**ETL Project (aka NOT Pineapple Project)**

**Justin Hobbs**

**Mychal Spivey**

**Katy Webb**

**Intro:**

In the midst of one of our typical, nonsensical table conversations we joked about pulling in all of Kanye West’s discography from Wikipedia and using it to develop a music database that allowed the user to search for albums based on sales, the recording studio, certification status (gold, platinum, etc.), number of tracks, etc. From there we discovered that Wikipedia URLs are inconsistent for artists (some are formatted for just the recording artist, some are formatted for the artist as a producer, some are formatted for the artist as a collaborator, etc.), and that the number of tables and the variations in table formatting for each artist made it very difficult to use Wikipedia as a source. Instead, we used an API and some web scraping to create our album database, which is fine because Justin and I discovered that only Mychal likes Ye’s music.

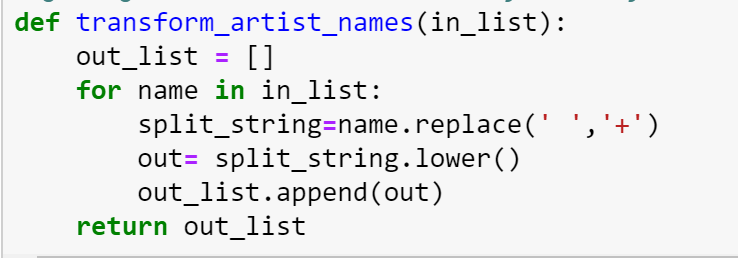
**Extract:**

Our data sources are the iTunes Search API (<https://affiliate.itunes.apple.com/resources/documentation/itunes-store-web-service-search-api/>) which pulls in JSON-formatted data, and the list of Billboard’s Top 100 artists (<https://www.billboard.com/charts/artist-100>) which pulls in plain text.

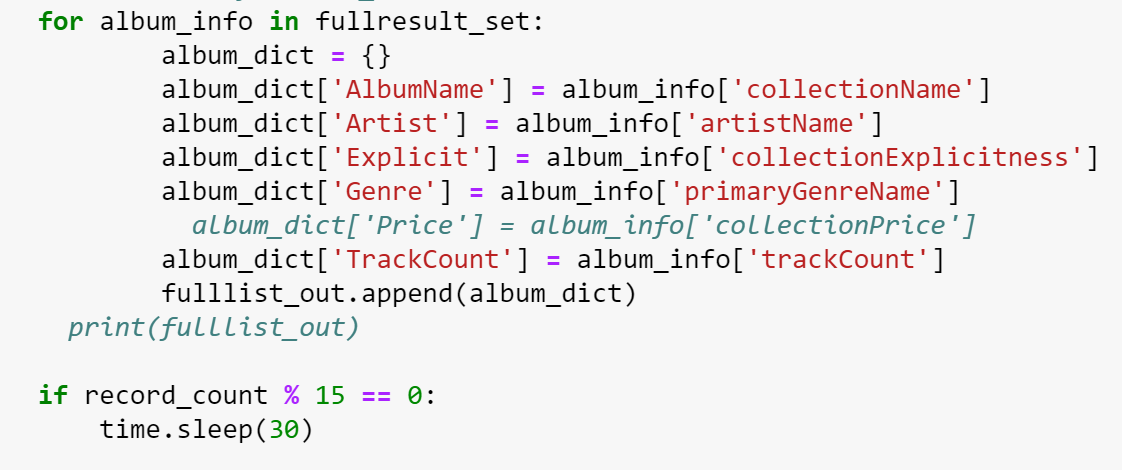
The iTunes Search API is a non-keyed database (woot!) which is not particularly user-friendly (not woot!): the API is built so that calls are made using a unique identifier assigned to the various searchable parameters (artist, album, etc.). For example, if we wanted to pull info on albums from Jack Johnson (no worries, none of us listens to that crap) we would use the URL: <https://itunes.apple.com/lookup?id=909253&entity=album> It isn’t possible to know the specific identifier for all artists and albums which spurred several minutes of panic and/or staring blankly at our screens; however, in viewing the different sample searches on the API site we stumbled across the URL <https://itunes.apple.com/search?term=jack+johnson&entity=album> We assumed we could simply switch out the existing ‘jack+johnson’ portion of the URL with our artists of interest and still be able to complete the search. Our assumption seemed to be correct based on a few test calls made using <https://itunes.apple.com/search?term=the+beatles&entity=album> and <https://itunes.apple.com/search?term=u2&entity=album>, which then led to us to pull in a list of artist names that we could pass through the iTunes Search API URL.

**Transform:**

Names pulled in from the Billboard list of Top 100 Artists needed to be transformed to fit into the iTunes URL format of artist\_firstname+artist\_lastname. Many artist names contain special characters that may or may not result in a 404 error if that special character wasn’t captured in the unknown-to-us URL (e.g. p!nk, for KING & COUNTRY, and dan + shay). Transformed names were defined by a variable that could then be passed into a for loop that would insert the formatted names into a stock URL to make artist-specific calls to the API.



The iTunes API pulled in JSON-formatted data which contained a significant amount of information we didn’t intend to use, such as album price, the URL for the album cover art, copyright information, and release date. To parse out only the information of interest, we used the following bit of code:



We used a 30 second sleep function with a record count of 15 because iTunes allows a maximum of 20 calls per minute, and Katy got tired of repeatedly getting 429 errors.

**Load:**

We combined the JSONified data in a pandas dataframe, then saved it to a csv file. From there we imported the csv into mongoDB after an incredible amount of special character ignore-code courtesy of John that in the end turned out to be some junk characters that weren’t where the error message said they would be or even visible to us. Thankfully we have hair dye, because stuff like this will seriously age you. We chose to use a non-relational database given how quickly music changes; we expect that Billboard will continue to publish lists of top artists, and rather than wiping the database clean with each new iteration we can simply add to the collection where at some point we will have very large volumes of data. Kind of like the Now That’s What I Call Music album series (currently on volume 103), but just without the bad cover art.

Our database captures the following types of data: Album name, Artist, Explicitness, the genre of music to which it belongs, and the number of tracks on that album.

**Conclusions:**

Our database can be used to allow Top 40 fans to quickly pull out compiled album info on a number of top-trending artists without having to use more traditional and time-consuming methods of searching the internet for individual artist discography.