

Rakshit Sareen - rs5606
Lucas O'Rourke - lor215

Project 1 Writeup

INTRODUCTION:

The goal of this project is to create and design a database-backed website music service, similar to popular services such as SoundCloud or Spotify. A user can create a password protected account and search for information pertaining to Artists, Tracks, Albums, Playlists, etc. The database will keep track of the User's favorite artists and songs, based on the songs and artists they listen to most often, or rate highly.

CREATING ACCOUNTS:

The user will begin by creating an account, where they will create a unique username and provide their name, email, and residing city. The user will also enter a password to protect their information from other users.

ARTISTS, ALBUMS, PLAYLISTS, AND TRACKS:

Once the user signs up for the database, they will be able to view Artist's categorized by a artist ID (unique), name, and short description. They can also view different Tracks, which have a track ID (unique), title, duration, genre, and associated artist. These tracks may belong to Albums, or user created Playlists.

Playlists and Albums are very similar in the sense that they contain an ID (unique), title, release/creation date, and an ordered list of tracks. The difference is that Albums are associated with Artists (and created in the backend side of the database), while Playlists are created by Users on the frontend, and then translated to the backend database.

USER FUNCTIONALITIES:

Beyond making Playlists, the User also has the ability to Like an Artist, Follow another User, Play a Track, and Rate a Track. Each time one of these events occur, it is logged in the database in order to keep track of each Users preferences and favorites.

The User also has the ability to search for an Artist or Track based on a keyword, or a list of keywords. This will be handled by queries done in MySQL (shown below)

STATISTICS:

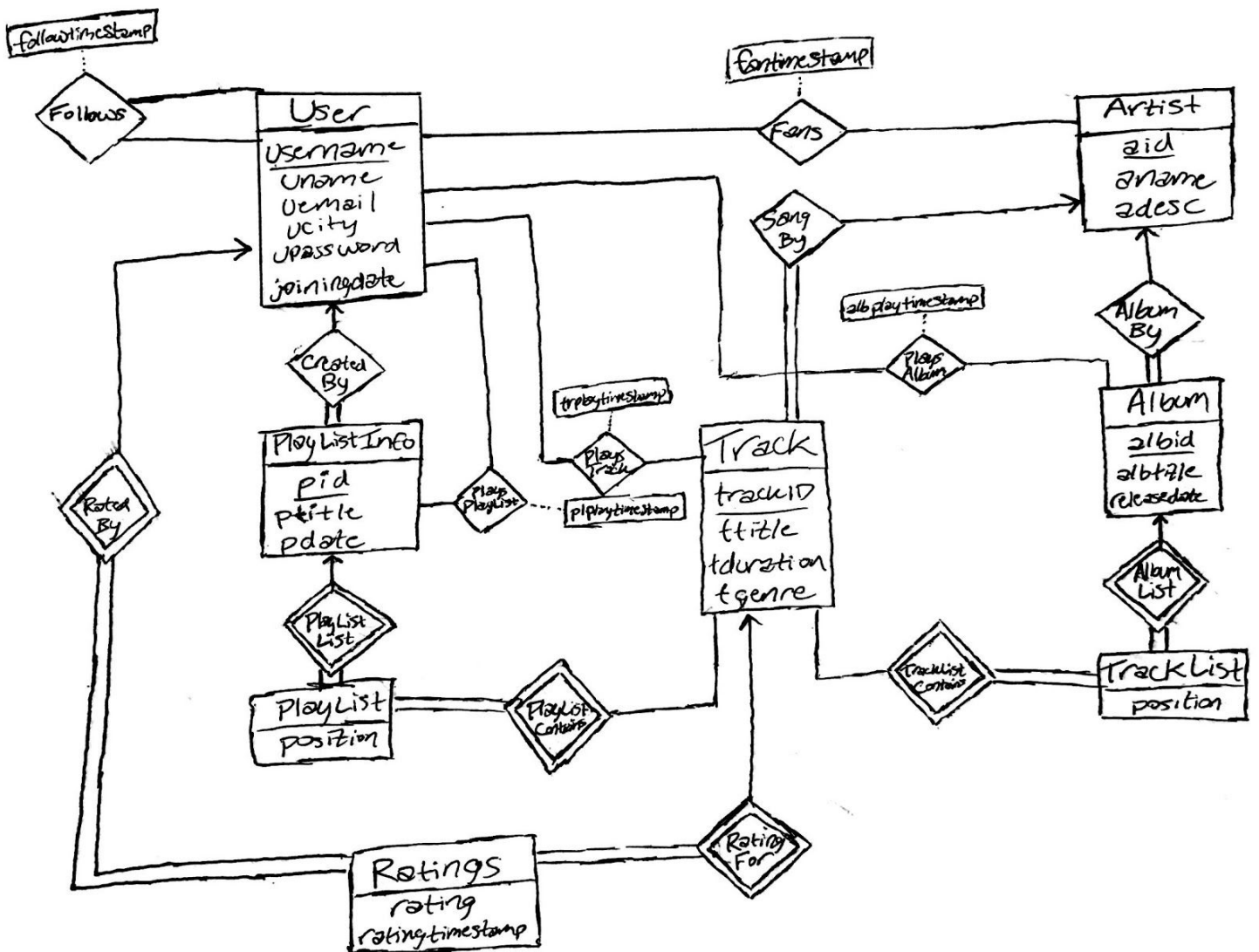
We will be keeping track of statistics in this project based on user interactions with the service, such as most popular Album, Playlist, Track, etc. This will allow for some advanced SQL queries, as well as a better overall user interface.

DEVELOPMENT PLAN:

We plan to create this web application using MySQL as the underlying database that will store all records for Users, Artists, Albums, Tracks, etc. Although the frontend aspects are still in discussion, we will be using some mix of Java and their Spring Framework in order to run the server and create the web app.

Spring supports SQL, but also provides simplified security, which will be necessary for User password encryption and protection.

ER-DIAGRAM:



We chose to keep track of each time a Track, Playlist, or Album is Played. This will allow us to keep track of the most popular Played. In our relational schema, we combined the Plays relationships into one entity (as shown below).

