Music Review and Ranking Database for Tracking Industry Trends

Julian Dorsey and Meena Reddy

601.315

A database system designed for querying and inserting music reviews and chart ranking data.

jdorse28@jhu.edu
mreddy10@jhu.edu

Project Link: http://68.183.126.196/

Target Domain

Every year, the release of new music yields a large amount of statistics, articles, and chart rankings analyzing the quality and sales figures of new tracks. However, despite the abundance of data available online reviewing and quantifying these releases, there is no easy way to examine trends in how music is reviewed and / or consumed over time. Our music review and ranking database was created to address this gap. Our database design combined thousands of album reviews from Pitchfork, a leading music journalism site, and yearly sales figures from the Billboard Year-End Hot 100, a list of the 100 best-selling tracks for each calendar year. By creating such a database, we hope to gain insight into some of the notable trends in music consumption and criticism from the 1960s to the present day, and observe the relationship, if any, between critical acclaim and chart performance.

The primary area of specialization in our project was data extraction. The two datasets we obtained (Pitchfork and Billboard data) did not share any primary keys on which we could join the two relations - thus, we had to use an API to obtain the necessary information to link the two tables. Following this, we had to clean the data, generate primary keys when necessary, and format it according to our proposed schema. We then generated customizable stored procedures to allow viewers to get an overview of chart and review trends. We also implemented functionality which would allow users to insert their own reviews into the database, similar to that of other review aggregate websites like Rotten Tomatoes or Metacritic. Finally, we created a view for artists to examine their own commercial and critical performance and trends without viewing potentially sensitive or irrelevant content, such as the usernames of reviewers.

Modifications from Phase I.

Initially, we planned to specialize in data mining and analysis. However, the project ended up naturally requiring a lot more data extraction work than originally planned, so we made this our main area of focus (and thus, did not implement some of our Phase I queries that focused on text and sentiment analysis). Our original design also did not plan on users being able to insert or delete accounts and reviews. Our new schema, which does support this functionality, now includes tables to keep track of user names, passwords, and IDs.

Data Extraction and Loading

Our initial data for this project was obtained from two sources:

1. A dataset of 18, 393 Pitchfork album reviews from 1999-2017 (source)

2. A dataset of song information and lyrics for every song on the Billboard Year-End Hot 100, from 1964 to 2015 (source).

While these provided us with a great deal of information, we were not able to easily compare the two datasets; the Billboard dataset of songs did not include album titles, so we could not connect a song's ranking to its source album's rating on Pitchfork.

To address this gap, we wrote a Python script to request track information from Last.fm's API. We automated requests to get track metadata using the track.getInfo service, which returned JSON with information about the album and artist. Our script then parsed that JSON and appended the relevant information to an additional column of the Billboard dataset before exporting it as a .CSV file. The Pitchfork data was converted from .sqlite to a .sql file, and both files were uploaded through mySQL WorkBench. The relevant Python code is available in the **extract.py**


```
{
  - track: {
        name: "Believe",
        mbid: "32ca187e-ee25-4f18-b7d0-3b6713f24635",
        url: "https://www.last.fm/music/Cher/_/Believe",
        duration: "240000",
      - streamable: {
            #text: "0",
            fulltrack: "0"
        },
        listeners: "453685",
        playcount: "2158133",
      - artist: {
            name: "Cher",
            mbid: "bfcc6d75-a6a5-4bc6-8282-47aec8531818",
            url: "https://www.last.fm/music/Cher"
        },
      - album: {
            artist: "Cher",
            title: "Believe",
            mbid: "63b3a8ca-26f2-4e2b-b867-647a6ec2bebd",
            url: "https://www.last.fm/music/Cher/Believe",
          - image: [
              - {
                    #text: "https://lastfm-img2.akamaized.net/i/u/34s/3b54885952161aaea4ce2965b2db1638.png",
                    size: "small"
                },
              - {
                    #text: "https://lastfm-img2.akamaized.net/i/u/64s/3b54885952161aaea4ce2965b2db1638.png",
                    size: "medium"
                },
              - {
                    #text: "https://lastfm-img2.akamaized.net/i/u/174s/3b54885952161aaea4ce2965b2db1638.png",
                    size: "large"
                },
              - {
                    #text: "https://lastfm-img2.akamaized.net/i/u/300x300/3b54885952161aaea4ce2965b2db1638.png",
                    size: "extralarge"
                }
            ],
          - @attr: {
                position: "1"
            }
```


After the datasets were uploaded, we faced more issues with cleaning and matching the data.

