID) ON DELETE SET NULL
);
```

```
CREATE TABLE Genre
(
   GenreID   VARCHAR(5),
   GenreName VARCHAR(20),
   GenreDesc TEXT,
   PRIMARY KEY (GenreID)
);


CREATE TABLE Album
(
   AlbumID     VARCHAR(5),
   AlbumName   VARCHAR(75),
   ReleaseDate DATE,
   ArtistID    VARCHAR(5),
   PRIMARY KEY (AlbumID),
   FOREIGN KEY (ArtistID) REFERENCES Artist (ArtistID) ON DELETE SET NULL
);

CREATE TABLE Song
(
   SongID   VARCHAR(5),
   SongName VARCHAR(100),
   GenreID  VARCHAR(5),
   Duration TIME,
   AlbumID  VARCHAR(5) NULL ,
   PRIMARY KEY (SongID),
   FOREIGN KEY (GenreID) REFERENCES Genre (GenreID) ON DELETE SET NULL,
   FOREIGN KEY (AlbumID) REFERENCES Album (AlbumID) ON DELETE SET NULL
);


CREATE TABLE Playlist
(
   PlaylistID   VARCHAR(5),
   PlaylistName VARCHAR(100),
   UserID       VARCHAR(5),
   PlaylistDesc TINYTEXT,
   PRIMARY KEY (PlaylistID),
   FOREIGN KEY (UserID) REFERENCES User (UserID) ON DELETE CASCADE
);

CREATE TABLE Produce
(
   ArtistID       VARCHAR(5),
   SongID         VARCHAR(5),
   PRIMARY KEY (ArtistID, SongID),
   FOREIGN KEY (ArtistID) REFERENCES Artist (ArtistID) ON DELETE CASCADE,
   FOREIGN KEY (SongID) REFERENCES Song (SongID) ON DELETE CASCADE
);
```