SCHEMA:

User(username,uname,uemail,ucity,upassword,joiningdate)

Artist(aid,aname,adesc)

Track(trackID,ttitle,tduration,tgenre,aid)

Album(albid,albtitle,releaseDate,aid)

PlaylistInfo(pid,ptitle,pdate,username)

TrackList(albid,trackID,position)

Playlist(pid,trackID,position)

Fans(username,aid,atimestamp)

Follows(username,followee,ftimestamp)

Plays(username,trackID,playtimestamp, fromAlbum,fromPlayList)

Ratings(username,trackID,rating,ratetimestamp)

FOREIGN KEY CONSTRAINTS:

- Track.aid references Artist.aid
- Album.aid references Artist.aid
- TrackList.albid references Album.albid
- TrackList.trackID references Track.trackID
- Playlist.pid references PlaylistInfo.pid
- Playlist.trackID references Track.trackID
- Fans.aid references Artist.aid
- Follows.username references User.username
- Follows.followee references User.username
- Plays.username references User.username
- Plays.fromAlbum references Album.albid
- Plays.fromPlayList references Playlist.pid
- Plays.trackID references Track.trackID
- Ratings.username references User.username
- Ratings.trackID references Track.trackID

ASSUMPTIONS:

- We assume that each Track only has one Artist (solo, band, duet, etc.) ID
- We added a fromAlbum and fromPlaylist to the Plays table in order to keep track of most popular Album or Playlist. These can be NULL if the track is played independently. Also, we are not checking while entering these two fields that the corresponding track in the row is a valid track. The website will control this.
- If a User un-Fans an Artist, un-Follows another User, or un-Rates a Track, the original record will be deleted from the database.
- We assume that a user can play only one song at a time. Due to this assumption, we have not included fromAlbum and fromList in the primary key. This can also be overcome by adding a sessionID which tracks that a user is playing a song through a session(eg, the same song in another tab of the same or another browser)

CREATE DATABASE SCHEMA:

```
CREATE TABLE `User` (  
    `username` VARCHAR(45) NOT NULL,  
    `uname` VARCHAR(45) NULL,  
    `uemail` VARCHAR(45) NULL,  
    `ucity` VARCHAR(45) NULL,  
    `upassword` VARCHAR(45) NOT NULL,  
    `joiningdate` DATETIME NOT NULL,  
    PRIMARY KEY (`username`)  
);
```

```
CREATE TABLE `Artist` (  
    `aid` INT NOT NULL,  
    `aname` VARCHAR(45) NULL,  
    `adesc` VARCHAR(45) NULL,  
    PRIMARY KEY (`aid`)  
);
```

```
CREATE TABLE `Track` (  
    `trackID` INT NOT NULL,  
    `title` VARCHAR(45) NOT NULL,  
    `tduration` INT(11) NOT NULL,  
    `tgenre` VARCHAR(10) NULL,  
    `aid` INT NOT NULL,  
    PRIMARY KEY (`trackID`),  
    FOREIGN KEY (`aid`)  
        REFERENCES `Artist` (`aid`)  
);
```

```
CREATE TABLE `Album` (  
    `albid` INT NOT NULL,  
    `albtitle` VARCHAR(45) NOT NULL,  
    `release_date` DATETIME NOT NULL,  
    `aid` INT NOT NULL,  
    PRIMARY KEY (`albid`),  
    FOREIGN KEY (`aid`)  
        REFERENCES `Artist` (`aid`)  
);
```

```
CREATE TABLE `PlaylistInfo` (  
  `pid` INT NOT NULL,  
  `ptitle` VARCHAR(45) NOT NULL,  
  `pdate` DATETIME NOT NULL,  
  `username` VARCHAR(45) NOT NULL,  
  PRIMARY KEY (`pid`),  
  FOREIGN KEY (`username`)  
    REFERENCES `User` (`username`)  
);
```

```
CREATE TABLE `TrackList` (  
  `albid` INT NOT NULL,  
  `trackID` INT NOT NULL,  
  `position` INT NOT NULL,  
  PRIMARY KEY (`albid`, `trackID`),  
  FOREIGN KEY (`albid`)  
    REFERENCES `Album` (`albid`),  
  FOREIGN KEY (`trackID`)  
    REFERENCES `Track` (`trackID`)  
);
```

```
CREATE TABLE `Playlist` (  
  `pid` INT NOT NULL,  
  `trackID` INT NOT NULL,  
  `position` INT NOT NULL,  
  PRIMARY KEY (`pid`, `trackID`),  
  FOREIGN KEY (`pid`)  
    REFERENCES `PlaylistInfo` (`pid`),  
  FOREIGN KEY (`trackID`)  
    REFERENCES `Track` (`trackID`)  
);
```

```
CREATE TABLE `Fans` (  
  `username` VARCHAR(45) NOT NULL,  
  `aid` INT NOT NULL,  
  `timestamp` DATETIME NOT NULL,  
  PRIMARY KEY (`username`, `aid`),  
  FOREIGN KEY (`username`)  
    REFERENCES `User` (`username`),  
  FOREIGN KEY (`aid`)  
    REFERENCES `Artist` (`aid`)  
);
```

```

CREATE TABLE `Follows` (
  `username` VARCHAR(45) NOT NULL,
  `followee` VARCHAR(45) NOT NULL,
  `timestamp` DATETIME NOT NULL,
  PRIMARY KEY (`username`, `followee`),
  FOREIGN KEY (`username`)
    REFERENCES `User` (`username`),
  FOREIGN KEY (`followee`)
    REFERENCES `User` (`username`)
);

```

```

CREATE TABLE `Ratings` (
  `username` VARCHAR(45) NOT NULL,
  `trackID` INT NOT NULL,
  `rating` INT NOT NULL,
  PRIMARY KEY (`username`, `trackID`),
  FOREIGN KEY (`username`)
    REFERENCES `User` (`username`),
  FOREIGN KEY (`trackID`)
    REFERENCES `Track` (`trackID`)
);

```

```

CREATE TABLE `Plays` (
  `username` VARCHAR(45) NOT NULL,
  `trackID` INT NOT NULL,
  `time` DATETIME NOT NULL,
  `fromAlbum` INT NULL,
  `fromList` INT NULL,
  PRIMARY KEY (`username`, `trackID`, `time`),
  FOREIGN KEY (`username`)
    REFERENCES `User` (`username`),
  FOREIGN KEY (`trackID`)
    REFERENCES `Track` (`trackID`),
  FOREIGN KEY (`fromAlbum`)
    REFERENCES `Album` (`albid`),
  FOREIGN KEY (`fromList`)
    REFERENCES `playlistinfo` (`pid`)
);

```

SAMPLE DATA:

mysql> select * from user;

username	uname	uemail	ucity	upassword	joiningdate
adam.lambert	Adam Lambert	adam.lambert@gmail.com	Brooklyn	password@123	2016-01-03 11:20:00
aman.madaan	Aman Madaan	aman.madaan@gmail.com	Brooklyn	password@123	2016-01-03 11:05:00
divya.sharma	Divya Sharma	divya.sharma@gmail.com	Brooklyn	password@123	2016-01-03 11:00:00
raks.sareen	Rakshit Sareen	raks.sareen@gmail.com	Brooklyn	password@123	2016-01-13 11:20:00
sameer.tuli	Sameer Tuli	sameer.tuli@gmail.com	Brooklyn	password@123	2016-01-03 11:00:00

Records from the User table will be inserted via the frontend's signup page. A user will be prompted to select a username, enter their name, email, city, choose a password, and a timestamp will be associated with the entry created.