Several of the relations had duplicate tuples, which we removed. We also had to auto-generate primary keys and create new relations to fit our proposed schema (specified later in DDL).

Software

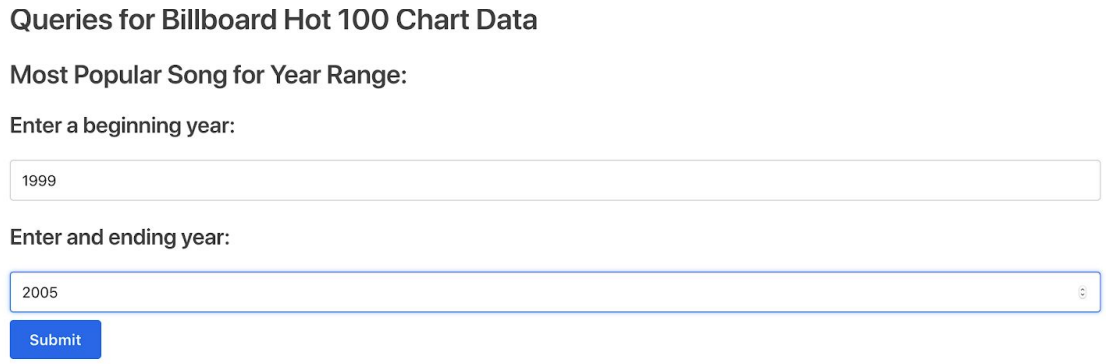
Our mySQL database was hosted by Microsoft Azure, and the web server was hosted on a DigitalOcean droplet.

User's Guide

The user interface for this project is a website, accessible here: http://68.183.126.196/

Once on the website, a viewer has three options:

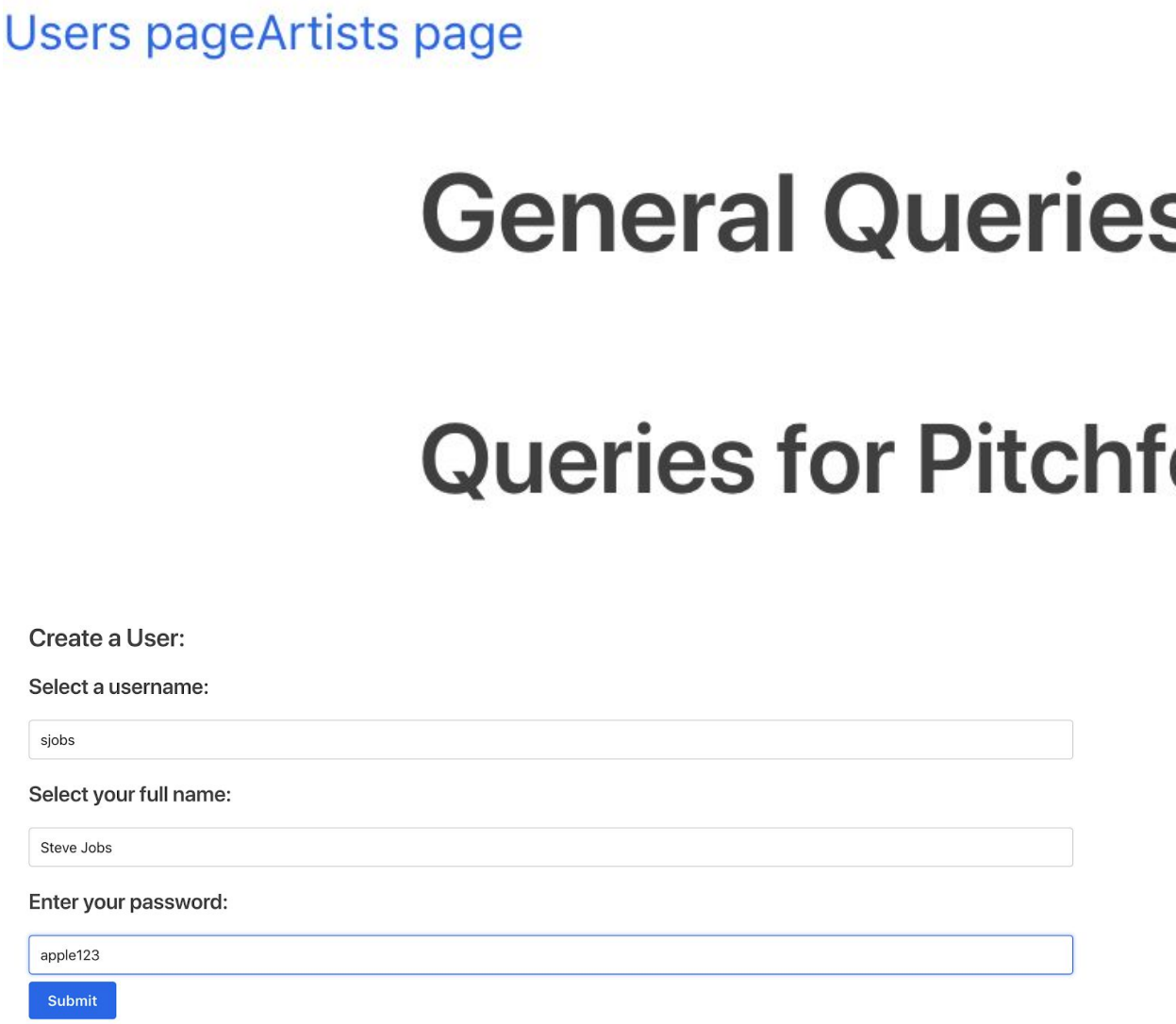
1) View basic summary statistics (accessible to all visitors)
    a) To perform general queries, simply enter the desired information into a specific form (under the appropriate heading), and click "SUBMIT" (Score values range from 0 - 10)



