Jerald Liu - 600.315

Tokio Nakamoto - 600.315

2) Domain: The domain of our project is a database containing song information and associated meta data, including artist, year written, key, tempo, tags, etc. This data will be used in order to determine similarity between songs so users can discover songs based on their current tastes.

3) Description and Database Design has not changed since Phase I, except that we did not webscrape image searches to get album cover art for each song.

4) How To Load Data into the Database: We pulled raw .h5 data from a website called Million Song Dataset (<http://labrosa.ee.columbia.edu/millionsong/>) and analyzed the data using scripts in order to translate the data into a readable format. From there we inserted the data into our database in the tables shown in the Data Models section. In our python/ directory in our submission, there is code to analyze hdf5 files. Note that there must be installation of certain dependencies in order to run the code.

5) Platform:

MySQL version 5.1.73 on Fedora 20 (Heisenbug)

Python will be used for processing .h5 files containing raw data of song information.

6) Using our application is a short and easy process. Visit the url: <http://ugrad.cs.jhu.edu/~jliu79/dashboard.php>.

Once the page has loaded, a user can discover new music by searching according to different categories (e.g. by artist, similar songs, genre, etc.).

A user clicks one of several buttons, each representing a category. Then, a category specific search bar pops up in which a user can type in the desired input. The application then shows the user the resulting table from a query specified by the user input.

7) UI: We made a simple, elegant, and intuitive interface to make the application stylish and appealing to its users.

We also analyze files of a compressed format (hdf5) that are used for storing large amounts of indexed data. This data is located in a folder /data which was not included in the submission because it was so large (over 2GB)

8) Ideally, we’d like to have previews/song samples for each song. However, licensing issues prevent us from doing so. Also, we found that some attributes of songs after analysis were always a default value, so we could not include some attributes in our output (namely, danceability of tracks).

We also could not figure out how to get rid of some tables in our existing database cs415\_jliu79\_db. Though, that did not interfere with how our output was.

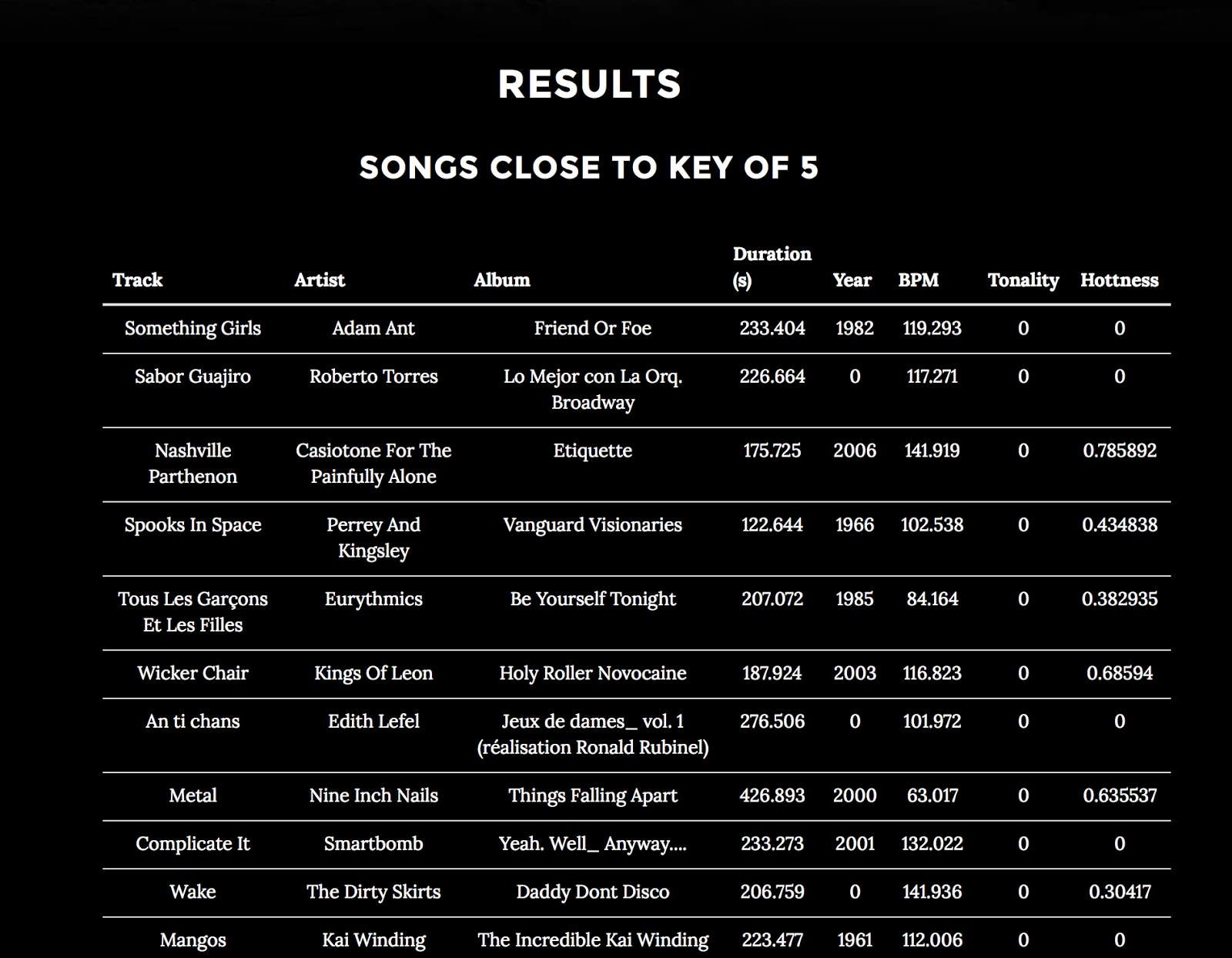
9) Output of project:



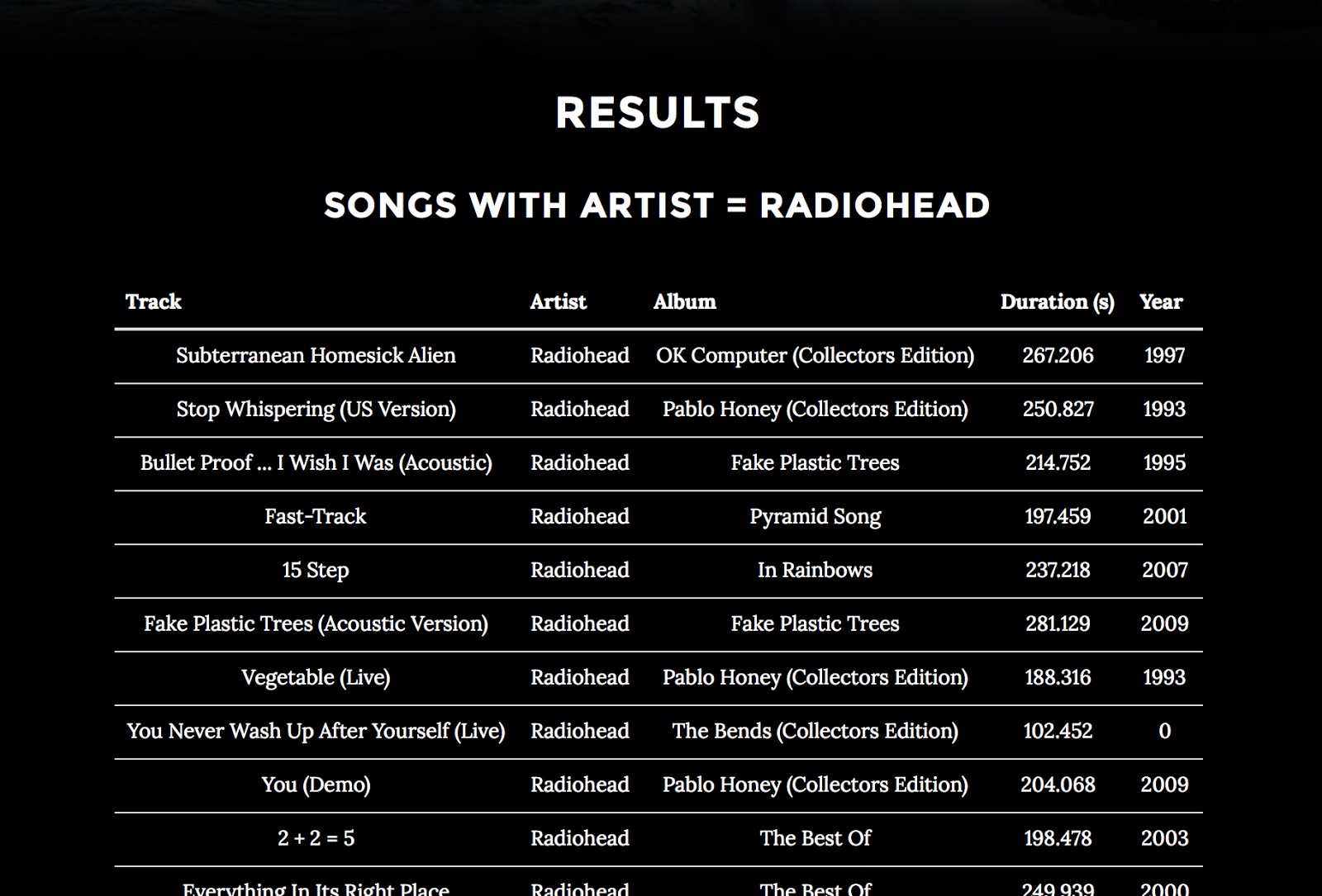
Using Tempo and Key as parameters to search for songs