mysql> select * from Artist;

aid	aname	adesc
0	Megha Sisaudia	Jazz Player, Indian
1	Rohit Sisaudia	Vocalist
2	Amit Tandon	Newbie
3	Eminem	Rapper, American Artist
4	BackSteet Boys	Inactive
5	Lehman Brothers	NULL

Artists will be implemented by us, the application developers. Each artist has a unique aid (artist ID). Each artist will also have an associated name and description in order for a better frontend interface and more efficient querying in the backend implementation.

mysql> select * from Track;

trackID	title	tduration	tgenre	aid
0	Lose Yourself	245	Rock	3
1	Stan	266	Rock	3
2	Without Me	256	Soft Rock	3
3	Ziddi Hun	256	Soft Rock	1
4	Tu Hai	246	Soft Rock	1
5	Naraaz Mujhse	246	Jazz	2
6	Naraaz Tujhse	246	Jazz	2
7	Tere Bin Jee Lenge	256	Rock	2
8	Mohabatein Iutaunga	256	Rock	4
9	Tere Bin Jee Lenge Remix	276	Rock	4
10	Tere Naina	266	Country	4
11	Abhimaan	246	Country	5

Tracks will also be inserted by the application developers. A track has a unique trackID, and an associated non unique title, duration, and genre. It also has a foreign key aid that points to the artist that sang the track.

mysql> select * from album;

albid	albttitle	release_date	aid
0	Never Back Down	2009-01-03 11:00:00	1
1	Recovery	2010-01-03 11:00:00	1
2	The Slim	2011-01-03 11:00:00	3
3	The Slim	2011-01-03 11:00:00	3
4	Infinite	2011-10-12 11:00:00	4
5	Infinite Part 2	2011-10-12 11:00:00	4
6	Get The Guns	2011-11-12 11:00:00	4
7	8 Mile	2013-12-12 11:00:00	5

The album table is another table that will be filled by the application developers. Each album has a unique ID, and an album title and release date. The album is associated with an artist using a foreign key referencing aid in the artist table

mysql> select * from tracklist;

+-----+-----+-----+		
albid	trackID	position
+-----+-----+-----+		
0	5	1
0	6	2
0	7	3
1	1	1
1	2	2
1	11	3
2	0	1
2	3	2
2	11	3
3	4	1
3	8	2
3	9	3
4	5	1
4	10	2
4	11	3
5	4	1
5	6	2
5	9	3
6	2	1
6	6	2
6	10	3
7	0	1
7	1	2
+-----+-----+-----+		

The tracklist table references the album table and fills each album with its associated tracks. If you want to view what tracks are in an album, you would use the foreign key albid to find out which album you are viewing, and then the foreign key trackID tells you which song is in that album. Each track in an album has a unique position.

mysql> select * from playlistinfo;

+-----+-----+-----+-----+			
pid	p title	pdate	username
+-----+-----+-----+-----+			
0	Office	2009-01-03 11:00:00	adam.lambert
1	Workout	2009-01-03 12:00:00	adam.lambert
2	Driving	2009-11-03 12:00:00	adam.lambert
3	Driving	2009-11-13 12:00:00	aman.madaan
4	Studying	2009-11-13 12:00:00	aman.madaan
5	Inspirational	2009-10-13 14:00:00	divya.sharma
6	Study	2009-10-13 14:00:00	divya.sharma
7	Travel	2010-10-13 14:00:00	divya.sharma
8	Office	2010-09-13 14:00:00	divya.sharma
9	Untitled	2010-09-13 14:00:00	raks.sareen
10	Workout	2010-09-13 14:00:00	raks.sareen
11	Car	2010-09-11 14:00:00	sameer.tuli
+-----+-----+-----+-----+			

Playlists are very similar to albums. They have a playlist ID (pid) and a title, creation date, and an associated username for the user that created it. The creation of playlists will be done in the front end by the user, and the data to fill this table will be pulled from the frontend scripts.

mysql> select * from playlist;

+-----+-----+-----+		
pid	trackID	position
+-----+-----+-----+		
0	1	1
0	2	2
0	3	3
1	4	1
1	5	2
1	6	3
2	7	1
2	8	2
2	9	3
3	10	1
3	11	2
4	2	1
4	4	2
4	6	3
4	8	4
5	5	1
5	9	2
5	10	3
5	11	4
6	3	1
6	8	2
7	9	1
7	11	2
8	2	1
8	8	2
9	4	1
9	7	2
9	10	3
10	6	1
10	8	2
10	11	3
11	3	1
11	5	2
11	9	3
+-----+-----+-----+		

The playlist table acts the same as the tracklist table. It references the playlistinfo table and fills each playlist with its associated tracks. If you want to view what tracks are in a playlist, you would use the foreign key pid to find out which playlist you are viewing, and then the foreign key trackID tells you which song is in that playlist. Each track in a playlist has a unique position.

mysql> select * from fans;

username	aid	timestamp
adam.lambert	0	2016-01-03 11:20:00
aman.madaan	1	2016-01-03 11:20:00
divya.sharma	2	2016-01-03 11:20:00
raks.sareen	2	2016-01-03 11:20:00
raks.sareen	3	2016-01-03 11:20:00
raks.sareen	4	2016-01-03 11:20:00
sameer.tuli	2	2016-01-03 11:20:00
sameer.tuli	4	2016-01-03 11:20:00

The fans table allows a user to be a fan of an artist. This table is filled in the front end when a user chooses to “like” or “become a fan” of an artist. That user’s username is then placed in this table with the artist ID that they choose and a timestamp is associated with it.

mysql> select * from plays;

username	trackID	time	fromAlbum	fromList
adam.lambert	0	2016-01-03 11:12:00	NULL	NULL
aman.madaan	0	2016-01-03 11:12:00	NULL	NULL
divya.sharma	0	2016-01-03 11:12:00	NULL	NULL
divya.sharma	9	2017-04-03 11:12:00	0	NULL
divya.sharma	9	2017-04-03 11:12:01	NULL	0
raks.sareen	0	2016-01-03 11:12:00	NULL	NULL
raks.sareen	8	2017-04-03 11:12:01	NULL	NULL
raks.sareen	9	2017-04-03 11:12:01	NULL	0
sameer.tuli	0	2016-01-03 11:12:00	NULL	NULL
sameer.tuli	1	2017-01-03 11:12:00	NULL	NULL
sameer.tuli	10	2017-01-03 11:12:00	NULL	NULL
sameer.tuli	11	2017-01-03 11:12:00	NULL	NULL

For most outputs, fromAlbum and fromList hold NULL values. But if a User chose to play a Track directly from an Album or List (say in the 4th row) then one of those values will not be NULL. This allows us to keep track of the most popular Albums and PlayLists.