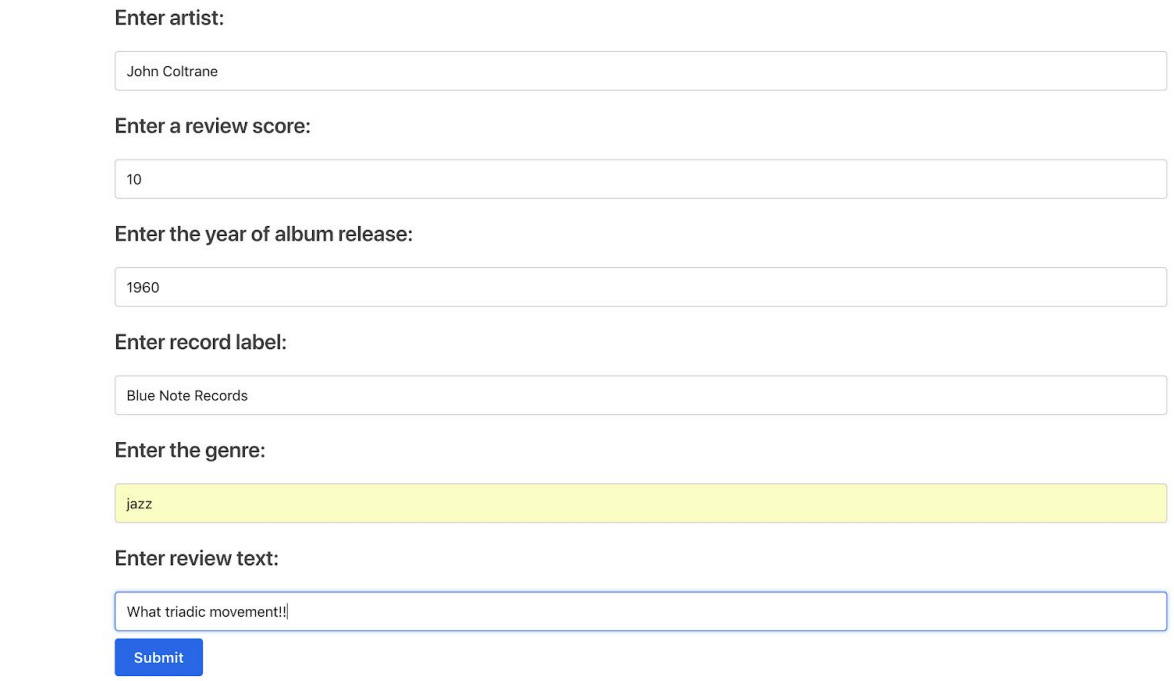
Song	Year	Average Rank
what its like	1999	28.0000
he cant love u	2000	62.0000
survivor	2001	23.0000
without me	2002	21.0000
shake ya tailfeather	2003	13.0000
lean back	2004	10.0000
disco inferno	2005	11.0000