```sql
CREATE TABLE PlaylistContains
(
   PlaylistID VARCHAR(5),
   SongID     VARCHAR(5),
   AddDate    DATE,
   PRIMARY KEY (PlaylistID, SongID),
   FOREIGN KEY (PlaylistID) REFERENCES Playlist (PlaylistID) ON DELETE CASCADE,
   FOREIGN KEY (SongID) REFERENCES Song (SongID) ON DELETE CASCADE
);
```

5. Database Instance

User

UserID	CountryID	Username	UFirstName	ULastName	UBirthDate	City	ZipCode	SubID
10660	MX	user23	Juan	Gonzalez	1995-10-10	Mexico City	6010	2
12788	AU	user8	Sarah	Anderson	1988-09-10	Sydney	2000	2
17111	JP	user6	Linda	Davis	1998-06-25	Tokyo	100-0001	2
18720	CA	user22	David	Lee	1989-09-09	Toronto	M5V 1J2	1
26875	AU	user21	Emily	Brown	1991-08-08	Sydney	2000	0
28077	JP	user20	Taro	Suzuki	1993-07-07	Tokyo	100-0005	1
41461	CN	user19	Li	Wang	1998-06-06	Beijing	100005	0
50759	ES	user3	Bob	Johnson	1995-08-15	Madrid	28001	2
57053	DE	user15	Hans	Schmidt	1992-02-02	Berlin	10178	0
60347	CN	user9	Peter	Lee	1993-12-30	Shanghai	200000	0
6235	MX	user11	Juan	Perez	1997-07-20	Mexico City	1000	2
69091	DE	user2	Jane	Smith	1985-05-10	Berlin	10115	1
69944	RU	user12	Anna	Kovaleva	1994-02-28	Moscow	109012	1
70585	CH	user13	Mark	Schmidt	1996-10-15	Zürich	8001	0
76095	CA	user7	David	Wilson	1991-02-15	Toronto	M5G 1C7	0
78662	GB	user17	John	Smith	1994-04-04	London	SW1A 1AA	0
80235	FR	user1	John	Doe	1990-01-01	Paris	75001	0
82449	IT	user18	Marco	Rossi	1996-05-05	Rome	118	1
88395	BR	user10	Maria	Garcia	1989-04-05	São Paulo	01310-000	1
90590	RU	user24	Ivan	Petrov	1987-11-11	Moscow	101000	2
91215	BR	user25	Ana	Silva	1999-12-12	São Paulo	01000-000	0
94161	ES	user16	Maria	Garcia	1988-03-03	Madrid	28001	1
96136	GB	user5	Tom	Brown	1987-11-05	London	SW1A 0AA	0
9645	IT	user4	Alice	Williams	1992-03-20	Rome	118	1
98193	FR	user14	Jean	Dupont	1990-01-01	Paris	75001	1

Artist

ArtistID	ArtFirstName	ArtLastName	ArtDesc	ArtBirthDate	CountryID
11556	Janelle	Monáe	Singer-songwriter, actress, and producer	1985-12-01	US
13944	Mariah	Carey	Singer, songwriter, and actress	1970-03-27	US
20150	Ella	Fitzgerald	Singer	1917-04-25	US
24126	Nina	Simone	Singer, songwriter, and civil rights activist	1933-02-21	US
28074	Frank	Sinatra	Singer and actor	1915-12-12	US
30349	Ed	Sheeran	Singer-songwriter and record producer	1991-02-17	GB
33547	Elton	John	Pianist and composer	1947-03-25	GB
35632	Aretha	Franklin	Singer, songwriter, and pianist	1942-03-25	US
39909	Joni	Mitchell	Singer-songwriter and painter	1943-11-07	CA
42340	Beyoncé	Knowles	Singer, songwriter, and actress	1981-09-04	US
46035	Prince	Nelson	Singer, songwriter, and musician	1958-06-07	US
46082	Kendrick	Lamar	Rapper, songwriter, and record producer	1987-06-17	US
48428	Billie	Eilish	Singer-songwriter	2001-12-18	US
49643	Ludwig	van Beethoven	Composer and pianist	1770-12-16	AT
50798	Johann	Sebastian Bach	Composer and musician	1685-03-21	DE
51060	John	Lennon	Singer, songwriter, and peace activist	1940-10-09	GB
51997	Jimi	Hendrix	Singer, songwriter, and guitarist	1942-11-27	US
56246	David	Bowie	Singer, songwriter, and actor	1947-01-08	GB
58841	Lana	Del Rey	Singer-songwriter and record producer	1985-06-21	US
59126	Wolfgang	Amadeus Mozart	Composer	1756-01-27	AT
60085	Freddie	Mercury	Singer, songwriter, and record producer	1946-09-05	TZ
64042	Leonard	Cohen	Singer-songwriter, poet, and novelist	1934-09-21	CA
70862	Madonna	Ciccone	Singer, songwriter, actress, and businesswoman	1958-08-16	US
73863	Miles	Davis	Trumpeter, bandleader, and composer	1926-05-26	US
74540	Whitney	Houston	Singer and actress	1963-08-09	US
74677	Bob	Dylan	Singer-songwriter, author, and visual artist	1941-05-24	US
7667	Stevie	Wonder	Singer, songwriter, and multi-instrumentalist	1950-05-13	US
77210	Amy	Winehouse	Singer-songwriter	1983-09-14	GB
83473	Giuseppe	Verdi	Composer	1813-10-10	IT
94854	Adele	Adkins	Singer-songwriter	1988-05-05	GB

Song

SongID	SongName	GenreID	Duration	AlbumID
ADE01	Rolling in the Deep	27867	00:03:48	77380
ADE02	Someone Like You	27867	00:04:45	77380
ADE03	Set Fire to the Rain	27867	00:04:01	77380
ADE04	Hello	27867	00:04:55	57182
ADE05	When We Were Young	27867	00:04:50	89227
ARE01	Respect	68971	02:29:00	59559
ARE02	Chain of Fools	68971	02:46:00	60065
ARE03	Think	68971	02:19:00	99342
ARE04	A Natural Woman (You Make Me Feel Like)	68971	02:44:00	60065
ARE05	I Say a Little Prayer	68971	03:32:00	99342
AWH01	Rehab	15985	00:03:33	57260
AWH02	Back to Black	15985	00:04:08	57260
AWH03	Valerie	15985	00:03:53	31558
AWH04	Tears Dry On Their Own	15985	00:03:07	2828
AWH05	Love Is A Losing Game	15985	00:02:35	2828
BOW01	Space Oddity	12356	05:16:00	2693
BOW02	Heroes	12356	03:37:00	75849
BOW03	Starman	12356	04:14:00	60
BOW04	Let's Dance	27867	04:10:00	14389