mysql> select * from ratings;

username	trackID	rating
adam.lambert	0	3
adam.lambert	1	4
adam.lambert	9	4
adam.lambert	11	4
divya.sharma	7	2
divya.sharma	8	2
divya.sharma	10	2
divya.sharma	11	1
raks.sareen	7	4
raks.sareen	8	3
raks.sareen	10	4
raks.sareen	11	4
sameer.tuli	7	2
sameer.tuli	8	2
sameer.tuli	10	2
sameer.tuli	11	1

The ratings table allows a user to rate a song on a scale of 1-5. This will be handled by a form in the frontend. The signed in user's username will then be added to the table with the trackID that they rated, and the rating that they chose.

mysql> select * from follows;

username	followee	timestamp
aman.madaan	adam.lambert	2016-01-03 11:12:00
divya.sharma	adam.lambert	2016-01-03 11:12:00
raks.sareen	adam.lambert	2016-01-03 11:12:00
raks.sareen	aman.madaan	2016-01-03 11:12:00
raks.sareen	divya.sharma	2016-01-03 11:12:00
raks.sareen	sameer.tuli	2016-01-03 11:12:00
sameer.tuli	adam.lambert	2016-01-03 11:12:00
sameer.tuli	aman.madaan	2016-01-03 11:12:00

The follows table allows one user to follow another user. When a user follows another user, both of their usernames are placed in a record in the table, along with a timestamp of when the event occurred.

SQL QUERIES AND OUTPUTS:

1. **Create a record for a new user account, with a name, a login name, and a password**

```
INSERT INTO `User` (`username`, `uname`, `uemail`, `ucity`, `upassword`, `joiningdate`)
VALUES ('divya.sharma','Divya
Sharma','divya.sharma@gmail.com','Brooklyn','password@123','2016-1-3 11:00:00');
NO OUTPUT
```

2. **For each artist, list their ID, name, and how many times their tracks have been played by users.**

```
select aid, aname, count(*) from artist natural join track natural join plays group by aid;
```

```
+-----+-----+-----+
| aid | aname           | count(*) |
+-----+-----+-----+
| 3   | Eminem          | 6        |
| 4   | BackSteet Boys | 5        |
| 5   | Lehman Brothers | 1        |
+-----+-----+-----+
```

3. **List all artists that are mainly playing Jazz, meaning that at least half of their tracks are of genre Jazz**

```
select aid from ( select aid,count(*) as tcnt from track group by aid) a natural join (select
aid,count(*) as jcnt from track where tgenre = 'Jazz' group by aid)b where jcnt/tcnt>.5;
```

```
+-----+
| aid |
+-----+
| 2   |
+-----+
```

As shown in the sample data, aid = 2 (Amit Tandon) is the only artist that is playing at least half of their tracks as Jazz.

4. **Insert a new rating given by a user for a track**

```
Insert into `ratings`(`username`,`trackID`,`rating`) Values ('sameer.tuli',1,3)
NO OUTPUT
```

5. **For a particular user, say “NancyInQueens”, list all playlists that were made by users that she follows**

```
select pid,ptitle
from (select followee from follows where username ='sameer.tuli')a join playlistinfo on
a.followee = playlistinfo.username;
```

pid	ptitle
0	Office
1	Workout
2	Driving
3	Driving
4	Studying

6. List all songs where the track title or artist title matches some set of keywords (if possible, user “contains”, or otherwise “like”, for this query)

select title from Track natural join Artist where title like '%Hai%' or aname like '%it%';

title
Ziddi Hun
Tu Hai
Naraaz Mujhse
Naraaz Tujhse
Tere Bin Jee Lenge

7. Find pairs of related artists, where two artists are related if they have many fans in common (Define this appropriately.)

```
select a.aid, b.aid
from fans as a join fans as b on (a.aid < b.aid and a.username = b.username)
group by a.aid, b.aid
having count(*) > 1
```

aid	aid
2	4

Here two artists are similar if they have common users. The count can be changed at the end so as to make the result more stringent in terms common users. (eg count(*) > 5)

PROJECT PART II

Introduction:

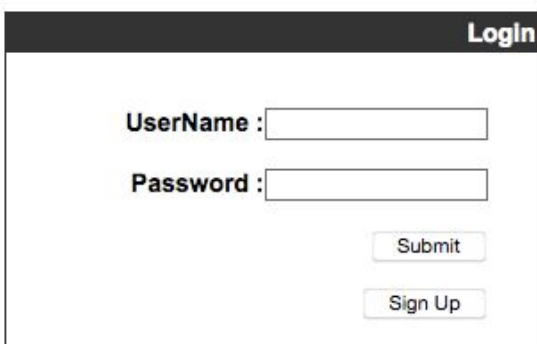
In this part of the project, we have created a frontend Web application to connect to our backend MySQL database. In the first part of the project, we intended to use Spring in order to handle the back end development along with JSP for front end, but we ultimately ended up using PHP, JQuery, and MySQL to bridge between our frontend and backend development. We incorporated the Javascript library JQuery to run scripted functions within our PHP files. JQuery acts as a bridge for us to send data and run other PHP files without completely redirecting to a new page.

The website incorporates various user case scenarios that we cater to. We can Like an artist, play a song, Follow a user, sign in, create a profile, create a playlist, add tracks to playlist. Below we provide the user flow, detailed explanations of functionalities, and will cover the decisions that we made while designing our application.

In our explanations below, it is important to note that we used the same sample data that we provided above. We included a SQL script below at the bottom that can be copy and pasted into any MySQL interface and create the exact same sample data as us.

Logging In:

The flow of the website starts with a user either signing into the system, if their information is already in our system's backend database, or by creating a profile and then redirecting to the Welcome Page. If a user signs up, then a query inserts the given information into the database as a new User record.



A screenshot of a web form titled "Login" in a dark header. The form contains two input fields: "UserName :" and "Password :". Below these fields are two buttons: "Submit" and "Sign Up".



A screenshot of a web form titled "Sign Up" in a dark header. The form contains five input fields: "UserName*", "Name :", "City :", "Password*", and "Confirm Password*", followed by "Email*:". Below these fields is a single "Submit" button.

Welcome Page:

The welcome page uses specialized SQL queries that are catered to each User. We are able to keep track of individual interests and listening habits for each user based on the way that they interact with our site. For example, the user is able to pick up where they left off and see the most recently played 5 tracks. They are also able to see the top 3 (easily changed if we wanted to display more) current most popular songs and play directly from their home page. We also show the users that they follow.

One of the functionality that we have provided to the user is a search bar. A user can submit a keyword into the search bar, and then we present the user with multiple options by querying every Track, Album, Artist, etc. to see if there is a match with their search. The search query is used to retrieve information from the database which might be relevant to the user. For eg: Users, artists and Tracks that are related to the search query. For example, if a user entered “Jazz” into the search bar, all songs with genre Jazz, all albums or artists that may have Jazz in their name or description, and even playlists like “Jazzercise” will be presented to the user.