Using genre to search for songs



Using Key to search for songs



Search by artist

10) Data Model:

CREATE TABLE artist (

artist\_id VARCHAR(32) PRIMARY KEY,

artist\_name VARCHAR(64) NOT NULL,

hottness FLOAT(5) NOT NULL,

familiarity FLOAT(5) NOT NULL

);

CREATE TABLE artist\_genres (

artist\_id VARCHAR(32) NOT NULL,

genre VARCHAR(32) NOT NULL,

FOREIGN KEY(‘artist\_id’) REFERENCES artist (‘artist\_id’)

);

CREATE TABLE track (

track\_id VARCHAR(128) PRIMARY KEY,

title VARCHAR(128) NOT NULL,

album VARCHAR(128) NOT NULL,

artist\_id VARCHAR(32) NOT NULL,

artist\_name VARCHAR(256) NOT NULL,

duration FLOAT(5) NOT NULL,

year INT NOT NULL,

FOREIGN KEY(‘artist\_id’) REFERENCES artist(‘artist\_id’)

);

CREATE TABLE track\_analysis (

track\_id VARCHAR(128) PRIMARY KEY,

tempo FLOAT(2) NOT NULL,

tonality INT NOT NULL,

hottness FLOAT(2) NOT NULL,

FOREIGN KEY(‘track\_id’) REFERENCES track(‘track\_id’)

);

11)

finalproject.sql, for db initialization and table population

#0 List songs by artist

DELIMITER //

CREATE PROCEDURE ArtistSong(IN artistin VARCHAR(256))

BEGIN

SELECT \*

FROM track as t, artist as a

WHERE t.artist\_id = a.artist\_id AND a.artist\_name=artistin;

END //

DELIMITER ;

List of SQL statements:

#1 List all songs from a specific year.

