University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: ___2_____

Date: _____October 20 2023_______

Group Number: _____148_____

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Rajan Sapkota	86981594	q3q2d	therealrajansapkota@gmail.com
Finn Toner	42264903	e2m8u	toner.finn@gmail.com

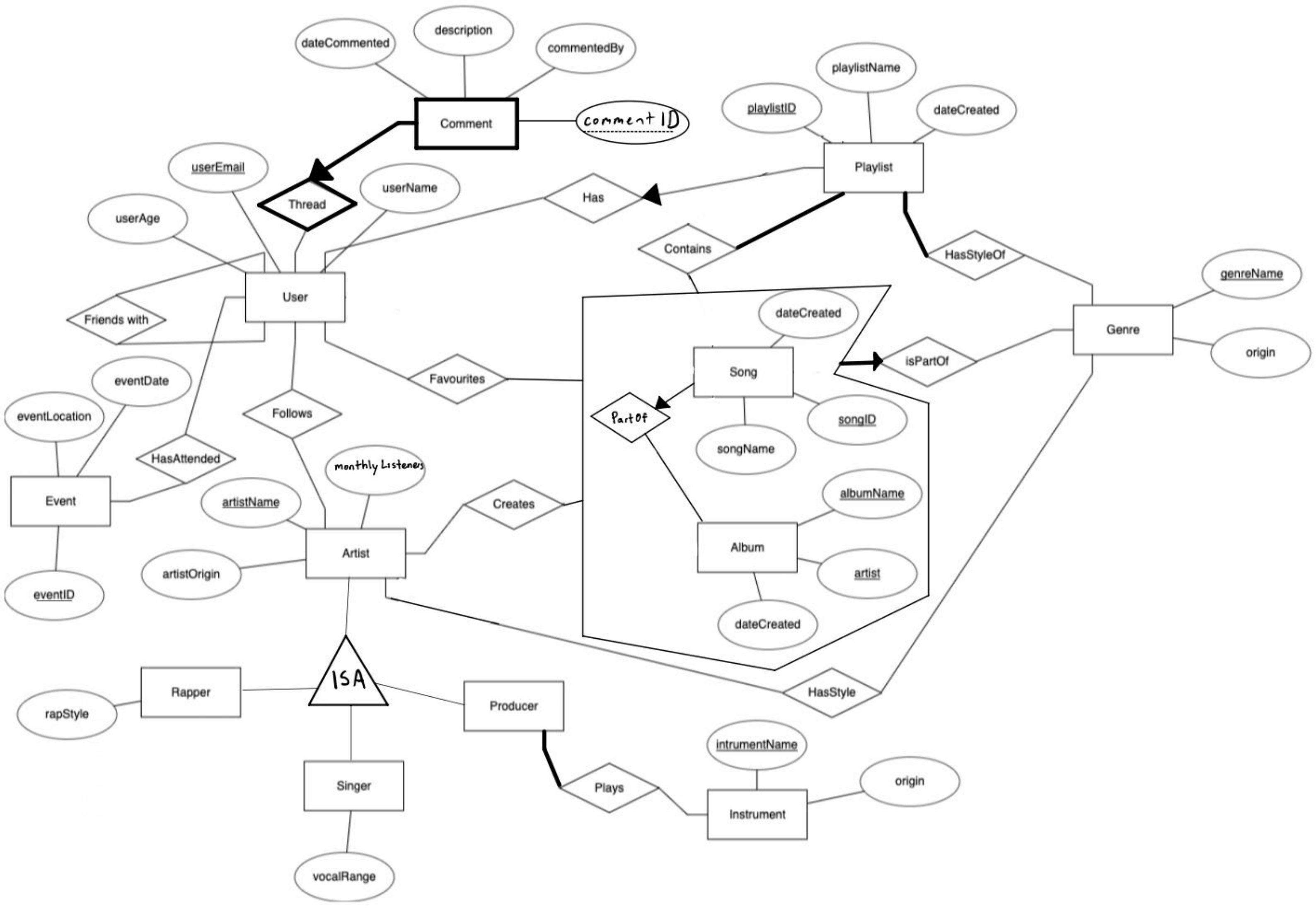
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

University of British Columbia, Vancouver
Department of Computer Science

Project Description:

We are creating a social platform for music lovers to connect and share their tastes. Users can add friends, explore and comment on their friends playlists, explore events that they are going to and more.



Notes:

- We added the following attributes for FDs:
    - User: userBirthday
    - Event: eventType, maxCapacity, minPrice
    - Rapper: era, city
    - Song: mood, BPM, numListens
- Our ISA's constraint is total overlapping (every artist must be either a rapper, singer or producer)

Pre Normalization:

User (userName: string, userEmail: string, userAge: integer (userAge > 0), userBirthday: date)

String)

PrimaryKey: userEmail

FDs:

userEmail -> displayName, userName, userBirthday

userBirthday -> userAge

Minimal Cover

Step 1 - Standardize such that for $X \ -> \ b$, b is a single attribute:

$userEmail \ -> \ userName$

$userEmail \ -> \ userBirthday$

$userBirthday \ -> \ userAge$

Step 2 - Reduce the left hand side to a single attribute:

Already reduced

Step 3 - Remove Redundancies:

No redundancies

Normalization:

$userEmail \ -> \ userName$

$userEmail \ -> \ userBirthday$

$userBirthday \ -> \ userAge$ *Violates BCNF/3NF*

User1(userBirthday, userAge), User2(userBirthday, userEmail, userName, userBirthday)

Thread (commentID: int (commentID > 0), threadName: string, commentedBy: string, description: string, dateCommented: date)

PrimaryKey: commentID, threadName (NOT NULL)

ForeignKey: commentedBy REFERENCES User

threadName REFERENCES User

FDs:

commentID, threadName -> commentedBy, description, dateCommented

FriendsWith (friend: string, friended: string)

PrimaryKey: friend, friended
ForeignKey: friend REFERENCES User
friended REFERENCES User

Follows (follower: string, following: string)
PrimaryKey: follower, following
ForeignKey: follower REFERENCES User
following REFERENCES Artist

HasAttended(attendee: string, eventName: string, eventDate: date)
PrimaryKey: attendee, eventName, eventDate
ForeignKey: attendee REFERENCES User
(eventName, eventDate) REFERENCES Event(eventName, eventDate)