Once the user inputs a keyword in the search bar and clicks submit, the database will execute the queries described above and bring a user to a browsing page that displays the top results.

Welcome divya.sharma

[Sign Out](#)

Recently Played:

Track Name	Play
Lose Yourself	<input type="button" value="Play"/>
Stan	<input type="button" value="Play"/>
Naraaz Mujhse	<input type="button" value="Play"/>
Mohabatein lutaunga	<input type="button" value="Play"/>
Tere Bin Jee Lenge Remix	<input type="button" value="Play"/>

Top Most Played:

Track Name	Play
Lose Yourself	<input type="button" value="Play"/>
Tere Bin Jee Lenge Remix	<input type="button" value="Play"/>
Stan	<input type="button" value="Play"/>

Users You follow




Name	Username
Adam Lambert	adam.lambert

Results Page:

The results page is where the user has the option to query on any song, artist, album, playlist, etc. within our database. This page allows the user to browse and discover new music based on simple key words or phrases. Since this page is so important for a user, we spent the most time implementing it and writing detailed queries that allows the user to navigate through our entire application within this page.

TRACKS: From this results page, the user has many options. For simplicity, and in order to display a full range of functionalities, all screenshots will be used with the keyword “a”. First, a list of tracks can be found. The top 3 are shown below. From these tracks, the user can play, rate, or add the track to a playlist. These 3 functions will be covered in detail later.

Tracks:

Track Name	Artist Name	Duration	Play	Rate	Add to Playlist
Stan	Eminem	4m 26s	Play	Rate Track  Submit Rating	Add To Playlist
Ziddi Hun	Rohit Sisaudia	4m 16s	Play	Rate Track  Submit Rating	Add To Playlist
Tu Hai	Rohit Sisaudia	4m 6s	Play	Rate Track  Submit Rating	Add To Playlist

Artists: The user will also see a list of artists that match their keyword search. As you can see below, Each artist has an attached hyperlink to it so the user can click on that artist and view their complete discography (albums and tracks that they have been a part of). The results of clicking on an artist (Eminem) are shown below. Each page also has a button where they can return home as well.

Artists:

Artist Name
Megha Sisaudia
Rohit Sisaudia
Amit Tandon
Eminem
BackStreet Boys
Lehman Brothers

About Eminem

Tracks:

Track Name	Artist Name	Duration	Play
Lose Yourself	Eminem	4m 5s	Play
Stan	Eminem	4m 26s	Play
Without Me	Eminem	4m 16s	Play

Albums:

Album Name	Artist
The Slim	Eminem
The Slim	Eminem

[Home](#)

Albums/Playlists: Another important displayed feature on the search page, is the ability to view albums as well as user created playlists. Within these tables, a user can click on an album or playlist and go view all of the tracks that are featured on them. The user can also click on the artist and it will take them to a page like the one shown above. The tracklist of the album titled “Never Back Down” is shown below. The user can play that song from either the playlist or album. This is an important feature that we wanted to add, because with the play button implemented within an album or playlist, we are able to keep track of the most played albums/playlists. This would allow us to do in depth analysis in the future if we wanted to figure out in order to find out which albums have the most streams.

Albums:

Album Name	Artist
Never Back Down	Rohit Sisaudia
Recovery	Rohit Sisaudia
Infinite	BackSteet Boys
Infinite Part 2	BackSteet Boys
Get The Guns	BackSteet Boys
8 Mile	Lehman Brothers

Playlists:

Playlist Name	Created By User
Inspirational	divya.sharma
Travel	divya.sharma
Car	sameer.tuli

Never Back Down Track List

Track Name	Artist	Duration	Play
Naraaz Mujhse	Amit Tandon	4m 6s	Play
Naraaz Tujhse	Amit Tandon	4m 6s	Play
Tere Bin Jee Lenge	Amit Tandon	4m 16s	Play

[Home](#)

There are more features on the search page, but these features fall into the category of user functionalities. We will explain the process of playing and rating a song, adding a song to a playlist, and following another user in the section below.

User Functionalities:

Plays: As described above the user has the ability to play songs from the search page, artist information page, album track list, and playlist track list pages. In order to save space, MP3 files are not stored in our database, so when a user clicks on the play button, a confirmation window comes up (as shown below). The backend PHP file then sends a query to our database inserting the record of that track being played by the logged in user.

Tracks:

Track Name	Artist Name	Duration	Play	Rate	Add to Playlist
Stan	Eminem	4m 26s			Add To Playlist
Ziddi Hun	Rohit Sisaudia	4m 16s			Add To Playlist
Tu Hai	Rohit Sisaudia	4m 6s	Play	Rate Track Submit Rating	Add To Playlist

Rate Track: The user also has the ability to rate a track on a scale of 1-5 stars. This allows us to keep track of things like which songs are highest rated or most popular. The user submits the rating, and then a confirmation message is again displayed. That rating is also inserted into our database via a backend query through our PHP file.

Submit Rating

Rate Track

1 STAR

2 STARS

3 STARS

✓ 4 STARS

5 STARS

Add to Playlist: A playlist is very similar to an album, except rather than having an associated artist ID, each playlist has an associated username to denote who created the playlist. When a user clicks “Add to Playlist” from the search page, they are redirected to another page. This page uses the username to find all playlists that they have already created. These existing playlists will be displayed, and the user has the option to add the track to one of these playlists. The user can also create a new playlist with that track as the first entry.

Add to Playlist:

Inspirational Add

Study Add

Travel Add

Office Add

hi Add

Insert Name..

Create New Playlist

Home

Following Users: In order to create an interactive, almost social media like application, we have added the ability for a user to follow another user. Once the user clicks follow, a confirmation is displayed, and that record is added to the database. This is a very important feature for both current and future implementations. This allows us to run queries on our homepage such as “display all users that the logged in user follows” or “display all playlists that have been created by a user that I follow”. In the future, we could add implementations such as profile pages, where users could interact more in depth with other users. We could also display things in real time like “user1 is listening to track1”.

Users:

Name	Username	City	Follow
Adam Lambert	adam.lambert	Brooklyn	<input type="button" value="Follow User"/>
Aman Madaan	aman.madaan	Brooklyn	<input type="button" value="Follow User"/>
Divya Sharma	divya.sharma	Brooklyn	<input type="button" value="Follow User"/>
Rakshit Sareen	raks.sareen	Brooklyn	<input type="button" value="Follow User"/>
Sameer Tuli	sameer.tuli	Brooklyn	<input type="button" value="Follow User"/>

Design Decisions:

Simplicity: One of the reasons why streaming services such as Spotify are so popular is because of their simplicity and easy to use, user friendly interface. We modeled our design decisions after this. We did not do any fancy front end implementations that might distract the user. We worked hard to make a very simple and easy to navigate frontend system, so that we could focus our time and efforts on making complex decisions in the backend.