2) Enter a viewer page and signup as a user

a) Go to the users page. Then follow the login form to sign up as a user. Once created, go back to the user page and leave a review by filling out the form and hitting submit. Similar forms for review deletion and account deletion exist. Upon account deletion, all reviews for that user are removed from the database:



User Created



Review Created

3) Enter an artist page and link a user account with a popular artist already in the database

a) Go the artists page. From there, register your account with a popular artist likely to exist in the Billboard and Pitchfork databases. After doing so, generate an artist report which consists of the average ranking of an artists song on the Billboard HOt 100 charts and the artists average Pitchfork album review score:

Users pageArtists page

General

Create a User:

Select a username:

Drake

Enter your full name:

Drake

Select a password:

drake

Submit

Generage Artist Summary:

Enter your username:

Drake

Enter your password:

drake

Submit

Artist	Average Score	Average Rank
drake	8.016666666666666	57.1250

Area of Specialization

As mentioned earlier, our area of specialization as .315 students was data extraction. We not only sourced our data from real-life sources (Billboard and Pitchfork), we also implemented a Python script that would call on a third-party API (from Last.fm, a popular music streaming site) to help build out our database design.

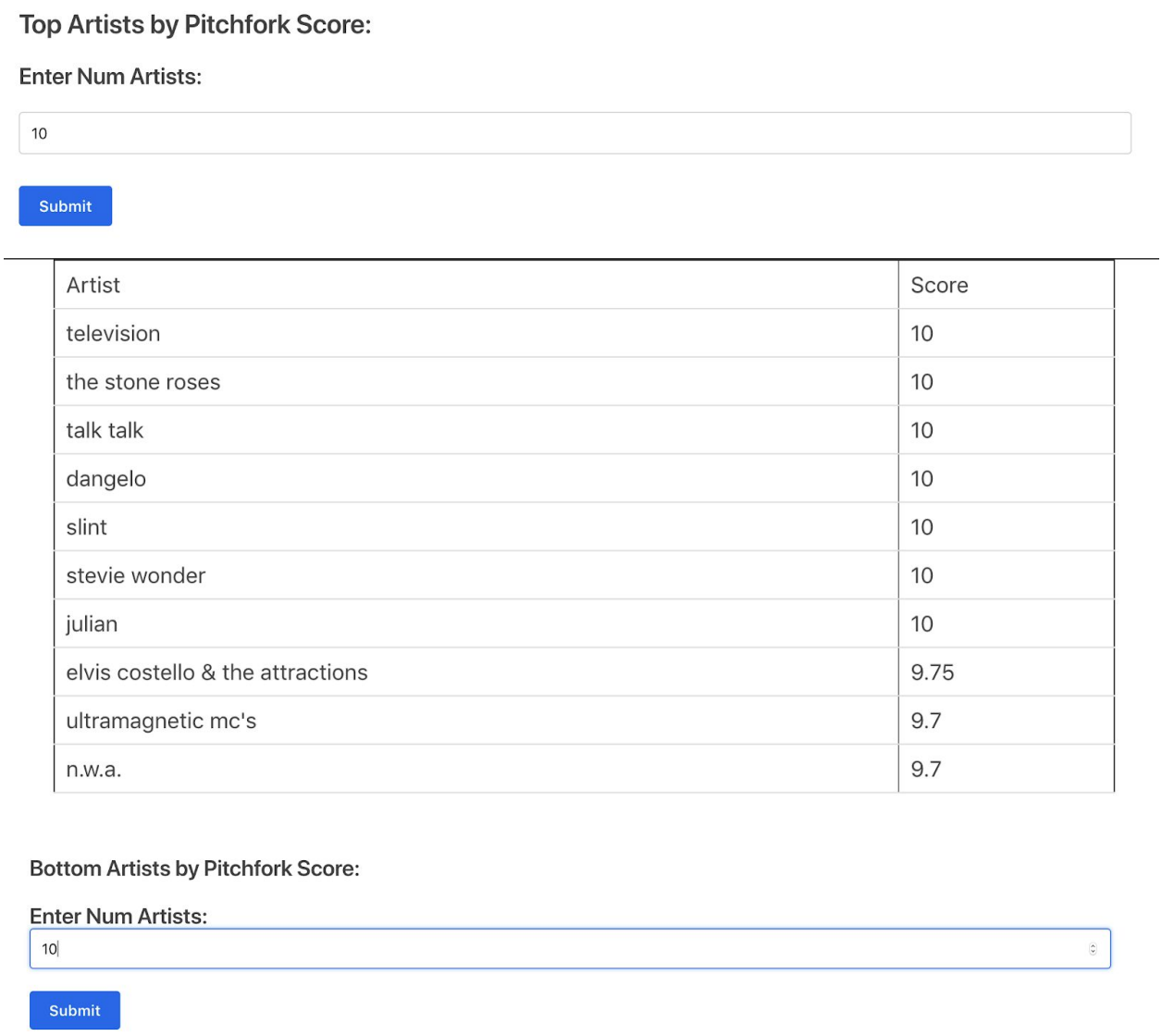
Project Strengths

- Solves a real-world problem → there is currently no good way to look at long-term trends between music reviews and sales over time
- Uses real data scraped from well-known music sources
- Project gave us valuable first-hand experience working with the inconsistency, messiness, and many formats of real data
- Expands on database concept by implementing functionality to create and add users and user reviews that are distinguishable from critic reviews -- combines the best features of review websites like Rotten Tomatoes with historical background and context for the reviews