Event (eventName: string, eventLocation: string, eventDate: date, eventType: string, maxCapacity: int > 0, minPrice: int > 0)
PrimaryKey: eventName, eventDate
FDs:
eventLocation -> eventType
eventName, eventDate -> eventLocation
eventLocation -> maxCapacity
maxCapcity -> minPrice

Artist (artistName: string, artistOrigin: string, monthlyListeners: int (monthlyListeners >= 0), artistDescription: string, monthlyEarnings: int)
PrimaryKey: artistName

FDs:
artistName -> artistOrigin
artistName -> monthlyListeners
artistName -> artistDescription
artistDescription -> monthlyEarnings

Creates(songID: int, albumName: string, artist: string)
PrimaryKey: songID, albumName, artist
ForeignKey: songID REFERENCES Song
albumName REFERENCES Album
artistName REFERENCES Artist

Rapper(artistName: name, rapStyle: string, era: int, city: string)

PrimaryKey: artistName

ForeignKey: artistName REFERENCES Artist

FDs:

artistName -> rapStyle
rapStyle -> era
rapStyle -> city

Singer(artistName: name, vocalRange: string)

PrimaryKey: artistName

ForeignKey: artistName REFERENCES Artist

FDs:

artistName -> vocalRange

Producer(artistName: string)

PrimaryKey: artistName

ForeignKey: artistName REFERENCES Artist

Plays(artistName: string, instrumentName: string, yearsExperience: int, level: string)

PrimaryKey: artistName(NOT NULL), instrumentName

ForeignKey: artistName REFERENCES Artist
instrumentName REFERENCES Instrument

FDs:

artistName, instrumentName -> yearsExperience
yearsExperience -> level

Instrument(instrumentName: string, origin: string)

PrimaryKey: instrumentName

FDs:

instrumentName -> origin

Playlist(playlistID: int, playlistName: string, dateCreated: date, createdBy: string)

PrimaryKey: playlistID

ForeignKey: createdBy REFERENCES User

FDs:

playlistID -> playlistName, dateCreated, createdBy

Contains(playlistID: int, songID: string, albumName: string, artist: string)

PrimaryKey: playlistID (NOT NULL), songID, albumName, artist
ForeignKey: playlistID REFERENCES Playlist
songID REFERENCES Song
(albumName, artist) REFERENCES Album(albmName, artist)

HasStyleOf(playlistID: int, genreName: string)
PrimaryKey: playlistID (NOT NULL), genreName
ForeignKey: playlistID REFERENCES Playlist
genreName REFERENCES Genre

HasStyle(artistName: string, genreName: string)
PrimaryKey: artistName (NOT NULL), genreName
ForeignKey: artistName REFERENCES Artist
genreName REFERENCES Genre

Song(dateCreated: date, songID: int, songName: string, albumName: string, artistName: string, isPartOf: string, BPM: int (BPM >= 24), mood: string, numListens: int (numListens > 0))
PrimaryKey: songID
ForeignKey: (albumName, artistName) REFERENCES Album(albumName, artist)
isPartOf REFERENCES Genre
FDs:
songID -> dateCreated
songID -> songName
songID -> albumName
songID -> isPartOf
songID -> artistName
songName, albumName -> numListens
isPartOf -> mood
mood -> BPM

Favorites(favoritedBy: string, songID: string, albumName: string, artist: string)
PrimaryKey: favorited, songID, albumName, artist
ForeignKey: favoritedBy REFERENCES User
songID REFERENCES Song
(albumName, artist) REFERENCES Album(albmName, artist)

Album(albumName: string, artist: string,)
PrimaryKey: albumName, artist
ForeignKey: artist REFERENCES Artist

Genre(genreName: string, origin: string)

PrimaryKey: genreName

FDs:

genreName -> origin

Normalization:

User:

Minimal Cover:

Step 1 - Standardize such that for $X \ -> \ b$, b is a single attribute:

$userEmail \ -> \ userName$

$userEmail \ -> \ userBirthday$

$userBirthday \ -> \ userAge$

Step 2 - Reduce the left hand side to a single attribute:

Already reduced

Step 3 - Remove Redundancies:

No redundancies

Normalization:

$userEmail \ -> \ userName$

$userEmail \ -> \ userBirthday$

$userBirthday \ -> \ userAge$ *Violates BCNF/3NF*

User1(userBirthday, userAge), User2(userBirthday, userEmail, userName, userBirthday)

Thread:

Minimal Cover:

commentID, threadName -> commentedBy,

commentID, threadName -> description

commentID, threadName -> dateCommented

Thread is already normalized, since the left side is the primary key and the right side is all single attributes

Event:

Minimal Cover:

Step 1 - Standardize such that for $X \ -> \ b$, b is a single attribute:

$eventName, eventDate -> eventName$
$eventName, eventDate -> eventLocation$
$eventLocation -> eventType$
$eventLocation -> maxCapacity$
$maxCapacity -> minPrice$

Step 2 - Reduce LHS:
Already reduced

Step 3 - Remove Redundancies:
No redundancies

Normalization:

Event1(eventLocation, eventType), Event2(eventLocation, eventName, eventDate, maxCapacity, minPrice)

Decompose further:
Event1(eventLocation, eventType), Event2(eventLocation, maxCapacity), Event3(eventLocation, eventName, eventDate, minPrice)

Final:
<u>Event1(eventLocation, eventType), Event2(eventLocation, maxCapacity), Event3(eventLocation, minPrice), Event4(eventLocation, eventName, eventDate), Event5(maxCapacity, minPrice)</u>

Artist:

Minimal Cover:

Step 1 - Standardize such that for X -> b, b is a single attribute:
All FDs are already in this form:
artistName -> artistOrigin
artistName -> monthlyListeners
artistName -> artistDescription
artistDescription -> monthlyEarnings
Step 2 - Reduce LHS:
All FDs are already reduced with single attributes on the left-hand side.

Step 3 - Remove Redundancies:
There are no redundancies.

Final:
Decompose artist into:
Artist1(artistName, artistOrigin, monthlyListeners, artistDescription), Artist2(artistDescription, monthlyEarnings)

Rapper:

Minimal Cover:
- Step 1 - Standardize such that for X -> b, b is a single attribute:
    - artistName -> rapStyle
    - rapStyle -> era
    - rapStyle -> city
- Step 2 - Reduce LHS:
    - All FDs are already reduced with single attributes on the left-hand side.
- Step 3 - Remove Redundancies:
    - There are no redundancies.