BOW05	Life on Mars?	12356	03:53:00	80192
BY001	Single Ladies (Put a Ring on It)	27867	03:13:00	95749
BY002	Crazy in Love	27867	03:56:00	28904
BY003	Irreplaceable	27867	03:48:00	89321
BY004	Halo	27867	04:21:00	62560
BY005	Formation	68971	03:26:00	16334
EJ001	Your Song	27867	04:04:00	47987
EJ002	Rocket Man	12356	04:43:00	23840
EJ003	Tiny Dancer	27867	06:15:00	32884
EJ004	Candle in the Wind	27867	03:50:00	21615
EJ005	Don't Let the Sun Go Down on Me	27867	05:38:00	60671
ESH01	Shape of You	27867	00:03:53	34814
ESH02	Thinking Out Loud	27867	00:04:41	17826
ESH03	Photograph	27867	00:04:19	17826
ESH04	Castle on the Hill	27867	00:04:21	34814
FM001	Bohemian Rhapsody	12356	05:55:00	36808
FM002	We Are the Champions	12356	03:00:00	29977
FM003	Somebody to Love	12356	04:56:00	75187
FM004	Killer Queen	12356	03:03:00	23370
FS001	My Way	82654	00:04:36	63582
FS002	New York, New York	82654	00:03:33	86746
FS003	Fly Me to the Moon	82654	00:02:29	95244
FS004	Strangers in the Night	82654	00:02:45	3709
FS005	The Way You Look Tonight	82654	00:02:21	19488
JIM01	Purple Haze	12356	02:51:00	87597
JIM02	Hey Joe	12356	03:30:00	87597

Playlist

PlaylistID	PlaylistName	UserID	PlaylistDesc
11520	Eclectic Mix of Indie and Alternative Rock for Road Trips and Adventures	26875	A diverse mix of indie and alternative rock songs to accompany you on your road trips and adventures.
12865	Upbeat and Energetic Pop Hits for Workouts and Dance Parties	80235	A playlist featuring high-energy and upbeat pop songs that will keep you motivated during your workouts and dance parties.
78513	Soothing Acoustic Melodies for Relaxation and Focus	57053	A collection of calm and peaceful acoustic songs perfect for unwinding and focusing.
78911	Mellow Jazz and Blues for Cozy Nights In	10660	A selection of smooth and relaxing jazz and blues tracks to set the mood for a cozy night in.
84356	Classic Pop Rock Anthems	9645	A playlist featuring timeless hits from the 60s, 70s, and 80s that are sure to get you singing and dancing along.

Country

CountryID	CountryName
AD	Andorra
AE	United Arab Emirates
AF	Afghanistan
AG	Antigua and Barbuda
AI	Anguilla
AL	Albania
AM	Armenia
AO	Angola
AQ	Antarctica
AR	Argentina
AS	American Samoa
AT	Austria
AU	Australia
AW	Aruba
AX	Aland Islands
AZ	Azerbaijan
BA	Bosnia and Herzegovina
BB	Barbados
BD	Bangladesh
BE	Belgium
BF	Burkina Faso
BG	Bulgaria
BH	Bahrain
BI	Burundi
BJ	Benin
BL	Saint Barthélemy
BM	Bermuda
BN	Brunei Darussalam
BO	Bolivia
BQ	Bonaire, Sint Eustatius and Saba
BR	Brazil
BS	Bahamas
BT	Bhutan
BV	Bouvet Island
BW	Botswana
BY	Belarus
BZ	Belize
CA	Canada
CC	Cocos (Keeling) Islands
CD	Congo, the Democratic Republic of the
CF	Central African Republic
CG	Congo
CH	Switzerland
CI	Cote dIvoire
CK	Cook Islands
CL	Chile
CM	Cameroon
CN	China

Album

AlbumID	AlbumName	ReleaseDate	ArtistID
1335	Kind of Blue	1959-08-17	73863
14389	Let's Dance	1983-04-14	56246
16334	Lemonade	2016-04-23	42340
17826	x	2014-06-20	30349
18861	Milestones	1958-04-02	73863
19488	Days of Wine and Roses	1963-10-14	35632
2053	Parade	1986-03-31	56246
21615	Goodbye Yellow Brick Road	1973-10-05	33547
23359	Innervisions	1973-08-03	7667
23370	Sheer Heart Attack	1974-11-08	74677
23840	Honky Château	1972-05-19	33547
2693	David Bowie (a.k.a. Space Oddity)	1969-11-14	56246
27425	good kid, m.A.A.d city	2012-10-22	46082
27740	Electric Ladyland	1968-10-16	51997
2828	Back to Black	2006-10-27	77210
28452	Like a Virgin	1984-11-12	70862
28904	Dangerously in Love	2003-06-23	42340
29977	News of the World	1977-10-28	60085
31558	Back to Black: B-Sides	2012-11-12	77210
32884	Madman Across the Water	1971-11-05	33547
34647	Court and Spark	1974-01-01	39909
34814	÷	2017-03-03	30349
35514	To Pimp a Butterfly	2015-03-15	46082
36808	A Night at the Opera	1975-11-21	60085
3709	Strangers in the Night	1966-06-06	28074
38038	Kind of Blue	1959-08-17	73863
39196	The Bodyguard Soundtrack	1992-11-17	56246
4338	Songs in the Key of Life	1976-09-28	7667
43506	Talking Book	1972-10-28	7667
45339	Songs in the Key of Life	1976-09-28	7667
47010	Blue	1971-06-22	39909
47987	Elton John	1970-04-10	33547
50852	Whitney	1987-06-02	74540
57182	25	2015-11-20	94854
57260	Back to Black	2006-10-27	77210
58136	Whitney Houston	1985-02-14	74540
58931	Purple Rain	1984-06-25	56246
59559	I Never Loved a Man the Way I Love You	1967-03-10	35632
60	The Rise and Fall of Ziggy Stardust and the Spiders from Mars	1972-06-16	56246
60065	Lady Soul	1968-01-22	35632
60671	Caribou	1974-06-28	33547
61094	Like a Prayer	1989-03-21	70862
62560	I Am... Sasha Fierce	2008-11-14	42340
63582	My Way	1969-03-18	28074
74746	Talking Book	1972-10-28	7667
75187	A Day at the Races	1976-12-10	60085
75849	Heroes	1977-10-14	56246
