**ETL Project Spotify Write Up**

Members: Yasmine Aitouny, Chris Krokus, Ryan Thomas, and Ahasan Hossain

**Extraction:**

Using various rating charts from [Spotify Charts](https://spotifycharts.com/regional/), a table of the top songs was captured. The columns that make up the html for this chart include:

* Image (the album cover art)
* Rank/Position
* Trending Arrow (whether or not the Rank/Position moved from the previous day/week)
* Track (containing both the Song Title and the Artist)
* Streams

**Transformation:**

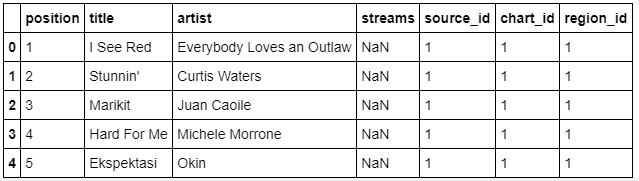
The first step after turning the HTML into a dataframe is to clean the data. We decided that the album art and the trending direction arrow were not necessary for our data and were subsequently dropped.. A string split on “”\_by\_” was applied to separate the Track and the Artist into separate columns. After renaming some columns we were left with Rank, Track, Artist, and Streams. Separate data frames were then created to represent regions (US, UK, and Global) for both Top 200, and Viral 50.

One thing to mention is that there is no stream count for the Viral 50, like there is for the Top 200. So we defaulted that column to NaN.

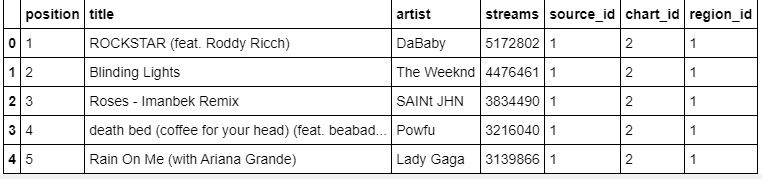
**Top 200**

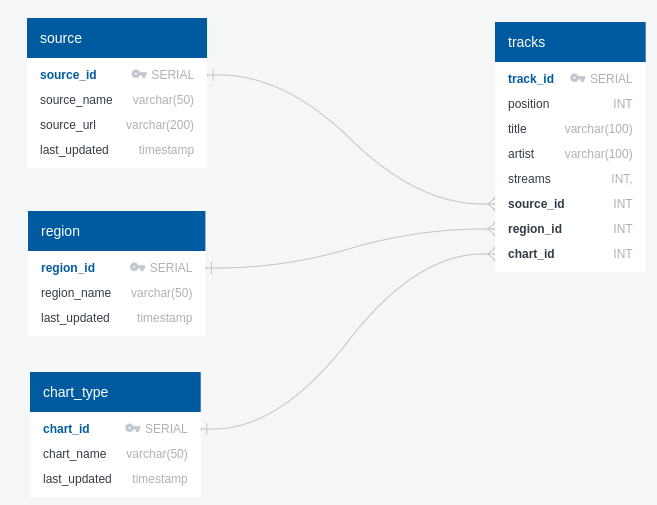


**Viral 50**



Columns for Chart type and Region were changed to match the ID fields from the ERD diagram.