Final:
Decompose Rapper into:
Rapper1(rapStyle, era), Rapper2(rapStyle, city), Rapper3(artistName, rapStyle)

Plays:
Minimal Cover:
- Step 1 - Standardize such that for X -> b, b is a single attribute:
    - artistName, instrumentName -> yearsExperience
    - yearsExperience -> level
- Step 2 - Reduce LHS:
    - FDs are maximally reduced on the LHS
- Step 3 - Remove Redundancies:
    - There are no redundancies.

Final::
Decompose Plays into:
Plays1(artistName, instrumentName, yearsExperience), Plays2(yearsExperience, level)

Song:
Minimal Cover:
- Step 1 - Standardize such that for X -> b, b is a single attribute:

songID -> dateCreated
songID -> songName
songID -> albumName
songID -> isPartOf
songID -> artistName
songName, albumName -> numListens
isPartOf -> mood
isPartOf -> BPM
mood -> BPM

Step 2 - Reduce LHS:
FDs are maximally reduced on the LHS

Step 3 - Remove Redundancies:
isPartOf -> BPM removed due to redundancy

Normalization:
Decompose Song into:

Song1(songName, albumName, numListens), Song2(songID, dateCreated, songName, albumName, isPartOf, artistName, BPM, mood)

Decompose further:

Song1(songName, albumName, numListens), Song2(isPartOf, mood), Song3(songID, dateCreated, songName, albumName, isPartOf, artistName, BPM, mood)

Decompose further

Song1(songName, albumName, numListens), Song2(isPartOf, mood), Song3(isPartOf, BPM), Song4(songID, dateCreated, songName, albumName, isPartOf, artistName, BPM, mood)

Final:

<u>Song1(songName, albumName, numListens), Song2(isPartOf, mood), Song3(isPartOf, BPM), Song4(songID, dateCreated, songName, albumName, isPartOf, artistName, BPM, mood), Song5(mood, BPM</u>

Post-Normalization:

User (userName: string, userEmail: string, userBirthday: date)
String)
PrimaryKey: userEmail
ForeignKey: userBirthday REFERENCES User_Birthday

User_Birthday (userBirthday: date, userAge: int)
PrimaryKey: userBirthday

Event (eventName: string, eventDate: date, eventLocation: string)
PrimaryKey: eventName, eventDate
ForeignKey: eventLocation REFERENCES Event_Ticket, Event_Venue, Event_Type

Event_Ticket (eventLocation: string, minPrice: int)
PrimaryKey: eventLocation

Event_Venue (eventLocation: string, maxCapacity: int)
PrimaryKey: eventLocation
ForeignKey: maxCapacity REFERENCES Event_Attendance

Event_Type (eventLocation: string, eventType: string)
PrimaryKey: eventLocation

Event_Attendance (maxCapacity: int, minPrice: int)
PrimaryKey: maxCapacity

Artist (artistName: string, artistOrigin: string, artistDescription: string, monthlyListeners: int)
PrimaryKey: artistName
ForeignKey: monthlyListeners REFERENCES Artist_Viewership

Artist_Viewership (monthlyListeners: int, monthlyEarnings: int)
PrimaryKey: monthlyListeners

Rapper (artistName: string, rapStyle: string)
PrimaryKey: artistName
ForeignKey: artistName REFERENCES artist
rapStyle REFERENCES Rapper_Time, Rapper_Origin

Rapper_Time (rapStyle: string, era: int)
PrimaryKey: rapStyle

Rapper_Origin (rapStyle: string, city: string)
PrimaryKey: rapStyle

Plays(artistName: string, instrumentName: string, yearsExperience: int)
PrimaryKey: artistName(NOT NULL), instrumentName
ForeignKey: artistName REFERENCES Artist
instrumentName REFERENCES Instrument
yearsExperience REFERENCES Plays_Experience

Plays_Experience (yearsExperience: int, level: string)
PrimaryKey: yearsExperience

Song(songID: int, songName: string, dateCreated: date, artistName: string, albumName: string, isPartOf: string)
PrimaryKey: songID
ForeignKey: (albumName, artistName) REFERENCES Album(albumName, artist)
isPartOf REFERENCES Genre, Song_Genre, Song_Vibe, Song_Mood
(songName, artistName) REFERENCES Song_Viewership(songName, artistName)

Song_Mood(mood: string, BPM: int (BPM >= 24))
PrimaryKey: mood

Song_Genre(isPartOf: string, BPM: int (BPM >= 24))
PrimaryKey: isPartOf

Song_Vibe(isPartOf: string, mood: string)

PrimaryKey: isPartOf
ForeignKey: mood REFERENCES Song_Mood(mood)

Song_Viewership(songName: string, artistName: string, numOfListeners: int)
PrimaryKey: songName, artistName

Thread (commentID: int (commentID > 0), threadName: string, commentedBy: string, description: string,
dateCommented: date)
CandidateKeys:
PrimaryKey: commentID, threadName (NOT NULL)
ForeignKey: commentedBy REFERENCES User
threadName REFERENCES User
FDs:
commentID, threadName -> commentedBy, description, dateCommented

FriendsWith (friend: string, friended: string)
PrimaryKey: friend, friended
ForeignKey: friend REFERENCES User
friended REFERENCES User

Follows (follower: string, following: string)
PrimaryKey: follower, following
ForeignKey: follower REFERENCES User
following REFERENCES Artist

HasAttended(attendee: string, eventName: string)
PrimaryKey: attendee, eventName
ForeignKey: attendee REFERENCES User
eventName REFERENCES Event

Creates(songID: int, albumName: string, artist: string)
PrimaryKey: songID, albumName, artist
ForeignKey: songID REFERENCES Song
albumName REFERENCES Album
artistName REFERENCES Artist

Singer(artistName: name, vocalRange: string)

PrimaryKey: artistName
ForeignKey: artistName REFERENCES Artist

Producer(artistName: string)
PrimaryKey: artistName
ForeignKey: artistName REFERENCES Artist

Instrument(instrumentName: string, origin: string)
PrimaryKey: instrumentName

Playlist(playlistID: int, playlistName: string, dateCreated: date, createdBy: string)
PrimaryKey: playlistID
ForeignKey: createdBy REFERENCES User