76404	Ray of Light	1998-02-22	70862
77380	21	2011-01-24	94854
78651	Imagine	1971-09-09	51060
790	Shaved Fish	1975-10-24	51060
80192	Hunky Dory	1971-12-17	56246
84443	Ladies of the Canyon	1970-03-01	39909
86090	I'm Breathless	1990-05-22	70862
86170	Both Sides Now	2000-03-21	39909
86746	Trilogy: Past Present Future	1980-10-01	28074
87597	Are You Experienced	1967-08-23	51997
89227	+	2011-09-09	30349
89321	B'Day	2006-09-01	42340
89352	DAMN.	2017-04-14	46082
95244	It Might As Well Be Swing	1964-07-27	28074
95749	I Am... Sasha Fierce	2008-11-14	42340
96522	1999	1982-10-27	46035
98787	John Lennon/Plastic Ono Band	1970-12-11	51060
98976	Axis: Bold as Love	1967-12-01	51997
99342	Aretha Now	1968-06-14	35632

Produce

ArtistID	SongID
28074	FS001
28074	FS002
28074	FS003
28074	FS004
28074	FS005
30349	ESH01
30349	ESH02
30349	ESH03
30349	ESH04
33547	EJ001
33547	EJ002
33547	EJ003
33547	EJ004
33547	EJ005
35632	ARE01
35632	ARE02
35632	ARE03
35632	ARE04
35632	ARE05
39909	JMS01
39909	JMS02
39909	JMS03
39909	JMS04
39909	JMS05
42340	BY001
42340	BY002
42340	BY003
42340	BY004
42340	BY005
46035	PN001
46035	PN002
46035	PN003
46035	PN004
46035	PN005
46082	KDL01
46082	KDL02
46082	KDL03
46082	KDL04
46082	KDL05
51060	JL001
51060	JL002
51060	JL003
51060	JL004
51060	JL005
51997	JIM01
51997	JIM02

Genre

GenreID	GenreName	GenreDesc
10673	Folk	A genre of traditional music that originated in rural communities and was passed down orally from generation to generation.
11181	Blues	A genre of African American music that originated in the Deep South of the United States around the end of the 19th century.
12356	Rock	A genre of popular music that originated as rock and roll" in the United States in the 1950s."
12908	Metal	A genre of rock music that developed in the late 1960s and early 1970s, with roots in blues rock and psychedelic rock.
13563	Punk	A genre of rock music that emerged in the mid - 1970s, characterized by its fast tempos, hard - edged melodies, and often politically charged lyrics.
14765	World	A genre of music that encompasses traditional music from around the world, as well as contemporary music that incorporates elements of traditional styles.
15342	Gospel	A genre of Christian music that originated in African American communities in the United States in the late 19th and early 20th centuries, characterized by its emotive vocals and Christian lyrics.
15985	Soul	Soul music is a genre of popular music that originated in African American communities in the United States and combines elements of gospel, R&B, and jazz
27867	Pop	A genre of popular music that originated in the mid-1950s in the United States and the United Kingdom.
37898	Hip Hop	A genre of popular music that originated in African American and Hispanic American communities in the United States during the 1970s.
43564	Electronic	A genre of music that is produced using electronic devices such as synthesizers, drum machines, and computers.
55543	Country	A genre of American popular music that originated in the Southern United States in the 1920s.
68971	R & B	A genre of popular music that originated in African American communities in the United States in the 1940s.
71453	Reggae	A music genre that originated in Jamaica in the late 1960s.
82654	Jazz	A music genre that originated in the African American communities of New Orleans, United States, in the late 19th and early 20th centuries.
92536	Classical	A genre of music that originated in Europe in the 18th century and is characterized by its complex forms, sophisticated harmony, and rich instrumentation.

PlaylistContains

PlaylistID	SongID	AddDate
11520	BY005	2017-04-21
11520	EJ002	2015-06-16
11520	EJ003	2022-08-07
11520	EJ005	2022-06-25
11520	FM001	2017-12-14
11520	FM004	2017-09-05
11520	FS001	2016-02-24
11520	FS003	2019-05-13
11520	JL001	2018-11-25
11520	PN001	2022-12-03
11520	PN003	2023-04-17
11520	PN004	2018-08-24
11520	SW001	2021-06-17
11520	SW003	2023-06-21
11520	SW005	2018-06-30
11520	WH003	2021-01-18
12865	BY005	2017-05-27
12865	EJ003	2019-01-11
12865	EJ004	2019-06-27
12865	FS005	2017-11-02
12865	JL005	2016-11-04
12865	SW004	2021-06-24
12865	SW005	2021-09-16
12865	WH001	2018-02-05
12865	WH002	2022-01-04
78513	JL002	2020-01-13
78513	PN002	2017-04-14
78513	WH003	2015-06-26
78513	WH004	2018-01-07
78911	BY001	2018-04-12
78911	BY004	2021-01-08
78911	FM002	2016-10-09
78911	FS001	2015-08-04
78911	FS004	2020-12-02
78911	JL003	2017-02-12
84356	EJ002	2015-02-12
84356	EJ005	2023-01-03
84356	FM001	2023-01-20
84356	JL004	2023-10-26
84356	PN001	2018-01-01
84356	PN003	2020-05-31
84356	SW001	2021-12-22
84356	SW003	2020-12-13

6. SQL data queries

SELECT commands

To show all the tables:

User:

```
SELECT * FROM User;
```

Country

```
SELECT * FROM Country;
```

Artist

```
SELECT * FROM Artist;
```

Genre

```
SELECT * FROM Genre;
```

Song

```
SELECT * FROM Song;
```

Album

```
SELECT * FROM Album;
```

PlayList

```
SELECT * FROM PlayList;
```

SubscriptionType;

```
SELECT * FROM PlaylistContains;
```

PlaylistContains;

```
SELECT * FROM PlaylistContains;
```

Produce

```
SELECT * FROM Produce;
```

To show the description of the POP genre:

```
SELECT GenreDesc  FROM Genre
 WHERE genreName = "Pop";
```

genreDescription

GenreDesc
A genre of popular music that originated in the mid-1950s in the United States and the United Kingdom.

To show the description of the artist with Billie as first name:

```
SELECT artDesc FROM Artist
WHERE artFirstName = "Billie";
```

artistDescription

artDesc
Singer-songwriter

To show the name and price of all subscription types where the price is between 50 and 80:

```
SELECT SubName , price FROM SubscriptionType
WHERE Price > 50  AND Price > 80;
```

SubscriptionBet

SubName	price
Premium	70

To show each artistID, artist first and second names and the number of songs produced by each of these artists:

```
SELECT ArtistID, artfirstname, artlastname,  COUNT(songID) AS SongCount
```

```
FROM Produce NATURAL JOIN Artist
WHERE Artist.ArtistID = Produce.ArtistID
GROUP BY ArtistID;
```

SongCount

ArtistID	artfirstname	artlastname	SongCount
28074	Frank	Sinatra	5
30349	Ed	Sheeran	4
33547	Elton	John	5
35632	Aretha	Franklin	5
39909	Joni	Mitchell	5
42340	Beyoncé	Knowles	5
46035	Prince	Nelson	5
46082	Kendrick	Lamar	5
51060	John	Lennon	5
51997	Jimi	Hendrix	5
56246	David	Bowie	5
60085	Freddie	Mercury	4
70862	Madonna	Ciccone	5
73863	Miles	Davis	4
74540	Whitney	Houston	5
7667	Stevie	Wonder	5
77210	Amy	Winehouse	5
94854	Adele	Adkins	5

To show countryName , countryID and corresponding number of artists:

```
SELECT CountryName, COUNT(artistID) AS ArtistCount
FROM Artist NATURAL JOIN Country
WHERE country.countryid = artist.countryid
GROUP BY CountryName
 ORDER BY ArtistCount;
```

numberArtistsPerCountry

CountryName	ArtistCount
Tanzania, United Republic of	1
Italy	1
Germany	1
Austria	2
Canada	2
United Kingdom	6
United States	17

To show subscription types and number of users of that type and money earned from each subscription type:

```
SELECT SubName AS Sname, COUNT(userID),
(SELECT CASE Sname
  WHEN "Free"
  THEN (SELECT PRICE FROM SubscriptionType WHERE SubName = sname)
  WHEN "Premium"
  THEN COUNT(userID)*(SELECT Price FROM SubscriptionType WHERE SubName
  = sname)
  ELSE count(userID)*(SELECT PRICE FROM SubscriptionType WHERE SubName =
  sname)
  END)
FROM SubscriptionType NATURAL JOIN User
WHERE user.subid = subscriptiontype.subid
group by SubName;
```

moneyEarnedBySubscription

Sname	COUNT(userID)	moneyEarned
Family	6	600
Free	10	0
Premium	9	630

To create function to calculate age from birth date in years:

```
CREATE FUNCTION getAge ( vDate DATE) RETURNS INTEGER
RETURN TIMESTAMPDIFF(YEAR, vDate, CURDATE());
```

To classify users with respect to their age:

```
SELECT CASE
   WHEN GetAge(UBirthDate) BETWEEN 13 AND 20 THEN '13-20'
    WHEN GetAge(UBirthDate) BETWEEN 20 AND 30 THEN '20-30'
    WHEN GetAge(UBirthDate) BETWEEN 30 AND 50 THEN '30-50'
    ELSE '50+'
  END AS AgeRange,
  COUNT(userID) AS UserCount
   FROM User
   GROUP  BY AgeRange;
```

ageUsers

AgeRange	UserCount
20-30	12
30-50	13

7. SQL programming

Give examples of functions, procedures and triggers and explain what they do.

FUNCTIONS:

This function can be used to retrieve the number of artists that have produced songs in a given genre.

- **Given a genre name, the functions returns the number of artist that have produced songs**

```
DELIMITER //

CREATE FUNCTION Artists_by_genre(genre_name VARCHAR(30)) RETURNS
INT
BEGIN
            DECLARE   genre_id INT;
            DECLARE num_artists INT;
            SELECT genreId INTO genre_id FROM Genre WHERE genreName
= genre_name;
            SELECT COUNT(distinct artistId) into num_artists FROM
PRODUCE
            WHERE songId IN (SELECT songId FROM Song where genreId
= genre_id);
            RETURN num_artists;
END;
```

```
DELIMITER ;
```

```
SELECT GenreName, Artists_by_genre(GenreName) AS Count From genre;
```

	GenreName	Count
1	Folk	1
2	Blues	0
3	Rock	6
4	Metal	0
5	Punk	0
6	World	0
7	Gospel	0
8	Soul	2
9	Pop	8
10	Hip Hop	1
11	Electronic	0
12	Country	0
13	R & B	4
14	Reggae	0
15	Jazz	2
16	Classical	0

PROCEDURE:

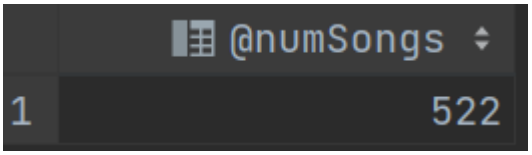
This procedure takes a country name as an input parameter and returns the number of songs by artists from that country as an output parameter.

```
DELIMITER //
```

```
CREATE PROCEDURE getNumSongsByCountry(IN country VARCHAR(255), OUT numSongs INT)
BEGIN
	SELECT COUNT(*) INTO numSongs
	FROM Song s
	 NATURAL JOIN Artist a
	 NATURAL JOIN Country c
	WHERE c.countryName = country;
END;
```

```
DELIMITER ;
```

```
CALL getNumSongsByCountry('United Kingdom', @numSongs);
SELECT @numSongs;
```



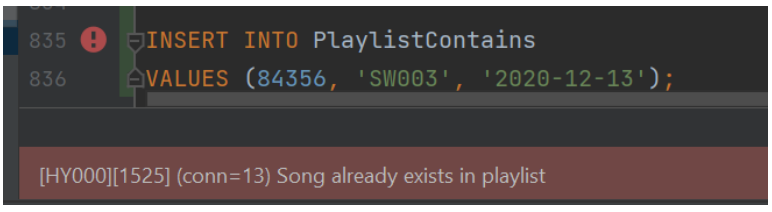
TRIGGERS

This is a trigger that will run before each insert on the PlaylistContains table. It checks if the new row being inserted already exists in the table by checking if there is a row with the same playlistId and songId.

```
DELIMITER //

CREATE TRIGGER before_playlist_contains_insert
BEFORE INSERT ON PlaylistContains
FOR EACH ROW
BEGIN
	IF EXISTS (SELECT * FROM PlaylistContains
		WHERE playlistId = NEW.playlistId AND songId = NEW.songId) THEN
	SIGNAL SQLSTATE 'HY000'
	SET MYSQL_ERRNO = 1525,
	MESSAGE_TEXT = 'Song already exists in playlist';
	END IF;
END;