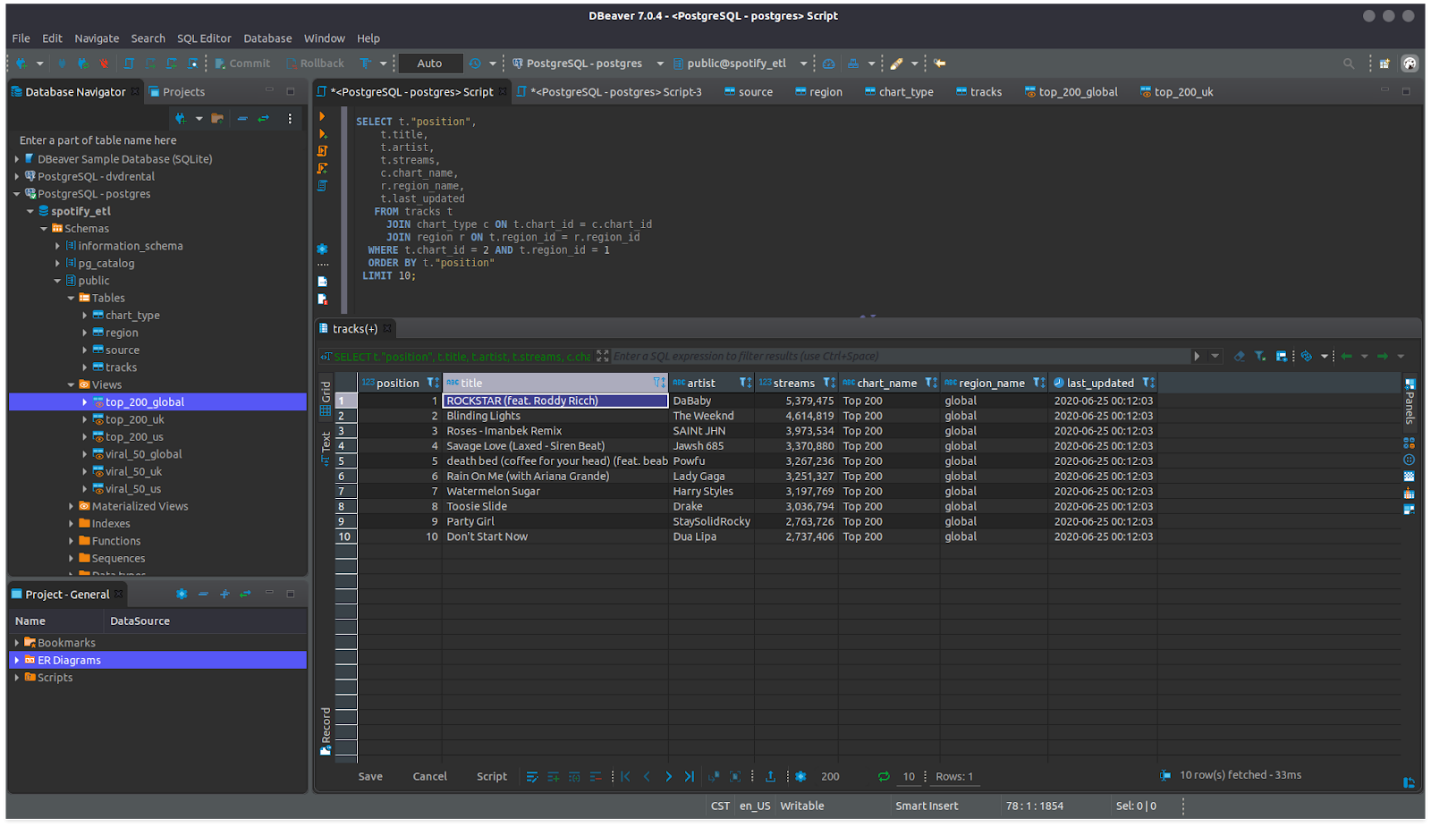


**Load:**

The final step in the process was to write the data into a database for storage. The steps for loading our DataFrames into the DataBase were to create a connection string (consisting of our login info, IP address, Port, and DataBase information), running that connection string through an engine, and connecting. Once connected, use df.to\_sql to fill the existing table. Once the date has been filled, close the connection and dispose of the engine. Repeat steps for all six DataFrames.

Our six DataFrames contained 750 rows of information, three lists of 200 and three lists of 50. In order to verify the entirety of the data had been transferred, you go into the DataBase and query = “““Select Count(\*) from tracks.”””

Below is a sample query from the top\_200\_global view that displays title, artist, streams, chart name, region, and a last updated column. This is joined from the tracks, chart\_type and region tables.



Scheduled Execution:

Team explored configuring the ETL process as a scheduled job. The web-scrape and loading functions were exported from the Jupyter Notebook into **spotify.py** and **web\_scape.py**. These files were uploaded to a linux server and a cron job configured to execute at regular intervals. Below is a outline of the steps:

* Create and activate a virtual environment
* Load dependencies
* Configure a cron job to execute the Python file. Below is a sample to run every 5 minutes:

\*/5 \* \* \* \* /home/ubuntu/envs/spotify/bin/python3 /home/ubuntu/Spotrack\_ETL/webscrape\_test.py

* Validate cron job ran at expected period using logs

$ sudo cat /var/log/cron.log | grep webscrape\_test.py

    Jun 24 **18:30**:01 k6-load-tester CRON[4684]: (ubuntu) CMD (/home/ubuntu/envs/spotify/bin/python3 /home/ubuntu/Spotrack\_ETL/webscrape\_test.py)

    Jun 24 **18:35**:01 k6-load-tester CRON[4712]: (ubuntu) CMD (/home/ubuntu/envs/spotify/bin/python3 /home/ubuntu/Spotrack\_ETL/webscrape\_test.py)

    Jun 24 **18:40**:01 k6-load-tester CRON[4724]: (ubuntu) CMD (/home/ubuntu/envs/spotify/bin/python3 /home/ubuntu/Spotrack\_ETL/webscrape\_test.py)

**Future Work:**

For future work, we would likely include s\_id 2 and s\_id 3 in the form of Billboard and Apple Music. Billboard has ‘Billboard 200’ and ‘Billboard Hot 100’ which would likely make up c\_id 3 and c\_id 4. Of note, both are tied to the United States music industry. On the other hand,  PopVortex.com has Apple Music Top 100 Songs data for 25 countries. Apple Music also provides a Genre for each song listed, which would be interesting to incorporate given more time to work with this expanded data set.