Josefa Kubitova

Student number: 16329260

Database Report: Music Streaming DB

# Database Description

A music streaming provider allows its users to stream songs made by various artists. Artists can be groups composed of multiple musicians and multiple artists can collaborate on one song, but every song has to be made by at least one artist. Users can ‘like’ a song they like, to make it easier to find again later. One user may ‘like’ many songs and songs can be ‘liked’ by many users.

The database contains the following tables:

**Song,** which contains the song’s ID (which serves as the primary key), its title, album, and runtime.

**Artist,** which contains the artist’s ID (which is the table’s primary key) and name.

**Users**, which contains all the users of the service. Users have an associated ID (which is the primary key of the table), username, a unique email address, whether or not their account is a premium account and their bank information.

**Musician,** which contains the musician’s ID (again, the primary key) and name.

**Song\_Genre,** which contains the song’s ID (a foreign key from the Song table), and the genre or genres it belongs to. Both of these combined are the primary key for this table.

**Song\_Artist,** which contains the song’s ID (a foreign key from the Song table) and the artist ID (Foreign key from the Artist table) for the artist or artists who created it. These two together are the primary key for this table.

**User\_Likes,** which contains the liked song’s ID (a foreign key from the Song table) and the user ID (foreign key from the User table) of whichever user or users liked it. These two combined are the primary key.

**Artist\_Member,** which contains the artist ID (foreign key from the Artist table), and

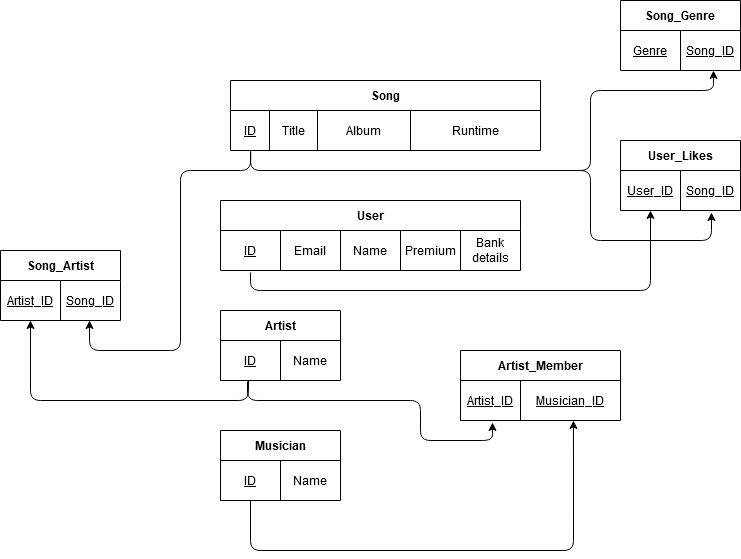
The database also contains the view **liked\_overview,** which creates a nice overview of the songs they liked for one of our users.

# Constraints

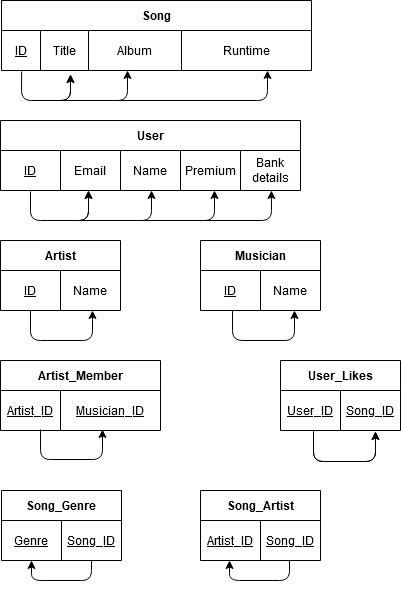
Most of the database constraints are implicit constraints on primary keys, but in User, the email attribute must also be unique and have the general form of an email address.

# C:\Users\Owner\Downloads\MusicDB.jpgRelational Diagram

# Relational Schema



# Functional Dependency Diagram



# Appendix: Code

--Create tables for the DB

CREATE TABLE Song(id INTEGER AUTO\_INCREMENT not null ,

title VARCHAR(50),

album VARCHAR(50),

runtime TIME,

PRIMARY KEY(id));

CREATE TABLE Artist(id INTEGER AUTO\_INCREMENT not null,

name VARCHAR(50),

PRIMARY KEY(id));

CREATE TABLE User(id INTEGER AUTO\_INCREMENT not null,

name VARCHAR(20),

--Create check: form of email address (Regex)

email VARCHAR(20) UNIQUE CHECK (email LIKE '%@%.%'),

premium BOOLEAN,

bank\_account VARCHAR(20),

PRIMARY KEY(id));

CREATE TABLE Musician(id INTEGER AUTO\_INCREMENT NOT NULL,

name VARCHAR(20),

PRIMARY KEY(id));

CREATE TABLE Song\_Genre(song\_id INTEGER,

genre VARCHAR(20),

FOREIGN KEY(song\_id) REFERENCES Song(id),

PRIMARY KEY(song\_id, genre));

CREATE TABLE Song\_Artist(song\_id INTEGER,

artist\_id INTEGER,

FOREIGN KEY(song\_id) REFERENCES Song(id),

FOREIGN KEY(artist\_id) REFERENCES Artist(id),

PRIMARY KEY(song\_id, artist\_id));