DELIMITER ;
```



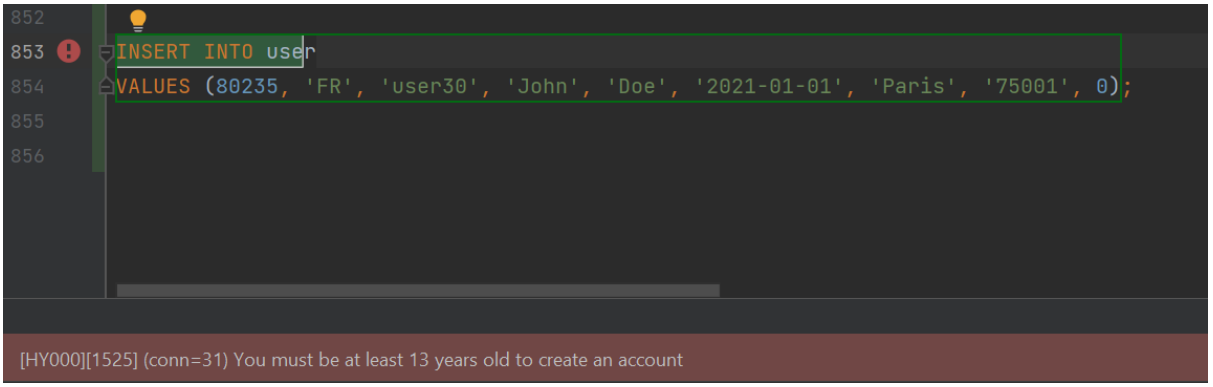
This is a trigger that will run before each insert on the User table. It checks that the new user is older than 13 years old .

```
DELIMITER //
```

```
CREATE TRIGGER before_user_insert
BEFORE INSERT ON User
FOR EACH ROW
BEGIN
   DECLARE age INT;
   SET age = TIMESTAMPDIFF(YEAR, NEW.UBirthDate, NOW());
   if age <13 then
       SIGNAL SQLSTATE 'HY000'
       SET MYSQL_ERRNO = 1525,
       MESSAGE_TEXT = 'You must be at least 13 years old to create an account';
   END IF;

   end ;
```

DELIMITER ;



8. SQL table modifications

a. INSERT commands

To insert a new SubscriptionType:

```
INSERT INTO SubscriptionType
VALUES (0, 'Free', 'A basic free subscription that allows you to
play music but with ads', 0);
```

To insert a new Country:

```
INSERT INTO Country
VALUES ('FR', 'France');
```

To insert a new User:

```
INSERT INTO User
VALUES (80235, 'FR', 'user1', 'John', 'Doe', '1990-01-01',
'Paris', '75001', 0);
```

To insert a new Artist:

```
INSERT INTO Artist
VALUES (51060, 'John', 'Lennon', 'Singer, songwriter, and peace
activist', '1940-10-09', 'GB');
```

To insert a new Genre:

```
INSERT INTO Genre
VALUES (12356, 'Rock',
       'A genre of popular music that originated as "rock and
roll" in the United States in the 1950s.');
```

To insert a new Album:

```
INSERT INTO Album
VALUES (98787, 'John Lennon/Plastic Ono Band', '1970-12-11',
51060);
```

To insert a new Song:

```
INSERT INTO Song
VALUES ('JL005', 'Working Class Hero', 12356, '03:48', 98787);
```

To insert a new Playlist:

```
INSERT Playlist
VALUES (12865, 'Upbeat and Energetic Pop Hits for Workouts and
Dance Parties', 80235,
       'A playlist featuring high-energy and upbeat pop songs that
will keep you motivated during your workouts and dance parties.');
```

To insert a new matching between a song and its composer/singer (Produce relationship):

```
INSERT INTO Produce
VALUES (51060, 'JL005');
```

To insert a new music in a playlist: (Playlist contains relationship):

```
INSERT INTO PlaylistContains
VALUES (12865, 'JL005', '2016-11-04');
```

b. DELETE commands

To delete a SubscriptionType:

```
DELETE FROM SubscriptionType WHERE SubID = 1;

DELETE FROM SubscriptionType WHERE SubName = 'Premium';
```

A SubID is a Foreign key of User, if a SubscriptionType is deleted, then the User is filled with the NULL subscription as:

```
CREATE TABLE User
(
   ......
   FOREIGN KEY (SubID) REFERENCES SubscriptionType (SubID) ON
DELETE SET NULL
);
```

To delete a Country:

```
DELETE FROM Country WHERE CountryID = 'FR';
DELETE FROM Country WHERE CountryName  = 'France';
```

A CountryID is a Foreign key of User and Artist, so if a Country is deleted, then the User and the Artist are filled with the NULL subscription as:

```
CREATE TABLE User
(
   ......
   FOREIGN KEY (CountryID) REFERENCES Country (CountryID) ON
DELETE SET NULL,
   ......
);

CREATE TABLE Artist
(
   ......
   FOREIGN KEY (CountryID) REFERENCES Country (CountryID) ON
DELETE SET NULL
);
```

To delete a User:

```
DELETE FROM User WHERE UserID = 10660;
DELETE FROM user WHERE CountryID = ‘KP’;
```

A UserID is a Foreign key of Playlist, so if a User is deleted, then all the playlists that belongs to this user are deleted:

```
CREATE TABLE Playlist
(
   ......
   FOREIGN KEY (UserID) REFERENCES User (UserID) ON DELETE
CASCADE
);
```

To delete an Artist:

```
DELETE FROM Artist WHERE ArtistID = 11556;
DELETE FROM Artist WHERE ArtName = Janelle Monáe
```

To delete a Genre:

```
DELETE FROM Genre WHERE GenreID = 10673;
DELETE FROM Genre WHERE GenreName = 'Folk';
```

To delete an Album:

```
DELETE FROM Album WHERE AlbumID = 01335;
DELETE FROM Album WHERE AlbumName = 'Kind of Blue';
```

An AlbumID is a Foreign key of Song, so if an Album is deleted, then the Song is filled with the NULL subscription as:

```
CREATE TABLE Song
(
  ......
   FOREIGN KEY (AlbumID) REFERENCES Album (AlbumID) ON DELETE
SET NULL
);
```

To delete a Song:

```
DELETE FROM Album WHERE SongID = 'ADE01';
DELETE FROM Album WHERE SongName =  'Rolling in the Deep';
```

A Song is a Foreign key of Produce and PlaylistContains, so if a Song is deleted, then the matching with the artist (Produce) and the matching with any playlist (PlaylistContains) are deleted:

```
CREATE TABLE Produce
(
   ......
   FOREIGN KEY (SongID) REFERENCES Song (SongID) ON DELETE
CASCADE
```

```
);

CREATE TABLE PlaylistContains
(
   ......
   FOREIGN KEY (SongID) REFERENCES Song (SongID) ON DELETE
CASCADE
);
```

To delete a Playlist:

```
DELETE FROM Playlist WHERE PlaylistID = 11520;
DELETE FROM Playlist WHERE PlaylistName = 'Eclectic Mix of Indie
and Alternative Rock for Road Trips and Adventures';
```

A Playlist is a Foreign key of PlaylistContains, so if a Playlist is deleted, then the matching with any Song(PlaylistContains) is deleted:

```
CREATE TABLE PlaylistContains
(
  ......
   FOREIGN KEY (PlaylistID) REFERENCES Playlist (PlaylistID) ON
DELETE CASCADE,
   ......
);
```

c. UPDATE command

Here is an Event that increases the price of each subscription every year due to inflation.

```
CREATE EVENT inflation
```

```
ON SCHEDULE EVERY 1 YEAR
STARTS '2023-01-01 00:00:00'
DO
 UPDATE SubscriptionType SET Price = Price * 1.05;
```

Another UPDATE statement that switches the subscription of a user to YOUNG if he is between 18 and 25 years old. It does it only if the user has not already the Young subscription (3) or the Family subscription (2).

```
UPDATE User SET SubID = 3
WHERE GetAge(UBirthDate) BETWEEN 18 AND 25
AND (SubID != 2 AND SubID != 3);
```

Also here is the opposite command that downgrade the subscription back to the free subscription (0) if the user is not between 18 and 25 years old anymore and had the Young subscription (3):

```
UPDATE User SET SubID = 0
WHERE GetAge(UBirthDate) NOT BETWEEN 18 AND 25
AND (SubId = 3);
```

Finally, here is an event with an update statement that gives a 1 month free premium subscription for all free users accounts when it's their birthday.

```
CREATE EVENT IF NOT EXISTS event_subscription_promotion ON
SCHEDULE EVERY 1 DAY STARTS CURRENT_TIMESTAMP
   DO
   BEGIN

        DECLARE birthday DATE;
        DECLARE pUserID VARCHAR(5);
        DECLARE oldSubID VARCHAR(1);

        SET birthday = CURDATE();

        -- Find users whose birthday is today and who have a free
subscription
        SELECT UserID, SubID
        INTO pUserID, oldSubID
        FROM User
```

```
        WHERE DAY(UBirthDate) = DAY(birthday)
          AND MONTH(UBirthDate) = MONTH(birthday)
          AND SubID = '0';

        IF pUserID IS NOT NULL THEN
            -- Give user a free Premium subscription for one month
            UPDATE User SET SubID = '1' WHERE UserID = pUserID;

            -- Schedule a job to run in 1 month that will update
the subscription back to the old subscription
            SET @sql = CONCAT('CREATE EVENT IF NOT EXISTS event_',
pUserID, ' ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 1 MONTH DO
UPDATE User SET SubID = ''', oldSubID, ''' WHERE UserID = ''',
pUserID, '''');
            PREPARE stmt FROM @sql; -- stmt is variable used in
prepared statements
            EXECUTE stmt;
            DEALLOCATE PREPARE stmt;
        END IF;
    END;
```