- Users are able to input custom parameters for most queries, allowing them to engage with what data is personally interesting or meaningful to them
- Artists can view queries specifically about them without having to sort through unrelated data

Project Limitations/Possibilities for Improvement

- Data was often inconsistent between Pitchfork and Billboard - e.g. the same song would have different spellings, different album title, etc. Songs that were released as singles were sometimes mislabeled as not belonging to a particular album.
    - Given more time, would implement a script to compare data scraped from a third-party source (e.g. Google APIs or Wikipedia) to reduce inconsistencies
- Data spanned different time frames - could not compare any Pitchfork data before 1999
- Given more time, would be interesting to apply more data analysis and/or sentiment analysis to it
    - Could use Python to do a textual analysis of review/lyric content, and not just review scores
    - Could incorporate more graphics and visualizations into UI
- Could implement “admin” portal for full functionality over inserting songs, values, etc.

Sample Output



Top Artists by Pitchfork Score:

Enter Num Artists:

10

Submit

Artist	Score
television	10
the stone roses	10
talk talk	10
dangelo	10
slint	10
stevie wonder	10
julian	10
elvis costello & the attractions	9.75
ultramagnetic mc's	9.7
n.w.a.	9.7

Bottom Artists by Pitchfork Score:

Enter Num Artists:

10

Submit

Artist	Score
travis morrison	0
push kings	0.1
shat	0.2
dan le sac vs. scroobius pip	0.2
liars academy	0.6
toe	0.8
geoff farina	1
non	1
avey tare, kra brekkan	1
lou reed, metallica	1

Average Pitchfork Genre Scores:

Enter the genre:

electronic

Enter a beginning year:

2004

Enter and ending year:

2015

Submit

Year	Average Score
2011	7.049999999999999
2012	6.895600000000004
2013	7.007894736842107
2014	6.980526315789475
2015	7.166990291262136

Reviewer Average Ratings:

Submit

Author	Average Score
samir khan	6.6499999999999995
maura johnston	10
nelson george	10
carvell wallace	9.833333333333334
julian	10
dorian lynskey	9.5
rollie pemberton & nick sylvester	9.4
dave stelfox	7.825
andrew nosnitsky	9.1
nathan rooney	6.915384615384616
sarah zupko	7.599999999999999