CREATE TABLE User\_Likes(user\_id INTEGER,

song\_id INTEGER,

FOREIGN KEY(song\_id) REFERENCES Song(id),

FOREIGN KEY(user\_id) REFERENCES User(id),

PRIMARY KEY(user\_id, song\_id));

CREATE TABLE Artist\_Member(artist\_id INTEGER,

musician\_id INTEGER,

FOREIGN KEY(artist\_id) REFERENCES Artist(id),

FOREIGN KEY(musician\_id) REFERENCES Musician(id),

PRIMARY KEY(artist\_id, musician\_id));

--Populate the database

INSERT INTO Song (id, title, album, runtime) VALUES(1, 'Blood And Whiskey', 'High Noon Over Camelot', '00:03:45');

INSERT INTO Song (id, title, album, runtime) VALUES(DEFAULT, 'Once and Future King', 'High Noon Over Camelot', '00:05:29');

INSERT INTO Song (id, title, album, runtime) VALUES(DEFAULT, 'Her Sweet Kiss', 'The Witcher', '00:02:00');

INSERT INTO Song (id, title, album, runtime) VALUES(DEFAULT, 'Fair', 'The Horror And The Wild', '00:06:18');

INSERT INTO Song (id, title, album, runtime) VALUES(DEFAULT, 'Pruning Shears', 'Love Run', '00:04:14');

INSERT INTO Song (id, title, album, runtime) VALUES(DEFAULT, 'Nina Cried Power', 'Wasteland, Baby!', '00:03:45');

INSERT INTO Song (id, title, album, runtime) VALUES(DEFAULT, 'Sunlight', 'Wasteland, Baby!', '00:04:17');

INSERT INTO Song (id, title, album, runtime) VALUES(DEFAULT, 'High Note', 'Livin On A High Note','00:03:31');

INSERT INTO Artist(id, name) VALUES(DEFAULT, 'The Mechanisms');

INSERT INTO Artist(id, name) VALUES(DEFAULT, 'The Amazing Devil');

INSERT INTO Artist(id, name) VALUES(DEFAULT, 'Hozier');

INSERT INTO Artist(id, name) VALUES(DEFAULT, 'Joey Batey');

INSERT INTO Artist(id, name) VALUES(DEFAULT, 'Mavis Stapele');

INSERT INTO User(id, name, email, premium, bank\_account) VALUES(DEFAULT, 'Jamie4', 'JSmith@gmail.com', FALSE, '');

INSERT INTO User(id, name, email, premium, bank\_account) VALUES(DEFAULT, 'Chriscat', 'namename@bing.com', TRUE, 'GR96 0810 0010 0000 0123 4567 890');

INSERT INTO User(id, name, email, premium, bank\_account) VALUES(DEFAULT, 'Olly77', 'jemapelle@email.com', FALSE, '');

INSERT INTO User(id, name, email, premium, bank\_account) VALUES(DEFAULT, 'MrA', 'clovicek@mail.ru', FALSE, '');

INSERT INTO User(id, name, email, premium, bank\_account) VALUES(DEFAULT, 'Vivielle', 'mort@gmail.com', TRUE, 'BE71 0961 2345 6769');

INSERT INTO Musician(id, name) VALUES(DEFAULT, 'Jonathan Sims');

INSERT INTO Musician(id, name) VALUES(DEFAULT, 'Jessica Law');

INSERT INTO Musician(id, name) VALUES(DEFAULT, 'Kofi Young');

INSERT INTO Musician(id, name) VALUES(DEFAULT, 'Joey Batey');

INSERT INTO Musician(id, name) VALUES(DEFAULT, 'Madeleine Hyland');

INSERT INTO Musician(id, name) VALUES(DEFAULT, 'Mavis Stapele');

INSERT INTO Musician(id, name) VALUES(DEFAULT, 'Andrew John Hozier-Byrne');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(1, 'steampunk');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(1, 'folk');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(2, 'folk');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(2, 'ballad');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(3, 'pop');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(3, 'ballad');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(4, 'folk');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(5, 'folk');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(5, 'pop');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(6, 'pop');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(6, 'folk');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(7, 'pop');

INSERT INTO Song\_Genre(song\_id, genre) VALUES(8, 'blues');

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(1, 1);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(2, 1);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(3, 4);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(4, 2);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(5, 2);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(6, 3);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(7, 3);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(8, 5);

INSERT INTO Song\_Artist(song\_id, artist\_id) VALUES(6, 5);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(1, 4);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(1, 2);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(2, 3);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(3, 5);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(2, 7);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(4, 3);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(4, 1);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(3, 1);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(3, 2);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(3, 6);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(4, 5);

INSERT INTO User\_Likes(user\_id, song\_id) VALUES(5, 8);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(1, 1);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(1, 2);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(1, 3);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(2, 4);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(2, 5);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(3, 7);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(4, 4);

INSERT INTO Artist\_Member(artist\_id, musician\_id) VALUES(5, 6);

--Create view

CREATE VIEW liked\_overview AS

SELECT Song.title, Song.album, Artist.name

FROM Song, Artist

WHERE Song.id IN ( SELECT User\_Likes.song\_id WHERE User\_Likes.user\_id=3)

AND Song.id = Song\_Artist.Song\_ID

AND Artist.id = Song\_Artist.artist\_id;