Contains(playlistID: int, songID: string, albumName: string, artist: string)
PrimaryKey: playlistID (NOT NULL), songID, albumName, artist
ForeignKey: playlistID REFERENCES Playlist
songID REFERENCES Song
(albumName, artist) REFERENCES Album(albmName, artist)

HasStyleOf(playlistID: int, genreName: string)
PrimaryKey: playlistID (NOT NULL), genreName
ForeignKey: playlistID REFERENCES Playlist
genreName REFERENCES Genre

HasStyle(artistName: string, genreName: string)
PrimaryKey: artistName (NOT NULL), genreName
ForeignKey: artistName REFERENCES Artist
genreName REFERENCES Genre

Favorites(favoritedBy: string, songID: string, albumName: string, artist: string)
PrimaryKey: favorited, songID, albumName, artist
ForeignKey: favoritedBy REFERENCES User
songID REFERENCES Song
(albumName, artist) REFERENCES Album(albmName, artist)

Album(albumName: string, artist: string, dateCreated: date)
PrimaryKey: albumName, artist
ForeignKey: artist REFERENCES Artist

Genre(genreName: string, origin: string)
	PrimaryKey: genreName

SQL CODE:

```
CREATE TABLE User_Birthday (
   userBirthday DATE PRIMARY KEY,
   userAge INT
);

CREATE TABLE User (
   userName VARCHAR(255),
   userEmail VARCHAR(255) PRIMARY KEY,
   userBirthday DATE,
   FOREIGN KEY (userBirthday) REFERENCES User_Birthday(userBirthday)
);

CREATE TABLE Event_Ticket (
   eventLocation VARCHAR(255) PRIMARY KEY,
   minPrice INT
);

CREATE TABLE Event_Attendance (
   maxCapacity INT PRIMARY KEY,
   minPrice INT
);

CREATE TABLE Event_Venue (
   eventLocation VARCHAR(255) PRIMARY KEY,
   maxCapacity INT,
   FOREIGN KEY (maxCapacity) REFERENCES Event_Attendance(maxCapacity)
);

CREATE TABLE Event_Type (
   eventLocation VARCHAR(255) PRIMARY KEY,
   eventType VARCHAR(255)
);
```

```
CREATE TABLE Event (
   eventName VARCHAR(255),
   eventDate DATE,
   eventLocation VARCHAR(255),
   PRIMARY KEY (eventName, eventDate),
   FOREIGN KEY (eventLocation) REFERENCES Event_Ticket(eventLocation),
   FOREIGN KEY (eventLocation) REFERENCES Event_Venue(eventLocation),
   FOREIGN KEY (eventLocation) REFERENCES Event_Type(eventLocation)
);

CREATE TABLE Artist_Viewership (
   monthlyListeners INT PRIMARY KEY,
   monthlyEarnings INT
);

CREATE TABLE Artist (
   artistName VARCHAR(255) PRIMARY KEY,
   artistOrigin VARCHAR(255),
   artistDescription TEXT,
   monthlyListeners INT,
   FOREIGN KEY (monthlyListeners) REFERENCES Artist_Viewership(monthlyListeners)
);

CREATE TABLE Rapper_Time (
   rapStyle VARCHAR(255) PRIMARY KEY,
   era INT
);

CREATE TABLE Rapper_Origin (
   rapStyle VARCHAR(255) PRIMARY KEY,
   city VARCHAR(255)
);

CREATE TABLE Rapper (
   artistName VARCHAR(255) PRIMARY KEY,
   rapStyle VARCHAR(255),
   FOREIGN KEY (artistName) REFERENCES Artist(artistName),
   FOREIGN KEY (rapStyle) REFERENCES Rapper_Time(rapStyle),
   FOREIGN KEY (rapStyle) REFERENCES Rapper_Origin(rapStyle)
);
```

```
CREATE TABLE Instrument (
   instrumentName VARCHAR(255) PRIMARY KEY,
   origin VARCHAR(255)
);

CREATE TABLE Plays_Experience (
   yearsExperience INT PRIMARY KEY,
   level VARCHAR(255)
);

CREATE TABLE Plays (
   artistName VARCHAR(255) NOT NULL,
   instrumentName VARCHAR(255),
   yearsExperience INT,
   PRIMARY KEY (artistName, instrumentName),
   FOREIGN KEY (artistName) REFERENCES Artist(artistName),
   FOREIGN KEY (instrumentName) REFERENCES Instrument(instrumentName),
   FOREIGN KEY (yearsExperience) REFERENCES Plays_Experience(yearsExperience)
);

CREATE TABLE Genre (
   genreName VARCHAR(255) PRIMARY KEY,
   origin VARCHAR(255)
);

CREATE TABLE Album (
   albumName VARCHAR(255),
   artist VARCHAR(255),
   dateCreated DATE,
   PRIMARY KEY (albumName, artist),
   FOREIGN KEY (artist) REFERENCES Artist(artistName)
);

CREATE TABLE Song_Mood (
   mood VARCHAR(255) PRIMARY KEY,
   BPM INT CHECK (BPM >= 24)
);

CREATE TABLE Song_Genre (
   isPartOf VARCHAR(255) PRIMARY KEY,
   BPM INT CHECK (BPM >= 24)
);

