**Top Songs Database**

**Project Description:**

One of my great passions is music. For the **E**xtract, **T**ransform & **L**oad Project I wanted to explore the music that has affected the way we live and love, the songs considered the best in the English language.

This project attempts to answer the following questions.

1. What are the best songs based on the ranking criteria?
2. Which artists were most prolific in the top 5000?
3. Did particular decades produce more “great” songs than others?

Luckily for me there is a project that ranked the top 5000 tracks in aggregate based on polls, sales and awards won (TopSongs.csv). So I started with that.

I then come across the Million Song Dataset (music.csv), which derived metadata about one million contemporary songs. This project contains esoteric information such as “tatums” (the shortest time between consecutive musical notes on the track) and fadeout in addition to a song’s duration and the location where the track was released.

Ideally I would have liked to have included the album on which these popular songs were first released. But this data proved difficult to obtain without API pulls. I have included an Excel dataset for Rolling Stone’s “The 500 Greatest Albums” (albumlist.xls), but I couldn’t find a way to include it that made sense.

## The Process:

Initially I performed research to gather the data pertinent to answer the questions posed. Then I imported the data (in csv and xlxs forms) into Jupyter notebook in order to clean and manipulate the datasets. The sheer number of columns involved made it difficult to do data cleaning, manipulation and analysis.

It became clear that much of the data (especially metadata) would not be relevant, so I filtered the original datasets in Pandas, extracting those columns relevant to the planned database. From the metadata, these included the following fields: artist\_familiarity, artist\_location, artist\_name, artist\_terms (genre), duration, loudness, and tempo.

I wanted to discover the artists whose songs permeate the top 5000. So I grouped the tracks by artist, making sure to give the same rank to performers who produced the same number of top tracks. Similarly I grouped the songs by decade to generate a visualization in pie chart form.

**Transform: (Data Cleanup & Analysis)**

Types of transformation needed for the data: Cleaning, Filtering, Aggregation

* Import and convert the .csv files into Pandas Data frames.
* Aggregate DataFrames to pull all values into one .csv file
* Group the data to determine and rank the prolific artists who produced the most “best” songs
* Analyze the tracks by year and visualizing in pie chart form by decade
* Define the classes (tables) and fields for the database, then convert the data frames back to csv form
* Import the new csv files into a SQL relational database using SQL Alchemy and SQLite

## Conclusions:

* No one decade produced the “best’ songs. The 5000 tracks range from the 1940s to the present.
* Here is a list of the top ten songs:

| **Rank** | **Artist** | **Song** |
| --- | --- | --- |
| 1 | Bing Crosby | White Christmas |
| 2 | Bill Haley & his Comets | Rock Around the Clock |
| 3 | Celine Dion | My Heart Will Go On |
| 4 | Sinead O'Connor | Nothing Compares 2 U |
| 5 | The Beatles | Hey Jude |
| 6 | Bryan Adams | (Everything I Do) I Do it For You |
| 7 | Whitney Houston | I Will Always Love You |
| 8 | Pink Floyd | Another Brick in the Wall (part 2) |
| 9 | Irene Cara | Flashdance... What a Feeling |
| 10 | Elton John | Candle in the Wind '97 |

* Here is a list of the artists producing the most songs in the top 5000:

| **Rank** | **Artist # Songs** |
| --- | --- |
| 1 | Elvis Presley | 43 |
| 2 | The Beatles | 42 |
| 3 | Madonna | 41 |
| 4 | The Rolling Stones | 31 |
| 5 | Michael Jackson | 29 |
| 6 | Elton John | 26 |
| 7 | Bing Crosby | 24 |
| 8 | Mariah Carey | 23 |
| 9 | U2 | 23 |
| 10 | Janet Jackson | 21 |

# This repo includes the following:

### Resources Folder

Contains the Jupyter Notebook that contains the code used for the project:

* music.ipynb – Python code that reads the data from csv files, aggregates it into a unitary Pandas data frame, uses Matplotlibt to plot pie chart and pushes three csv files into SQL database using SQLite and SQL Alchemy

### Output Folder

* number\_songs\_by\_decade.png  - a pie chart that breaks down the percentage of songs by decade
* music.sqlite – file generated by engine to import classes(tables) into SQL
* artist\_csv, rank\_csv, song\_csv – csv’s exported to engine with same fields as SQL tables

**Data Sources**

* music.csv - <https://github.com/angrbrd/top5000-playlist/blob/master/TopSongs.csv>

Aggregate ranking of tracks based on polls, radio play, sales and awards.

* artist.csv - <https://think.cs.vt.edu/corgis/csv/music/music.html>

This library comes from the Million Song Dataset, which used a company called the Echo Nest to derive data points about one million popular contemporary songs. The data contains standard information about the songs such as artist name, title, and year released. Additionally, the data contains more advanced information; for example, the musical key of the song,

* albumlist.xls – <https://www.kaggle.com/notgibs/500-greatest-albums-of-all-time-rolling-stone>

The 500 Greatest Albums of All Time" is a list compiled by American magazine “Rolling Stone” .The rankings were based on votes from selected rock musicians, critics, and industry figures. Ultimately I did not use this dataset because the only way to obtain the album associated with a particular song was through API queries.

**Future Work and Improvements**

To further enhance the project I would need the original album on which each of the tracks was released. This information is available on various APIs, the best of which is LastFM. Obtaining this information would allow me to add an album table to the schema.