A MUSICAL ANALYSIS

Preparing Billboard 100 and Youtube Statistics to Analyze Trends

Students: Yinka Adesanmi, Nitin Madaan, Yashwinie Shivanand

Submitted: Wednesday, December 18, 2019

Instructor: Ahmad El Nakib, Class 2

*Purpose: Showcasing trends in Youtube and Billboard 100 lists.*

*Approach: Tasks were divided among group members, who then compiled and compared data to form a cohesive deliverable.*

*The jupyter notebooks and raw files for these steps can be found in this github folder.*

#### Step 1 Extracting the data

**Data Source 1:**

**Trending YouTube Video Statistics (CSV)**

* We found a CSV dataset from Kaggle that provided “Trending YouTube Video Statistics” (link: <https://www.kaggle.com/datasnaek/youtube-new>). After downloading this csv file locally, we then proceeded to read the csv into our jupyter notebook using pandas (i.e. read\_csv syntax).
* The schema for this dataset can be found here:

{

video\_id: string,

trending\_date: date,

title: string,

channel\_title: string,

category\_id: string,

publish\_time: datetime,

tags: string

}

**Data Source 2:**

**YouTube Category IDs (JSON)**

* We downloaded a JSON dataset that provided information on YouTube category ids from the same kaggle website as above (link: <https://www.kaggle.com/datasnaek/youtube-new>). After downloading it locally, we imported the json file using the following syntax:

**with** open('US\_category\_id.json') **as** json\_file:

json\_data = json.load(json\_file)

* We did not need our own schema for this file because the json came with its own schema.
* After importing the JSON file, we needed to navigate into the json lists to track down category ids/titles. After finding where this information was located, we looped through the JSON to get a list of category id numbers and these corresponding category titles. This was done with the following syntax:

items = json\_data["items"]

ids = []

titles = []

**for** item **in** items:

ids.append(item["id"])

titles.append(item["snippet"]["title"])

titles

* After completing the loop, we created a dataframe using pandas to story the list of ids and category names.
* The schema for this dataset can be found here:

{

cat\_id: integer,

category: string

}

**Data Source 3:**

**Billboard’s The Hot 100 List (HTML)**

* We webscraped the Billboard Trending 100 list (link: <https://www.billboard.com/charts/hot-100>) to find the trending songs and corresponding artists. To do this, we used the Beautiful Soup syntax taught in class.
* The information was gathered using a “for loop,” and appending the found information to two empty lists (i.e. one list holds song information and another list holds artist information).
* When scraping through a loop, we saved the resulting artist and song information in a table with two corresponding columns.
* The schema for this dataset can be found here:

{

artist: string,

song\_title: date,

}

#### Step 2 Transforming the data

* After reading our csv and json files into our jupyter notebook with pandas to separate dataframes, we then changed the headings of relevant columns as needed and configured the data types (i.e. using .astype(str).astype(int)) so that we would be able to work with the information.
* We then merged both dataframes into one table for our use. We used a left join for this because we wanted to match information from the category id json file to the full csv of trending youtube statistics. The following syntax was used to complete this:

merged\_df = pd.merge(youtubecsv\_df, category\_list, how = "left", on="cat\_id")

* When this was done, we were then able to filter the dataframe to just get the columns that we want to upload into our production database.
  + The schema for this dataset can be found here:

{

video\_id: string,

cat\_id: integer,

views: integer,

likes: integer,

dislikes: integer,

comment\_count: integer,

category: string

}

* The Billboard information that we scraped using Beautiful Soup was also saved converted from a table to a dataframe.
  + The schema for this dataset can be found here:

{

artist: string,

song\_title: date,

}

#### Step 3 Loading the data

* After transforming our data on jupyter notebook using pandas, we then created a corresponding schema in postgreSQL using Pg Admin 4.
* Once this was done, we were able to use SQL Alchemy to load the data into Postgres. To do this, we needed to connect to the engine. Once connected, we checked our table was indeed present by using the following syntax:

engine.table\_names()

* After confirming we were connected to postgres, we were able to transforming our dataframes into databases using the to\_sql syntax.

#### Next steps

The information gathered here could be further analyzed to determine whether there is any relation between songs/artists trending on YouTube and songs/artists that Billboard identifies as in their “Top 100” categorization.