Data Centralization: We found it very important to showcase all of our features on one page, the search page. That way a user does not have to spend time figuring out how to use our site. They are brought to a very simple and very useful home page which has a search bar right at the top. When they search for a keyword, they are brought to a page that has every feature that a user would want implemented right there. The user can view all data in one centralized location. This allows them to explore different genres, tracks, artists, albums, playlists, and users.

User-Based Queries: We tried very hard to implement useful queries on the home page that will display unique results for each user. As we are both very interested in big data analyzation, we tried to use the small data sets that we had in our database to implement queries that would prove very useful for users if this were a very large operation application. We wanted to display the tracks that they have been interested in and playing recently, as well as show the users that they

follow directly on the front page. With these features, it makes the user experience a lot more enjoyable, because the data seems catered to the individual user.

Backend Functionalities:

SQL QUERIES:

Below are the SQL queries which we have used in our system.

INSERT INTO Follows (username, followee, timestamp) VALUES (?, ?, CURRENT_TIMESTAMP) : This query is used to insert into the follows table. The ‘?’ marks are replaced by appropriate values by the PHP prepared statements.

SELECT distinct trackID, ttitle from Plays natural join Track WHERE username = '{ \$loggedInUser }'

This query pulls out the recent tracks that the user have played.

SELECT trackID, ttitle, count(*) from Plays natural join Track group by trackID,ttitle order by count(*) desc limit 3"

This query is used to pull out the top three most played tracks from the system and display them to the user.

SELECT u.uname from follows f ,user u where (u.username = f.followee) and f.username = ?"

Used to pull out users that the loggedin user follows.

select aid, aname, adesc from Artist where aname like '%{ \$keyword }%' or adesc like '%{ \$keyword }%'

This pulls out the artists that have “keyword in their name of description”

select distinct(trackID), ttitle, tduration, tgenre, aname from Track natural join Artist"

Selects all distinct tracks and their artists from the artist table and the Track table;

select username, uname, ucity from User where username like '%{ \$keyword }%'

Selects users based on like criteria

select albid, albtitle, aname from Album natural join Artist where albtitle like '%{ \$keyword }%' or aname like '%{ \$keyword }%';

Selects album based on the keyword criteria, pulls out albums where either the title or name matches a part of keyword.

```
select pid, ptitle, username from PlaylistInfo" .
```

```
" where ptitle like '%{$keyword}%'";
```

Selects Playlists from the database where their title matches keyword.

```
SELECT pid, ptitle FROM PlaylistInfo WHERE username = '{$name}'"
```

Selects the pid and ptitle from the playlist info table of a particular user.

```
SELECT Track.ttitle, Track.trackID, Track.tduration, Artist.aname from ((Album join TrackList  
on Album.albid = TrackList.albid) join Track on TrackList.trackID = Track.trackID) join Artist  
on Track.aid = Artist.aid where Album.albid = {$id}"
```

This query pull out the tracks, album and artist information from the relevant tables.

```
INSERT INTO PlaylistInfo (pid, ptitle, pdate, username) VALUES (?, ?,  
CURRENT_TIMESTAMP, ?)"
```

Query to create a new playlist.

```
INSERT INTO Playlist (pid, trackID, position) VALUES (?, ?, 1)"
```

Adds a new Track into playlist and positions it 1.

```
INSERT INTO Follows (username, followee, timestamp) VALUES (?, ?,  
CURRENT_TIMESTAMP)"
```

Inserts a new follower into the follows table.

```
INSERT INTO Plays (username, trackID, time, fromAlbum, fromList) VALUES (?, ?,  
CURRENT_TIMESTAMP, NULL, NULL)"
```

Inserts a Play into the table, records the play being done by the user

```
INSERT INTO Playlist (pid, trackID, position) VALUES (?, ?, ?)"
```

Inserts a new track into a playlist at the given position. (automatically filled in by php using prepare statements)

```
INSERT INTO Ratings (username, trackID, rating) VALUES (?, ?, ?)"
```

Inserts a new rating, when the user wants to rate a song.

```
INSERT INTO `User` VALUES ('$username','$name','$email','$city','$pass',NOW())"
```

Inserts a new user into the database, i.e during sign up

```
SELECT ttitle, trackID, tduration, tgenre, aname from Track natural join Artist where aid = " .  
$id . ""
```


Selects tracks and artists from the database;

```
select albid, albtitle, aname from Album natural join Artist where aid = " . $id . "';
```

Pulls out albums of a particular artist.

Changes from Project1:

Between project1 and project2, we did not change a whole lot. Our schema did not have any recommended changes from the TAs, and the current design allowed us to fulfill all of the queries we wanted to execute.

Testing Features:

A SQL script is pasted below that will create the exact same schema and sample data that we used. This will allow simple testing of our database and front end application.

STEPS TO TEST:

- Begin at index.php and login with one of the users already created or create a new user
- Home page will be empty if new user, but current users already have data to fill their homepage
- Use the search bar on the welcome.php page. We included some recommend searches below
 - Keyword: "a" → this will display almost all of the information from our database because "a" is found in a lot of titles, descriptions, etc.
 - Keyword: "eminem" → our database is populated with enough information on Eminem to see our queries at work
 - Keyword: "" → this will not work and prompt you to enter again
- Browse the results in the search page. Test user functionalities as described below
 - Play: Click on the play track → receive confirmation and compare database entries before and after.
 - Rating: Choose any rating from the drop down menu and see database changes when submitting.
 - Playlist: Add a song to a playlist. If you create a new playlist and refresh the page, that playlist will show up when submitting the form again
 - Follows: follow another user and then go back to the welcome page to see a new user included in the "users I follow" feed
- Navigate and explore different albums and playlists and artist information within these search results

SQL SCRIPT:

```

DROP TABLE IF EXISTS `Album`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Album` (
  `albid` int(11) NOT NULL,
  `albtitle` varchar(45) NOT NULL,
  `release_date` datetime NOT NULL,
  `aid` int(11) NOT NULL,
  PRIMARY KEY (`albid`),
  KEY `aid` (`aid`),
  CONSTRAINT `album_ibfk_1` FOREIGN KEY (`aid`) REFERENCES `Artist` (`aid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `Album` WRITE;
/*!40000 ALTER TABLE `Album` DISABLE KEYS */;
INSERT INTO `Album` VALUES (0,'Never Back Down','2009-01-03 11:00:00',1),(1,'Recovery','2010-01-03
11:00:00',1),(2,'The Slim','2011-01-03 11:00:00',3),(3,'The Slim','2011-01-03 11:00:00',3),(4,'Infinite','2011-10-12
11:00:00',4),(5,'Infinite Part 2','2011-10-12 11:00:00',4),(6,'Get The Guns','2011-11-12 11:00:00',4),(7,'8 Mile','2013-12-12
11:00:00',5);
/*!40000 ALTER TABLE `Album` ENABLE KEYS */;
UNLOCK TABLES;