CREATE TABLE Song_Vibe (
   isPartOf VARCHAR(255) PRIMARY KEY,
   mood VARCHAR(255),
   FOREIGN KEY (mood) REFERENCES Song_Mood(mood)
);

CREATE TABLE Song_Viewership (
   songName VARCHAR(255),
   artistName VARCHAR(255),
   numOfListeners INT,
   PRIMARY KEY (songName, artistName)
);

CREATE TABLE Song (
   songID INT PRIMARY KEY,
   songName VARCHAR(255),
   dateCreated DATE,
   artistName VARCHAR(255),
   albumName VARCHAR(255),
   isPartOf VARCHAR(255),
   FOREIGN KEY (albumName, artistName) REFERENCES Album(albumName, artist),
   FOREIGN KEY (isPartOf) REFERENCES Genre(genreName),
   FOREIGN KEY (isPartOf) REFERENCES Song_Genre(isPartOf),
   FOREIGN KEY (isPartOf) REFERENCES Song_Vibe(isPartOf),
   FOREIGN KEY (isPartOf) REFERENCES Song_Mood(mood),
   FOREIGN KEY (songName, artistName) REFERENCES Song_Viewership(songName,
artistName)
);

CREATE TABLE Thread (
   commentID INT CHECK (commentID > 0),
   threadName VARCHAR(255) NOT NULL,
   commentedBy VARCHAR(255),
   description TEXT,
   dateCommented DATE,
   PRIMARY KEY (commentID, threadName),
   FOREIGN KEY (commentedBy) REFERENCES User(userEmail),
   FOREIGN KEY (threadName) REFERENCES User(userEmail)
);

CREATE TABLE FriendsWith (
   friend VARCHAR(255),
   friended VARCHAR(255),
   PRIMARY KEY (friend, friended),
   FOREIGN KEY (friend) REFERENCES User(userEmail),
   FOREIGN KEY (friended) REFERENCES User(userEmail)
);

CREATE TABLE Follows (
   follower VARCHAR(255),
   following VARCHAR(255),
   PRIMARY KEY (follower, following),
   FOREIGN KEY (follower) REFERENCES User(userEmail),
   FOREIGN KEY (following) REFERENCES Artist(artistName)
);

CREATE TABLE HasAttended (
   attendee VARCHAR(255),
   eventName VARCHAR(255),
   eventDate DATE,
   PRIMARY KEY (attendee, eventName, eventDate),
   FOREIGN KEY (attendee) REFERENCES User(userEmail),
   FOREIGN KEY (eventName, eventDate) REFERENCES Event(eventName, eventDate)
);


CREATE TABLE Singer (
   artistName VARCHAR(255) PRIMARY KEY,
   vocalRange VARCHAR(255),
   FOREIGN KEY (artistName) REFERENCES Artist(artistName)
);

CREATE TABLE Producer (
   artistName VARCHAR(255) PRIMARY KEY,
   FOREIGN KEY (artistName) REFERENCES Artist(artistName)
);
```

```
CREATE TABLE Playlist (
   playlistID INT PRIMARY KEY,
   playlistName VARCHAR(255),
   dateCreated DATE,
   createdBy VARCHAR(255),
   FOREIGN KEY (createdBy) REFERENCES User(userEmail)
);

CREATE TABLE Contains (
   playlistID INT NOT NULL,
   songID INT,
   albumName VARCHAR(255),
   artist VARCHAR(255),
   PRIMARY KEY (playlistID, songID, albumName, artist),
   FOREIGN KEY (playlistID) REFERENCES Playlist(playlistID),
   FOREIGN KEY (songID) REFERENCES Song(songID),
   FOREIGN KEY (albumName, artist) REFERENCES Album(albumName, artist)
);

CREATE TABLE HasStyleOf (
   playlistID INT NOT NULL,
   genreName VARCHAR(255),
   PRIMARY KEY (playlistID, genreName),
   FOREIGN KEY (playlistID) REFERENCES Playlist(playlistID),
   FOREIGN KEY (genreName) REFERENCES Genre(genreName)
);

CREATE TABLE HasStyle (
   artistName VARCHAR(255) NOT NULL,
   genreName VARCHAR(255),
   PRIMARY KEY (artistName, genreName),
   FOREIGN KEY (artistName) REFERENCES Artist(artistName),
   FOREIGN KEY (genreName) REFERENCES Genre(genreName)
);

CREATE TABLE Favorites (
   favoritedBy VARCHAR(255),
   songID INT,
   albumName VARCHAR(255),
   artist VARCHAR(255),
   PRIMARY KEY (favoritedBy, songID, albumName, artist),
   FOREIGN KEY (favoritedBy) REFERENCES User(userEmail),
   FOREIGN KEY (songID) REFERENCES Song(songID),
   FOREIGN KEY (albumName, artist) REFERENCES Album(albumName, artist)
);

CREATE TABLE Creates (
   songID INT,
   albumName VARCHAR(255),
   artist VARCHAR(255),
   PRIMARY KEY (songID, albumName, artist),
   FOREIGN KEY (songID) REFERENCES Song(songID),
   FOREIGN KEY (albumName, artist) REFERENCES Album(albumName, artist),
   FOREIGN KEY (artist) REFERENCES Artist(artistName)
);
```

