**ETL Project**

**Team Effort**

Team Members: **Hans Engelbrecht, Maclean Asante**

Presentation date**: 5/16/20**

**Project Proposal**

. Our ETL project proposal is YouTube viewer data downloaded from Kaggle. Our intention is to import the data into Pandas for transformation. When the transformation is complete, the data will be loaded in a MySQL database using PostgrSQL as the database front end.

**Finding Data (Data Sources)**

The source data from Kaggle can be found [here](https://www.kaggle.com/datasnaek/youtube-new). This data set is composed of 10 csv files and 10 json files. The csv files contain information about trending videos in a specific country between 2017-18. The country in question in contained in the name of the csv. The jsons are all the same copy of YouTube video categories that the videos contained in the csvs fall into. Ultimately, we used the csvs for the following countries.

* United States
* Canada
* Germany
* France
* Great Britain
* India
* Mexico

**Data Cleansing &Transformation & Load**

All cleaning and transforming of the data was done using Python in Jupyter notebook. Using the os library we loaded the csv file into our Jupyter notebook. Then we were able to loop through each row sorting all the individual columns into a series of lists. We were able to spin that list of lists into a Pandas dataframe that could be presented and exported to be loaded into SQL. Along the way the first transformation that we did was attempt to normalize the data by separating out the country of origin to create a lookup table for the countries. We then added a country ID to each video related csv to denote the country of origin. The next major transformation came about when we ran into repeated loading failures with Postgres. Initially when we attempted to load the data into SQL, the load would fail, citing that there was an unterminated cell in the ‘title’ column. Upon closer inspection we realized that many of the rows contained commas in the ‘title’ column which broke our load. To combat this we added code to replace any comma in that column with a space. From there loading our video csvs was successful. The next transformation that we did was to convert the publish date from a csv string to a datetime object so that we could do datetime based calculations in python if need be.

This example shows how we loaded the data and converted the publish dates.



This is where we created the dataframe with only the desired columns, removed the commas from the title column, and exported to csv to be loaded into Postgres.



Here is how we loaded and created the csv for the category json.

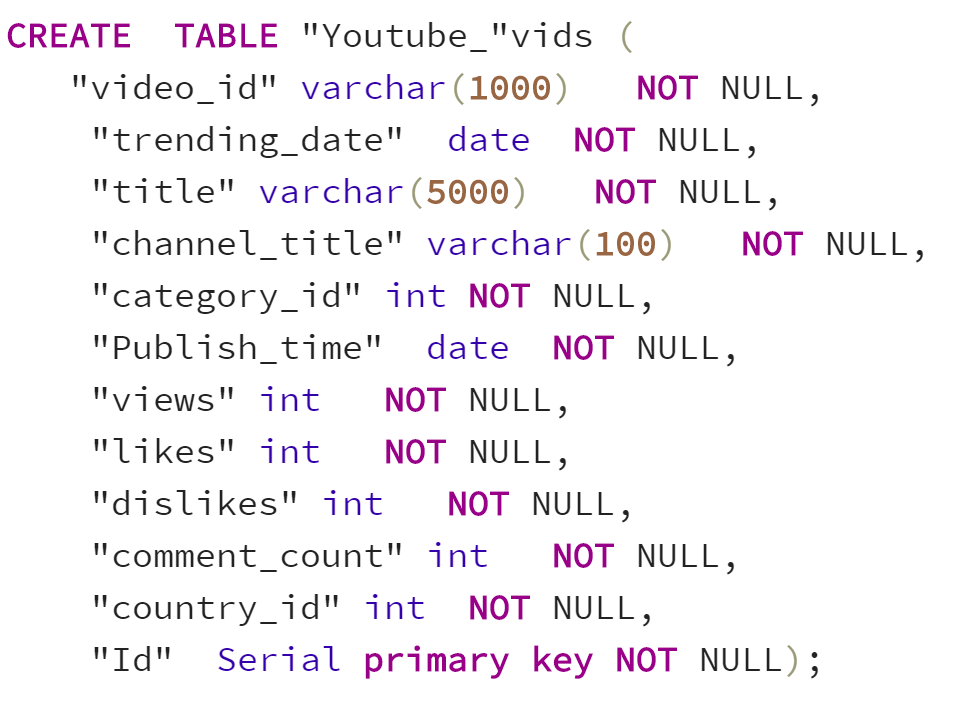


**DATA STORAGE**

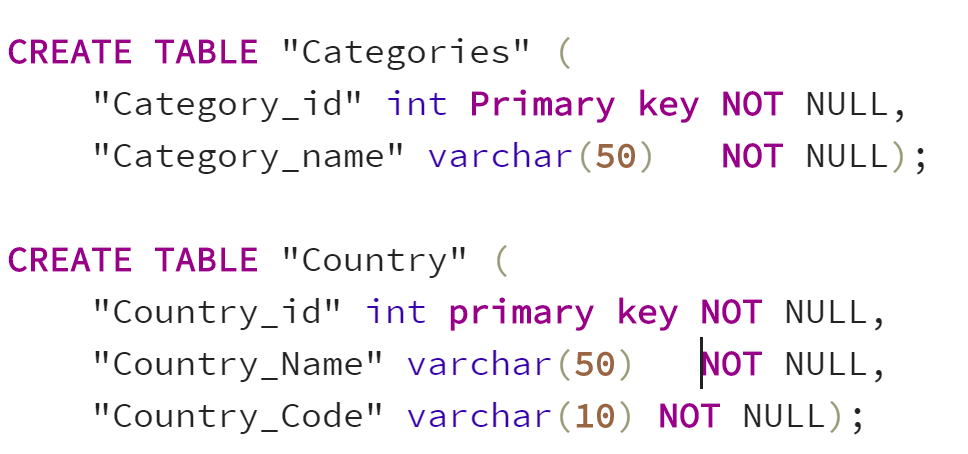
We chose to create a relational database to hold the data that we collected using Postgres. The reason why we chose this means of data storage is because the data was laid out in neat columns that were consistent across all the csvs that could be transformed and for the most part normalized eliminating the need for a non-relational database. Another, probably less relevant reason for our decision is that each one of these csv lists of videos has ~40,000 rows, so if this were to be a longer term study of Youtube engagement over several years and more countries, this database would have grown exponentially to potentially millions of rows. So to that end, a relational database seemed like it would be the more efficient strategy for storing data.

**Table Design**

This is the main table that contain the list of Youtube videos



Here are the categories and countries.



**\*\*Addendum\*\***

A database connection string that links our Jupyter notebook with Postgres was added to provide the ability to import each country’s trending videos csv to Postgres programmatically. The code creates a database engine, which is leveraged by the df.to\_sql command for the automatic import. The Juypter notebook opens one connection engine, on the first csv, then each subsequent csv transformation script contains a line of code to use the existing engine to send the data to the trending\_videos table in our YouTube\_db in Postgres

Here is the first csv transformation script that creates the engine.



Here is how each subsequent country specific csv is handled with respect to importing the containing data into the database.