Best New Music Averages:

Submit

Average Best New Music Score	Average Normal Score
8.67645811240721	6.91540754327638

Who Decides Best New Music?:

Submit

Author Type	Percentage
managing editor	33.3333
associate reviews editor	26.8293
news director	20.0000
executive editor	18.5263
senior editor	14.6091
assistant editor	12.5000
associate features editor	9.0909
editor-in-chief	8.2353
contributing editor	7.1429
senior staff writer	5.9226
tracks coordinator	5.7692
contributor	5.2899
deputy news editor	3.0000
associate staff writer	2.8571

Number of Charting Songs by Subject:

Enter the keyword:

love

Enter a beginning year:

1970

Enter and ending year:

2001

Submit

Year	Number of Songs
1970	97
1971	87
1972	95
1973	98
1974	98
1975	94
1976	97
1977	95
1978	96
1979	96
1980	93
1981	92
1982	95
1983	100
1984	98
1985	100

Queries for Combined Pitchfork and Billboard Hot 100 Data

Which songs ranking on some Billboard threshold appear on albums above some score threshold?:

Enter Rank Threshold (lower = better):

50

Enter Pitchfork Threshold (higher = better):

5

Song	Title	Artist	Score	Rank
the hills	beauty behind the madness	the weeknd	7.2	10
want to want me	everything is 4	jason derulo	6.6	17
im not the only one	in the lonely hour	sam smith	5.5	26
hotline bling	views	drake	6.8	30
hello	25	adele	7.3	35
talking body	queen of the clouds	tove lo	7.2	37
stay with me	in the lonely hour	sam smith	5.5	10
team	pure heroine	lorde	7.3	18
royals	pure heroine	lorde	7.3	20
habits stay high	queen of the clouds	tove lo	7.2	32
mirrors	the 20/20 experience	justin timberlake	8.4	6
royals	pure heroine	lorde	7.3	15

Which genre has the highest ranking songs on Billboard?:

Submit

Genre	Average Ranking
pop/r&b	42.6735
rap	44.3000
electronic	49.4286
rock	52.3654
metal	56.0000

Which record label produces tracks with the highest average rankings on Billboard?:

Submit

Record Label	Average Ranking
downtown	7.0000
xo	10.0000
roc nation	12.0000
dreamworks	14.0000
beluga heights	17.0000
lava	17.6667
rolling stones	18.0000
self-released	22.0000
republic	27.0000
motown	34.5000

What is the genre makeup of music produced by each record label?:

Submit

Label	Genre	% of Output
a&m	pop/r&b	100.0000
aftermath	pop/r&b	36.3636
aftermath	rap	63.6364
arista	rap	100.0000
asylum	rap	100.0000
atlantic	pop/r&b	16.6667
atlantic	rap	83.3333
beluga heights	pop/r&b	100.0000
big brother	rock	100.0000
capitol	electronic	10.0000
capitol	pop/r&b	40.0000
capitol	rock	50.0000
cash money	rap	100.0000

What is the average ranking of songs on Billboard whose albums receive Best New Music on Pitchfork?:

Submit

Song	Average Ranking
a milli	43.0000
adorn	90.5000
brown sugar	18.0000
epic	75.0000
go your own way	94.0000
headlines	85.0000
jesus walks	43.0000
mirrors	6.0000
off the wall	79.0000
rock with you	4.0000
started from the bottom	32.0000
stronger	27.0000
through the wire	61.0000
wonderwall	56.0000