INSERT STATEMENTS

```
INSERT INTO User_Birthday VALUES ('1995-06-15', 28);
INSERT INTO User_Birthday VALUES ('1989-02-22', 34);
INSERT INTO User_Birthday VALUES ('2001-11-30', 22);
INSERT INTO User_Birthday VALUES ('1990-04-04', 33);
INSERT INTO User_Birthday VALUES ('1999-08-08', 24);
INSERT INTO User_Birthday VALUES ('1985-01-01', 38);
INSERT INTO User_Birthday VALUES ('1998-12-25', 25);

INSERT INTO User VALUES ('JohnDoe', 'john.doe@email.com', '1995-06-15');
INSERT INTO User VALUES ('AliceSmith', 'alice.smith@email.com', '1989-02-22');
INSERT INTO User VALUES ('EveJackson', 'eve.jackson@email.com', '2001-11-30');
INSERT INTO User VALUES ('BobMartinez', 'bob.martinez@email.com', '1990-04-04');
INSERT INTO User VALUES ('ChrisGreen', 'chris.green@email.com', '1999-08-08');
INSERT INTO User VALUES ('DavidBrown', 'david.brown@email.com', '1985-01-01');
INSERT INTO User VALUES ('EmmaWhite', 'emma.white@email.com', '1998-12-25');

INSERT INTO Event_Ticket VALUES ('Madison Square Garden', 70);
INSERT INTO Event_Ticket VALUES ('O2 Arena', 80);
INSERT INTO Event_Ticket VALUES ('Staples Center', 90);
INSERT INTO Event_Ticket VALUES ('Red Rocks Amphitheatre', 100);
```

```sql
INSERT INTO Event_Ticket VALUES ('Rose Bowl', 85);
INSERT INTO Event_Ticket VALUES ('Tokyo Dome', 95);
INSERT INTO Event_Ticket VALUES ('Wembley Stadium', 100);

INSERT INTO Event_Attendance VALUES (20000, 70);
INSERT INTO Event_Attendance VALUES (18000, 80);
INSERT INTO Event_Attendance VALUES (25000, 90);
INSERT INTO Event_Attendance VALUES (9500, 100);
INSERT INTO Event_Attendance VALUES (88000, 85);
INSERT INTO Event_Attendance VALUES (55000, 95);
INSERT INTO Event_Attendance VALUES (90000, 100);

INSERT INTO Event_Venue VALUES ('Madison Square Garden', 20000);
INSERT INTO Event_Venue VALUES ('O2 Arena', 18000);
INSERT INTO Event_Venue VALUES ('Staples Center', 25000);
INSERT INTO Event_Venue VALUES ('Red Rocks Amphitheatre', 9500);
INSERT INTO Event_Venue VALUES ('Rose Bowl', 88000);
INSERT INTO Event_Venue VALUES ('Tokyo Dome', 55000);
INSERT INTO Event_Venue VALUES ('Wembley Stadium', 90000);

INSERT INTO Event_Type VALUES ('Madison Square Garden', 'Concert');
INSERT INTO Event_Type VALUES ('O2 Arena', 'Concert');
INSERT INTO Event_Type VALUES ('Staples Center', 'Basketball Game');
INSERT INTO Event_Type VALUES ('Red Rocks Amphitheatre', 'Outdoor Concert');
INSERT INTO Event_Type VALUES ('Rose Bowl', 'Football Game');
INSERT INTO Event_Type VALUES ('Tokyo Dome', 'Baseball Game');
INSERT INTO Event_Type VALUES ('Wembley Stadium', 'Soccer Match');

INSERT INTO Event VALUES ('Ariana Grande Concert', '2021-05-10', 'Madison Square
Garden');
INSERT INTO Event VALUES ('Ed Sheeran Concert', '2019-07-22', 'O2 Arena');
INSERT INTO Event VALUES ('NBA Finals', '2018-06-05', 'Staples Center');
INSERT INTO Event VALUES ('Taylor Swift Concert', '2020-08-15', 'Red Rocks
Amphitheatre');
INSERT INTO Event VALUES ('Super Bowl', '2017-02-05', 'Rose Bowl');
INSERT INTO Event VALUES ('MLB International Series', '2016-03-30', 'Tokyo Dome');
INSERT INTO Event VALUES ('UEFA Championship', '2021-08-11', 'Wembley Stadium');

INSERT INTO Artist_Viewership VALUES (5000000, 1000000);
INSERT INTO Artist_Viewership VALUES (3000000, 750000);
INSERT INTO Artist_Viewership VALUES (7000000, 1400000);
INSERT INTO Artist_Viewership VALUES (4000000, 850000);
INSERT INTO Artist_Viewership VALUES (2500000, 650000);
INSERT INTO Artist_Viewership VALUES (6000000, 1200000);
INSERT INTO Artist_Viewership VALUES (3500000, 700000);

INSERT INTO Artist VALUES ('Ariana Grande', 'USA', 'American singer and actress',
5000000);
INSERT INTO Artist VALUES ('Ed Sheeran', 'UK', 'British singer-songwriter', 3000000);
INSERT INTO Artist VALUES ('Taylor Swift', 'USA', 'American singer-songwriter', 7000000);
INSERT INTO Artist VALUES ('Billie Eilish', 'USA', 'American singer and songwriter',
4000000);
INSERT INTO Artist VALUES ('Post Malone', 'USA', 'American rapper and singer', 2500000);
INSERT INTO Artist VALUES ('Drake', 'Canada', 'Canadian rapper and singer', 6000000);
INSERT INTO Artist VALUES ('Dua Lipa', 'UK', 'English singer and songwriter', 3500000);

INSERT INTO Rapper_Time VALUES ('Trap', 2020);
INSERT INTO Rapper_Time VALUES ('Drill', 2018);
INSERT INTO Rapper_Time VALUES ('Emo Rap', 2017);
INSERT INTO Rapper_Time VALUES ('Boom Bap', 1990);
INSERT INTO Rapper_Time VALUES ('Hyphy', 2000);
INSERT INTO Rapper_Time VALUES ('Mumble Rap', 2016);
INSERT INTO Rapper_Time VALUES ('Conscious', 2005);

INSERT INTO Rapper_Origin VALUES ('Trap', 'Atlanta');
INSERT INTO Rapper_Origin VALUES ('Drill', 'Chicago');
INSERT INTO Rapper_Origin VALUES ('Emo Rap', 'Los Angeles');
INSERT INTO Rapper_Origin VALUES ('Boom Bap', 'New York');
INSERT INTO Rapper_Origin VALUES ('Hyphy', 'Bay Area');
INSERT INTO Rapper_Origin VALUES ('Mumble Rap', 'Atlanta');
INSERT INTO Rapper_Origin VALUES ('Conscious', 'New York');

INSERT INTO Rapper VALUES ('Post Malone', 'Trap');