DELIMITER //

CREATE PROCEDURE SongFromYear(IN yearin INT)

BEGIN

SELECT \*

FROM track as t

WHERE t.year = yearin;

END //

DELIMITER ;

#2 songs from a specific decade

DELIMITER //

CREATE PROCEDURE SongFromDecade(IN decade INT)

BEGIN

SELECT \*

FROM track as t

WHERE t.year >= (decade-(decade%10)) AND t.year < (decade-(decade%10)) + 10;

END //

DELIMITER ;

#3 List All Songs in a specific genre

DELIMITER //

CREATE PROCEDURE SongOfGenre(IN genrein VARCHAR(256))

BEGIN

SELECT \*

FROM track as t, artist\_genres as a

WHERE t.artist\_id=a.artist\_id AND a.genre=genrein;

END //

DELIMITER ;

#4 50 most popular songs.

DELIMITER //

CREATE PROCEDURE SongTop50()

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id = a.track\_id

ORDER BY a.hottness DESC LIMIT 50;

END //

DELIMITER ;

**#5 List songs with a specific tempo**

DELIMITER //

CREATE PROCEDURE SpecificTempo(IN tempoin FLOAT(2))

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id=a.track\_id AND a.tempo=tempoin;

END //

DELIMITER ;

#6 songs within a tempo range.

DELIMITER //

CREATE PROCEDURE SongWithinTempoRange(IN min FLOAT(2), IN max FLOAT(2))

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id = a.track\_id AND a.tempo >= min AND a.tempo <= max;

END //

DELIMITER ;

#7 List Songs close to a specific tempo (+/- 3 BPM)

DELIMITER //

CREATE PROCEDURE CloseToTempo(IN tempoin FLOAT(2))

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id=a.track\_id AND (a.tempo<=tempoin+3 AND a.tempo>=tempoin-3);

END //

DELIMITER ;

#8 songs with a specific key.

DELIMITER //

CREATE PROCEDURE SongWithKey(IN keyin INT)

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id = a.track\_id AND a.tonality = keyin;

END //

DELIMITER ;

#9 Song in a close key (+/- 7 Mod 12)

DELIMITER //

CREATE PROCEDURE SongCloseToKey(IN keyin INT)

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id = a.track\_id AND (a.tonality=(keyin - 7)%12 OR a.tonality=(keyin + 7)%12);

END //

DELIMITER ;

**#11 Fast Dance song**

DELIMITER //

CREATE PROCEDURE FastDanceSong()

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id=a.track\_id AND a.tempo > 120 AND a.danceability > .6;

END //

DELIMITER ;

**#12 songs that are danceable from the 80’s and sort by year, then popularity.**

DELIMITER //

CREATE PROCEDURE SongDance80()

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id = a.track\_id AND t.year >= 1980 AND t.year < 1990

AND a.danceability >= 0.5

ORDER BY t.year, a.hottness DESC;

END //

DELIMITER ;

#13 Electronic song between 128 and 140 BPM

DELIMITER //

CREATE PROCEDURE ElectronicDance()

BEGIN

SELECT \*

FROM track as t, track\_analysis as a, artist\_genres as g

WHERE t.track\_id=a.track\_id AND a.tempo<=140 AND a.tempo>128 AND

g.artist\_id=t.artist\_id AND g.genre='electronic';

END //

DELIMITER ;

**#14 all songs that are at least semi-danceable (>= .5 danceability) and group by key and sort by danceability.**

DELIMITER //

CREATE PROCEDURE SongSemiDance()

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id = a.track\_id AND a.danceability > 0.5

GROUP BY a.tonality

ORDER BY a.danceability DESC;

END //

DELIMITER ;

#15 (close in tempo, similar key)

DELIMITER //

CREATE PROCEDURE MashupSong(IN tempoin FLOAT(2), IN keyin INT)

BEGIN

SELECT \*

FROM track as t, track\_analysis as a

WHERE t.track\_id=a.track\_id AND (a.tonality=(keyin - 7)%12 OR a.tonality=(keyin + 7)%12) AND (a.tempo<=tempoin+3 AND a.tempo>=tempoin-3);

END //

DELIMITER ;