```

```

DROP TABLE IF EXISTS `Artist`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Artist` (
  `aid` int(11) NOT NULL,
  `aname` varchar(45) DEFAULT NULL,
  `adesc` varchar(45) DEFAULT NULL,
  PRIMARY KEY (`aid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `Artist` WRITE;
/*!40000 ALTER TABLE `Artist` DISABLE KEYS */;
INSERT INTO `Artist` VALUES (0,'Megha Sisaudia','Jazz Master'),(1,'Rohit Sisaudia','I am a newbie'),(2,'Amit Tandon','Always
wanted to be a rockstar :p'),(3,'Eminem','I am GOD'),(4,'BackSteet Boys','Dead after the 90s'),(5,'Lehman Brothers','There is no
coming back');
/*!40000 ALTER TABLE `Artist` ENABLE KEYS */;
UNLOCK TABLES;

```

```

DROP TABLE IF EXISTS `Fans`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Fans` (
  `username` varchar(45) NOT NULL,
  `aid` int(11) NOT NULL,
  `timestamp` datetime NOT NULL,
  PRIMARY KEY (`username`,`aid`),

```

```

KEY `aid` (`aid`),
CONSTRAINT `fans_ibfk_1` FOREIGN KEY (`username`) REFERENCES `User` (`username`),
CONSTRAINT `fans_ibfk_2` FOREIGN KEY (`aid`) REFERENCES `Artist` (`aid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `Fans` WRITE;
/*!40000 ALTER TABLE `Fans` DISABLE KEYS */;
INSERT INTO `Fans` VALUES ('adam.lambert',0,'2016-01-03 11:20:00'),('aman.madaan',1,'2016-01-03
11:20:00'),('divya.sharma',2,'2016-01-03 11:20:00'),('raks.sareen',2,'2016-01-03 11:20:00'),('raks.sareen',3,'2016-01-03
11:20:00'),('raks.sareen',4,'2016-01-03 11:20:00'),('sameer.tuli',2,'2016-01-03 11:20:00'),('sameer.tuli',4,'2016-01-03 11:20:00');
/*!40000 ALTER TABLE `Fans` ENABLE KEYS */;
UNLOCK TABLES;

DROP TABLE IF EXISTS `Follows`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Follows` (
  `username` varchar(45) NOT NULL,
  `followee` varchar(45) NOT NULL,
  `timestamp` datetime NOT NULL,
  PRIMARY KEY (`username`,`followee`),
  KEY `followee` (`followee`),
  CONSTRAINT `follows_ibfk_1` FOREIGN KEY (`username`) REFERENCES `User` (`username`),
  CONSTRAINT `follows_ibfk_2` FOREIGN KEY (`followee`) REFERENCES `User` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `Follows` WRITE;
/*!40000 ALTER TABLE `Follows` DISABLE KEYS */;
INSERT INTO `Follows` VALUES ('aman.madaan','adam.lambert','2016-01-03
11:12:00'),('divya.sharma','adam.lambert','2016-01-03 11:12:00'),('raks.sareen','adam.lambert','2016-01-03
11:12:00'),('raks.sareen','aman.madaan','2016-01-03 11:12:00'),('raks.sareen','divya.sharma','2016-01-03
11:12:00'),('raks.sareen','sameer.tuli','2016-01-03 11:12:00'),('sameer.tuli','adam.lambert','2016-01-03
11:12:00'),('sameer.tuli','aman.madaan','2016-01-03 11:12:00');
/*!40000 ALTER TABLE `Follows` ENABLE KEYS */;
UNLOCK TABLES;

DROP TABLE IF EXISTS `Playlist`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Playlist` (
  `pid` int(11) NOT NULL,
  `trackID` int(11) NOT NULL,
  `position` int(11) NOT NULL,
  PRIMARY KEY (`pid`,`trackID`),
  KEY `trackID` (`trackID`),
  CONSTRAINT `playlist_ibfk_1` FOREIGN KEY (`pid`) REFERENCES `PlaylistInfo` (`pid`),
  CONSTRAINT `playlist_ibfk_2` FOREIGN KEY (`trackID`) REFERENCES `Track` (`trackID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```

```

/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `Playlist` WRITE;
/*!40000 ALTER TABLE `Playlist` DISABLE KEYS */;
INSERT INTO `Playlist` VALUES
(0,1,1),(0,2,2),(0,3,3),(1,4,1),(1,5,2),(1,6,3),(2,7,1),(2,8,2),(2,9,3),(3,10,1),(3,11,2),(4,2,1),(4,4,2),(4,6,3),(4,8,4),(5,5,1),(5,9,2),(5,
10,3),(5,11,4),(6,3,1),(6,8,2),(7,9,1),(7,11,2),(8,2,1),(8,8,2),(9,4,1),(9,7,2),(9,10,3),(10,6,1),(10,8,2),(10,11,3),(11,3,1),(11,5,2),(11
,9,3);
/*!40000 ALTER TABLE `Playlist` ENABLE KEYS */;
UNLOCK TABLES;

DROP TABLE IF EXISTS `PlaylistInfo`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `PlaylistInfo` (
  `pid` int(11) NOT NULL,
  `ptitle` varchar(45) NOT NULL,
  `pdate` datetime NOT NULL,
  `username` varchar(45) NOT NULL,
  PRIMARY KEY (`pid`),
  KEY `username` (`username`),
  CONSTRAINT `playlistinfo_ibfk_1` FOREIGN KEY (`username`) REFERENCES `User` (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `PlaylistInfo` WRITE;
/*!40000 ALTER TABLE `PlaylistInfo` DISABLE KEYS */;
INSERT INTO `PlaylistInfo` VALUES (0,'Office','2009-01-03 11:00:00','adam.lambert'),(1,'Workout','2009-01-03
12:00:00','adam.lambert'),(2,'Driving','2009-11-03 12:00:00','adam.lambert'),(3,'Driving','2009-11-13
12:00:00','aman.madaan'),(4,'Studying','2009-11-13 12:00:00','aman.madaan'),(5,'Inspirational','2009-10-13
14:00:00','divya.sharma'),(6,'Study','2009-10-13 14:00:00','divya.sharma'),(7,'Travel','2010-10-13
14:00:00','divya.sharma'),(8,'Office','2010-09-13 14:00:00','divya.sharma'),(9,'Untitled','2010-09-13
14:00:00','raks.sareen'),(10,'Workout','2010-09-13 14:00:00','raks.sareen'),(11,'Car','2010-09-11 14:00:00','sameer.tuli');
/*!40000 ALTER TABLE `PlaylistInfo` ENABLE KEYS */;
UNLOCK TABLES;