INSERT INTO Rapper VALUES ('Drake', 'Emo Rap');


INSERT INTO Instrument VALUES ('Guitar', 'Spain');
INSERT INTO Instrument VALUES ('Piano', 'Italy');
INSERT INTO Instrument VALUES ('Violin', 'Italy');
INSERT INTO Instrument VALUES ('Trumpet', 'Germany');
INSERT INTO Instrument VALUES ('Drums', 'Africa');
INSERT INTO Instrument VALUES ('Saxophone', 'Belgium');
INSERT INTO Instrument VALUES ('Flute', 'China');

INSERT INTO Plays_Experience VALUES (1, 'Beginner');
INSERT INTO Plays_Experience VALUES (3, 'Intermediate');
INSERT INTO Plays_Experience VALUES (5, 'Experienced');
INSERT INTO Plays_Experience VALUES (7, 'Expert');
INSERT INTO Plays_Experience VALUES (10, 'Master');
INSERT INTO Plays_Experience VALUES (12, 'Grandmaster');
INSERT INTO Plays_Experience VALUES (15, 'Legend');

INSERT INTO Plays VALUES ('Billie Eilish', 'Vocal', 6);
INSERT INTO Plays VALUES ('Post Malone', 'Guitar', 7);
INSERT INTO Plays VALUES ('Dua Lipa', 'Vocal', 8);
INSERT INTO Plays VALUES ('The Weeknd', 'Keyboard', 7);
INSERT INTO Plays VALUES ('Tones and I', 'Vocal', 5);


INSERT INTO Genre VALUES ('Pop', 'USA');
INSERT INTO Genre VALUES ('Rock', 'UK');
INSERT INTO Genre VALUES ('Rap', 'USA');
INSERT INTO Genre VALUES ('Country', 'USA');
INSERT INTO Genre VALUES ('Classical', 'Europe');
INSERT INTO Genre VALUES ('Jazz', 'USA');
INSERT INTO Genre VALUES ('Blues', 'USA');

INSERT INTO Album VALUES ('Divide', 'Ed Sheeran', '2017-03-03');
INSERT INTO Album VALUES ('Lover', 'Taylor Swift', '2019-08-23');
INSERT INTO Album VALUES ('Hollywood's Bleeding', 'Post Malone', '2019-09-06');
INSERT INTO Album VALUES ('Thank U, Next', 'Ariana Grande', '2019-02-08');
INSERT INTO Album VALUES ('Scorpion', 'Drake', '2018-06-29');
INSERT INTO Album VALUES ('When We All Fall Asleep, Where Do We Go?', 'Billie Eilish',
'2019-03-29');
INSERT INTO Album VALUES ('Future Nostalgia', 'Dua Lipa', '2020-03-27');

INSERT INTO Song_Mood VALUES ('Happy', 120);
INSERT INTO Song_Mood VALUES ('Sad', 60);
INSERT INTO Song_Mood VALUES ('Energetic', 140);
INSERT INTO Song_Mood VALUES ('Relaxed', 80);
INSERT INTO Song_Mood VALUES ('Romantic', 90);
INSERT INTO Song_Mood VALUES ('Angry', 150);
INSERT INTO Song_Mood VALUES ('Melancholic', 70);

INSERT INTO Song_Genre VALUES ('Pop', 100);
INSERT INTO Song_Genre VALUES ('Rock', 130);
INSERT INTO Song_Genre VALUES ('Rap', 120);
INSERT INTO Song_Genre VALUES ('Country', 110);
INSERT INTO Song_Genre VALUES ('Classical', 60);
INSERT INTO Song_Genre VALUES ('Jazz', 90);
INSERT INTO Song_Genre VALUES ('Blues', 75);

INSERT INTO Song_Vibe VALUES ('Rock', 'Empowering');
INSERT INTO Song_Vibe VALUES ('Country', 'Melancholic');
INSERT INTO Song_Vibe VALUES ('Electronic', 'Euphoric');
INSERT INTO Song_Vibe VALUES ('R&B', 'Soulful');
INSERT INTO Song_Vibe VALUES ('Jazz', 'Relaxing');


INSERT INTO Song_Viewership VALUES ('Thank U, Next', 'Ariana Grande', 4200000);
INSERT INTO Song_Viewership VALUES ('Rockstar', 'Post Malone', 4800000);
INSERT INTO Song_Viewership VALUES ('Blinding Lights', 'The Weeknd', 4600000);
INSERT INTO Song_Viewership VALUES ('Dance Monkey', 'Tones and I', 4000000);
INSERT INTO Song_Viewership VALUES ('7 Rings', 'Ariana Grande', 4100000);


INSERT INTO Song VALUES (3, 'Rockstar', '2017-09-15', 'Post Malone', 'Beerbongs &
Bentleys', 'Rap');
INSERT INTO Song VALUES (4, 'Blinding Lights', '2019-11-29', 'The Weeknd', 'After Hours',
'Pop');
INSERT INTO Song VALUES (5, 'Dance Monkey', '2019-05-10', 'Tones and I', 'The Kids Are
Coming', 'Pop');
INSERT INTO Song VALUES (6, '7 Rings', '2019-01-18', 'Ariana Grande', 'Thank U, Next',
'Pop');
INSERT INTO Song VALUES (7, 'Thank U, Next', '2019-01-03', 'Ariana Grande', 'Thank U,
Next', 'Pop');


INSERT INTO Thread VALUES (1, 'john.doe@gmail.com', 'john.doe@gmail.com', 'Loving the
new Ed Sheeran Album!', '2020-01-05');
```

```
INSERT INTO Thread VALUES (2, 'jane.smith@gmail.com', 'john.doe@gmail.com', 'Totally
agree! "Shape of You" is my favorite.', '2020-01-06');
INSERT INTO Thread VALUES (3, 'alice.jones@gmail.com', 'john.doe@gmail.com', 'I prefer
his older songs, but this one is good too.', '2020-01-07');
INSERT INTO Thread VALUES (4, 'bob.james@gmail.com', 'jane.smith@gmail.com', 'Have
you heard Billie Eilish\'s new song?', '2020-02-05');
INSERT INTO Thread VALUES (5, 'charlie.brown@gmail.com', 'bob.james@gmail.com', 'Yes!
"Bad Guy" is amazing!', '2020-02-06');
INSERT INTO Thread VALUES (6, 'david.johnson@gmail.com', 'charlie.brown@gmail.com',
'It's one of her best for sure.', '2020-02-07');
INSERT INTO Thread VALUES (7, 'emily.wilson@gmail.com', 'david.johnson@gmail.com',
'Can't wait for her next album.', '2020-02-08');

INSERT INTO FriendsWith VALUES ('john.doe@gmail.com', 'jane.smith@gmail.com');
INSERT INTO FriendsWith VALUES ('john.doe@gmail.com', 'alice.jones@gmail.com');
INSERT INTO FriendsWith VALUES ('jane.smith@gmail.com', 'alice.jones@gmail.com');
INSERT INTO FriendsWith VALUES ('bob.james@gmail.com', 'charlie.brown@gmail.com');