DDL Specification

```
CREATE TABLE album (
  album_name varchar(92),
  artist varchar(59),
  album_id int(11)
  PRIMARY KEY (album_id)
```

```
)

CREATE TABLE album_review (
  album_id int(11) NOT NULL,
  reviewid int(11)
)

CREATE TABLE app_users (
  username varchar(32) NOT NULL,
  fullname varchar(96),
  PRIMARY KEY (username)
)

CREATE TABLE artists (
  reviewid int(11),
  artist text
)

CREATE TABLE content (
  reviewid int(11),
  content text
)

CREATE TABLE content (
  reviewid int(11),
  content text
)

CREATE TABLE `genres` (
  reviewid int(11),
  genre text
)
CREATE TABLE labels (
  reviewid int(11),
  label text
)

CREATE TABLE passwords (
  username varchar(32) NOT NULL,
```

```
  password varchar(32),
  PRIMARY KEY (username),
  FOREIGN KEY (`username`) REFERENCES app_users (username)
)

CREATE TABLE reviews (
  reviewid int(11) NOT NULL AUTO_INCREMENT,
  title text,
  artist text,
  url text,
  score double,
  best_new_music int(11),
  author text,
  author_type text,
  pub_date text,
  pub_weekday int(11),
  pub_day int(11),
  pub_month int(11),
  pub_year int(11),
  PRIMARY KEY (reviewid)
)

CREATE TABLE song (
  song_id int(11) NOT NULL AUTO_INCREMENT',
  rank int(11),
  song varchar(59),
  artist varchar(59),
  year int(11),
  lyrics varchar(5224)
)


CREATE TABLE song_album (
  song_id int(11) NOT NULL,
  album_id int(11) NOT NULL,
)

CREATE TABLE user_review (
  reviewid int(11) NOT NULL,
```

```
  username varchar(32)
  PRIMARY KEY (reviewid),
  FOREIGN KEY (username) REFERENCES app_users (`username`)
)

CREATE TABLE years (
  reviewid int(11) DEFAULT NULL,
  year int(11) DEFAULT NULL
)
```

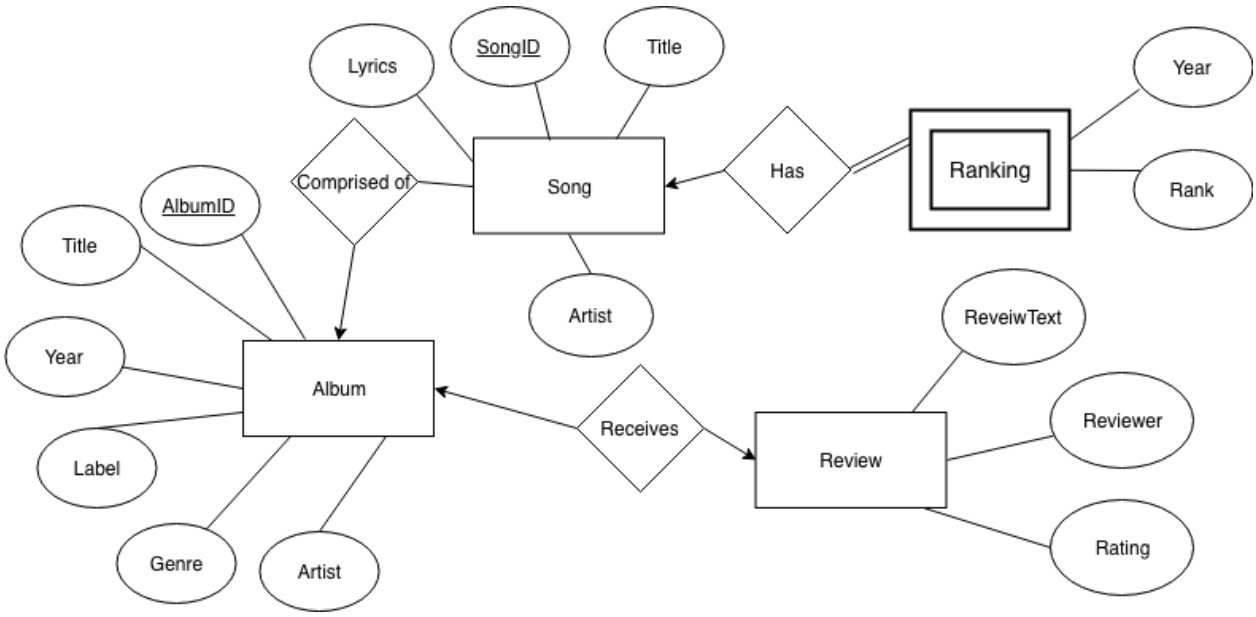
Additional steps:

1. Include all SQL code/all other code
2. Attach Phase I
3. Drop unnedeed tables

Final Project Phase 1
Meena Reddy & Julian Dorsey
601.315

1. Team Members: Julian Dorsey & Meena Reddy
2. Target domain: a database of music reviews/chart rankings (sourced from Billboard and Pitchfork)
3. Questions to Answer:
    a. Billboard:
        i. According to Billboard, who is the most popular artist of each decade?
        ii. Which artist has the highest average song ranking on the Billboard Hot 100?
        iii. For each decade, which artist was featured the most times on the Hot 100?
        iv. What is the most commonly used word amongst song titles/lyrics of Hot 100 songs?
    b. Pitchfork:
        i. Which artist has the highest/lowest average rating on Pitchfork?
        ii. Which genre has the highest/lowest average review rating on Pitchfork for each year?
        iii. What was the change in average rating for pop albums from 1980 to 2015?
        iv. What is the average album rating given by each reviewer?
        v. What is the average rating for “Best New Music” songs/albums compared to music that was not “Best New Music?”’
    c. Combined:
        i. How many songs that charted above 50 on the Billboard rankings received scores above 7 on Pitchfork?
        ii. Which contributor best predicted how an album’s songs would chart? Assume that a score above 7 is a marker of an album’s quality.
        iii. Return all the songs that were awarded ‘Best New Music’ and also charted on Billboard.
    d. NLP/data science:
        i. What is the average ranking for songs about ‘love’ compared to songs not about love?
        ii. Is the Pitchfork review consistent with the numerical score (using some sentiment analysis scheme)?
        iii. Based on a song’s genre and its Pitchfork rating, can we predict how it will chart on Billboard?

4.



SONGS

SongID	**Title**	**Artist**	**Lyrics**
0001	Baby	Justin Bieber	baby, baby, baby

RANKING

SongID	**Year**	**Rank**
0001	2010	07

ALBUMSONGS

SongID	**AlbumID**
0001	5555

ALBUM

AlbumID	**Title**	**Year**	**Artist**	**Genre**	**Label**
5555	My World 2.0	2010	Justin Bieber	Pop	Island

REVIEW

AlbumID	**Rating**	**Reviewer**	**ReviewText**
5555	6.5	John Smith	“This is great”

```sql

/* Query 3.a.ii */
SELECT Artists, AVG(R.rank) AS AVG_RANKING
FROM SONGS AS S
    INNER JOIN RANKING AS R ON S.songid = R.songid
GROUP BY ARTIST;

/* Query 3.b.iii */

SELECT ChangeInRating
FROM (SELECT AVG(rating) AS AvgRating
    FROM Album AS A
        INNER JOIN Review AS R
    WHERE A.albumid = R.albumid
        AND A.genre = 'pop'
        AND A.year = 1980) AS A,
    (SELECT AVG(rating) AS AvgRating
    FROM Album AS A
        INNER JOIN Review AS R
    WHERE A.albumid = R.albumid
        AND A.genre = 'pop'
        AND A.year = 2015) AS B
WHERE ChangeInRating = A - B;

/* Query 3.b.iv */
SELECT reviewer, avgrating
FROM (SELECT reviewer, AVG(rating) AS avgrating
    FROM Review
    GROUP BY reviewer);

/* Query 3.c.i */
SELECT COUNT(*) AS NumSongs
FROM Songs AS S
INNER JOIN Ranking AS R ON S.songid = R.songid
INNER JOIN AlbumSongs AS AlSongs ON s.songid = Alsongs.songid
INNER JOIN Review AS Rv ON Alsongs.albumid = Rv.albumid
WHERE R.rank > 50
AND Rv.rating > 7;

```

5.

6. Loading database with values:
    a. We will obtain Billboard data from: https://www.kaggle.com/rakannimer/billboard-lyrics
    b. We will obtain Pitchfork data from https://www.kaggle.com/nolanbconaway/pitchfork-data/version/1
    c. These are both .csv files, which can be easily imported into mySQL using Workbench or a similar tool.

    d. We will need to find a way to correlate the song (on Billboard) with its album, and are looking into solutions for this.
7. We will have pre-formatted inputs through which the user can submit parameters for various queries, and will display the results as views or tables.
8. We plan to focus on data mining, including basic machine learning and natural language processing and will also touch on data presentation using Python libraries.