DROP TABLE IF EXISTS `Plays`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Plays` (
  `username` varchar(45) NOT NULL,
  `trackID` int(11) NOT NULL,
  `time` datetime NOT NULL,
  `fromAlbum` int(11) DEFAULT NULL,
  `fromList` int(11) DEFAULT NULL,
  PRIMARY KEY (`username`,`trackID`,`time`),
  KEY `trackID` (`trackID`),
  KEY `fromAlbum` (`fromAlbum`),
  KEY `fromList` (`fromList`),
  CONSTRAINT `plays_ibfk_1` FOREIGN KEY (`username`) REFERENCES `User` (`username`),

```

```

CONSTRAINT `plays_ibfk_2` FOREIGN KEY (`trackID`) REFERENCES `Track` (`trackID`),
CONSTRAINT `plays_ibfk_3` FOREIGN KEY (`fromAlbum`) REFERENCES `Album` (`albid`),
CONSTRAINT `plays_ibfk_4` FOREIGN KEY (`fromList`) REFERENCES `playlistinfo` (`pid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `Plays` WRITE;
/*!40000 ALTER TABLE `Plays` DISABLE KEYS */;
INSERT INTO `Plays` VALUES ('adam.lambert',0,'2016-01-03 11:12:00',NULL,NULL),('aman.madaan',0,'2016-01-03
11:12:00',NULL,NULL),('divya.sharma',0,'2016-01-03 11:12:00',NULL,NULL),('divya.sharma',9,'2017-04-03
11:12:00',0,NULL),('divya.sharma',9,'2017-04-03 11:12:01',NULL,0),('raks.sareen',0,'2016-01-03
11:12:00',NULL,NULL),('raks.sareen',8,'2017-04-03 11:12:01',NULL,NULL),('raks.sareen',9,'2017-04-03
11:12:01',NULL,0),('sameer.tuli',0,'2016-01-03 11:12:00',NULL,NULL),('sameer.tuli',1,'2017-01-03
11:12:00',NULL,NULL),('sameer.tuli',10,'2017-01-03 11:12:00',NULL,NULL),('sameer.tuli',11,'2017-01-03
11:12:00',NULL,NULL);
/*!40000 ALTER TABLE `Plays` ENABLE KEYS */;
UNLOCK TABLES;

```

```

DROP TABLE IF EXISTS `Ratings`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Ratings` (
  `username` varchar(45) NOT NULL,
  `trackID` int(11) NOT NULL,
  `rating` int(11) NOT NULL,
  PRIMARY KEY (`username`,`trackID`),
  KEY `trackID` (`trackID`),
  CONSTRAINT `ratings_ibfk_1` FOREIGN KEY (`username`) REFERENCES `User` (`username`),
  CONSTRAINT `ratings_ibfk_2` FOREIGN KEY (`trackID`) REFERENCES `Track` (`trackID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

```

```

LOCK TABLES `Ratings` WRITE;
/*!40000 ALTER TABLE `Ratings` DISABLE KEYS */;
INSERT INTO `Ratings` VALUES
('adam.lambert',0,3),('adam.lambert',1,4),('adam.lambert',9,4),('adam.lambert',11,4),('divya.sharma',7,2),('divya.sharma',8,2),('div
ya.sharma',10,2),('divya.sharma',11,1),('raks.sareen',7,4),('raks.sareen',8,3),('raks.sareen',10,4),('raks.sareen',11,4),('sameer.tuli',7,
2),('sameer.tuli',8,2),('sameer.tuli',10,2),('sameer.tuli',11,1);
/*!40000 ALTER TABLE `Ratings` ENABLE KEYS */;
UNLOCK TABLES;

```

```

DROP TABLE IF EXISTS `Track`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `Track` (
  `trackID` int(11) NOT NULL,
  `title` varchar(45) NOT NULL,
  `tduration` int(11) NOT NULL,
  `tgenre` varchar(10) DEFAULT NULL,
  `aid` int(11) NOT NULL,

```

```

PRIMARY KEY (`trackID`),
KEY `aid` (`aid`),
CONSTRAINT `track_ibfk_1` FOREIGN KEY (`aid`) REFERENCES `Artist` (`aid`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `Track` WRITE;
/*!40000 ALTER TABLE `Track` DISABLE KEYS */;
INSERT INTO `Track` VALUES (0,'Lose Yourself',245,'Rock',3),(1,'Stan',266,'Rock',3),(2,'Without Me',256,'Soft
Rock',3),(3,'Ziddi Hun',256,'Soft Rock',1),(4,'Tu Hai',246,'Soft Rock',1),(5,'Naraaz Mujhse',246,'Jazz',2),(6,'Naraaz
Tujhse',246,'Jazz',2),(7,'Tere Bin Jee Lenge',256,'Rock',2),(8,'Mohabatein lutaunga',256,'Rock',4),(9,'Tere Bin Jee Lenge
Remix',276,'Rock',4),(10,'Tere Naina',266,'Country',4),(11,'Abhimaan',246,'Country',5);
/*!40000 ALTER TABLE `Track` ENABLE KEYS */;
UNLOCK TABLES;

DROP TABLE IF EXISTS `TrackList`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `TrackList` (
  `albid` int(11) NOT NULL,
  `trackID` int(11) NOT NULL,
  `position` int(11) NOT NULL,
  PRIMARY KEY (`albid`,`trackID`),
  KEY `trackID` (`trackID`),
  CONSTRAINT `tracklist_ibfk_1` FOREIGN KEY (`albid`) REFERENCES `Album` (`albid`),
  CONSTRAINT `tracklist_ibfk_2` FOREIGN KEY (`trackID`) REFERENCES `Track` (`trackID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
/*!40101 SET character_set_client = @saved_cs_client */;

LOCK TABLES `TrackList` WRITE;
/*!40000 ALTER TABLE `TrackList` DISABLE KEYS */;
INSERT INTO `TrackList` VALUES
(0,5,1),(0,6,2),(0,7,3),(1,1,1),(1,2,2),(1,11,3),(2,0,1),(2,3,2),(2,11,3),(3,4,1),(3,8,2),(3,9,3),(4,5,1),(4,10,2),(4,11,3),(5,4,1),(5,6,2),(
5,9,3),(6,2,1),(6,6,2),(6,10,3),(7,0,1),(7,1,2);
/*!40000 ALTER TABLE `TrackList` ENABLE KEYS */;
UNLOCK TABLES;

DROP TABLE IF EXISTS `User`;
/*!40101 SET @saved_cs_client      = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `User` (
  `username` varchar(45) NOT NULL,
  `uname` varchar(45) DEFAULT NULL,
  `uemail` varchar(45) DEFAULT NULL,
  `ucity` varchar(45) DEFAULT NULL,
  `upassword` varchar(45) NOT NULL,
  `joiningdate` datetime NOT NULL,
  PRIMARY KEY (`username`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

```