INSERT INTO FriendsWith VALUES ('charlie.brown@gmail.com',
'david.johnson@gmail.com');
INSERT INTO FriendsWith VALUES ('david.johnson@gmail.com', 'emily.wilson@gmail.com');
INSERT INTO FriendsWith VALUES ('emily.wilson@gmail.com', 'bob.james@gmail.com');

INSERT INTO Follows VALUES ('john.doe@gmail.com', 'Ed Sheeran');
INSERT INTO Follows VALUES ('jane.smith@gmail.com', 'Ed Sheeran');
INSERT INTO Follows VALUES ('alice.jones@gmail.com', 'Billie Eilish');
INSERT INTO Follows VALUES ('bob.james@gmail.com', 'Billie Eilish');
INSERT INTO Follows VALUES ('charlie.brown@gmail.com', 'Dua Lipa');
INSERT INTO Follows VALUES ('david.johnson@gmail.com', 'Post Malone');
INSERT INTO Follows VALUES ('emily.wilson@gmail.com', 'Ariana Grande');

INSERT INTO HasAttended VALUES ('john.doe@gmail.com', 'Summer Fest', '2020-06-15');
INSERT INTO HasAttended VALUES ('jane.smith@gmail.com', 'Rock Concert', '2020-07-10');
INSERT INTO HasAttended VALUES ('alice.jones@gmail.com', 'Pop Nights', '2020-08-05');
INSERT INTO HasAttended VALUES ('bob.james@gmail.com', 'Jazz Evening', '2020-09-12');
INSERT INTO HasAttended VALUES ('charlie.brown@gmail.com', 'Hip Hop Mania',
'2020-10-02');
INSERT INTO HasAttended VALUES ('david.johnson@gmail.com', 'Country Roads',
'2020-11-17');
INSERT INTO HasAttended VALUES ('emily.wilson@gmail.com', 'Metal Headbang',
'2020-12-03');
```

```
INSERT INTO Singer VALUES ('Ed Sheeran', 'Tenor');
INSERT INTO Singer VALUES ('Ariana Grande', 'Soprano');
INSERT INTO Singer VALUES ('Billie Eilish', 'Alto');
INSERT INTO Singer VALUES ('Post Malone', 'Baritone');
INSERT INTO Singer VALUES ('Dua Lipa', 'Mezzo-Soprano');
INSERT INTO Singer VALUES ('Taylor Swift', 'Soprano');
INSERT INTO Singer VALUES ('Drake', 'Tenor');

INSERT INTO Producer VALUES ('Pharrell Williams');
INSERT INTO Producer VALUES ('Rick Rubin');
INSERT INTO Producer VALUES ('Max Martin');
INSERT INTO Producer VALUES ('Dr. Dre');
INSERT INTO Producer VALUES ('Kenny Beats');
INSERT INTO Producer VALUES ('Metro Boomin');
INSERT INTO Producer VALUES ('Mike Will Made It');

INSERT INTO Playlist VALUES (1, 'Chill Vibes', '2020-01-01', 'john.doe@gmail.com');
INSERT INTO Playlist VALUES (2, 'Workout Pump', '2020-01-05', 'jane.smith@gmail.com');
INSERT INTO Playlist VALUES (3, 'Top Hits', '2020-01-10', 'alice.jones@gmail.com');
INSERT INTO Playlist VALUES (4, 'Relax and Sleep', '2020-01-15', 'bob.james@gmail.com');
INSERT INTO Playlist VALUES (5, 'Party Mix', '2020-01-20', 'charlie.brown@gmail.com');
INSERT INTO Playlist VALUES (6, 'Throwback Jams', '2020-01-25',
'david.johnson@gmail.com');
INSERT INTO Playlist VALUES (7, 'Daily Mix', '2020-01-30', 'emily.wilson@gmail.com');

INSERT INTO Contains VALUES (1, 1, 'Divide', 'Ed Sheeran');
INSERT INTO Contains VALUES (2, 2, 'Divide', 'Ed Sheeran');
INSERT INTO Contains VALUES (3, 3, 'When We All Fall Asleep, Where Do We Go?', 'Billie
Eilish');
INSERT INTO Contains VALUES (4, 4, 'Future Nostalgia', 'Dua Lipa');
INSERT INTO Contains VALUES (5, 5, 'Hollywood\'s Bleeding', 'Post Malone');
INSERT INTO Contains VALUES (6, 6, 'Thank U, Next', 'Ariana Grande');
INSERT INTO Contains VALUES (7, 7, 'Scorpion', 'Drake');

INSERT INTO HasStyleOf VALUES (1, 'Pop');
INSERT INTO HasStyleOf VALUES (2, 'Rock');
INSERT INTO HasStyleOf VALUES (3, 'Hip Hop');
INSERT INTO HasStyleOf VALUES (4, 'Jazz');
INSERT INTO HasStyleOf VALUES (5, 'R&B');
INSERT INTO HasStyleOf VALUES (6, 'Country');
INSERT INTO HasStyleOf VALUES (7, 'Electronic');

INSERT INTO HasStyle VALUES ('Ed Sheeran', 'Pop');
INSERT INTO HasStyle VALUES ('Ariana Grande', 'Pop');
INSERT INTO HasStyle VALUES ('Billie Eilish', 'Electronica');
INSERT INTO HasStyle VALUES ('Post Malone', 'Hip Hop');
INSERT INTO HasStyle VALUES ('Dua Lipa', 'Pop');
INSERT INTO HasStyle VALUES ('Taylor Swift', 'Country');
INSERT INTO HasStyle VALUES ('Drake', 'Hip Hop');

INSERT INTO Favorites VALUES ('john.doe@gmail.com', 1, 'Divide', 'Ed Sheeran');
INSERT INTO Favorites VALUES ('jane.smith@gmail.com', 2, 'Divide', 'Ed Sheeran');
INSERT INTO Favorites VALUES ('alice.jones@gmail.com', 3, 'When We All Fall Asleep,
Where Do We Go?', 'Billie Eilish');
INSERT INTO Favorites VALUES ('bob.james@gmail.com', 4, 'Future Nostalgia', 'Dua Lipa');
INSERT INTO Favorites VALUES ('charlie.brown@gmail.com', 5, 'Hollywood\'s Bleeding',
'Post Malone');
INSERT INTO Favorites VALUES ('david.johnson@gmail.com', 6, 'Thank U, Next', 'Ariana
Grande');
INSERT INTO Favorites VALUES ('emily.wilson@gmail.com', 7, 'Scorpion', 'Drake');

INSERT INTO Creates VALUES (1, 'Divide', 'Ed Sheeran');
INSERT INTO Creates VALUES (2, 'Divide', 'Ed Sheeran');
INSERT INTO Creates VALUES (3, 'When We All Fall Asleep, Where Do We Go?', 'Billie
Eilish');
INSERT INTO Creates VALUES (4, 'Future Nostalgia', 'Dua Lipa');
INSERT INTO Creates VALUES (5, 'Hollywood\'s Bleeding', 'Post Malone');
INSERT INTO Creates VALUES (6, 'Thank U, Next', 'Ariana Grande');
INSERT INTO Creates VALUES (7, 'Scorpion', 'Drake');
```