What's trending in North America?

ETL Project on YouTube Statistics

Team Members:

Carly Osborne | Kristen Mason | Rima Philip

Table of Contents

1 • INTRODUCTION

Background

Our extraction process started with reviewing available datasets from kaggle.com. Our goal was to find datasets with large volume of records with relevant data which can be used for analysis.

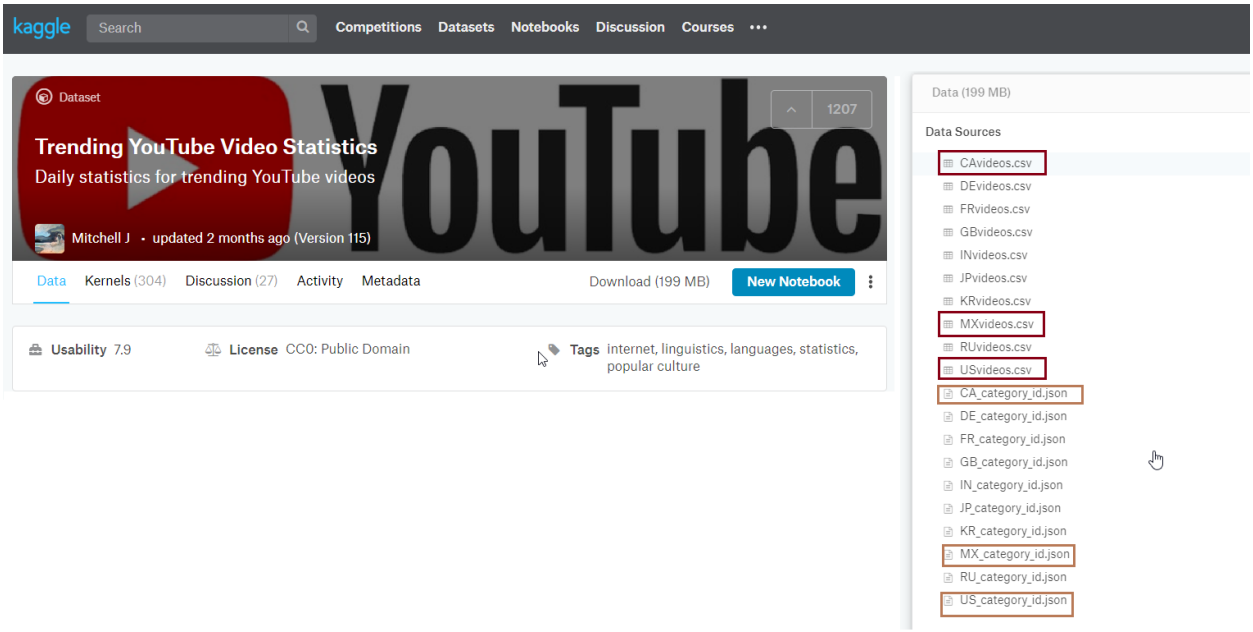
We came across a collection of datasets on trending YouTube video statistics which had datasets for different countries in both csv and json formats. Each dataset had over 41k records and had a lot of data(columns) with some good potential for cleanup as well.

All these aligned with our goal for this project and we decided to proceed with this collection and to limit our analysis within North America, we picked the datasets for US, Canada and Mexico.

We proposed compiling these data to better understand YouTube video popularity during an 8-month span (November 2017 - June 2018). Specifically, to determine what videos are most popular, how long it took to become trending, if certain categories tend to be more popular than others, and whether the observed trends are consistent across all of North America or if each country has a unique experience.

Data Source

https://www.kaggle.com/datasnaek/youtube-new/data



Data Sets


USvideos.csv


CAvideos.csv


MXvideos.csv


US_category_id.json

File Name	Type
US_videos	csv
CA_videos	csv
MX_videos	csv
US_category_id	json

Database Setup:

Next step was to visualize a database setup with the available data and we chose the opensource relational database management system 'Postgres' to setup our database since our data was already structured and we wanted to maintain that.

We decided to set up the youtube_db with the below tables.

Db Name	Table#	Table Name	Primary Key	Data Source
youtube_db	1	videos	video_id	csv
	2	country	country_id	csv
	3	popularity	id	csv
	4	category	category_id	json

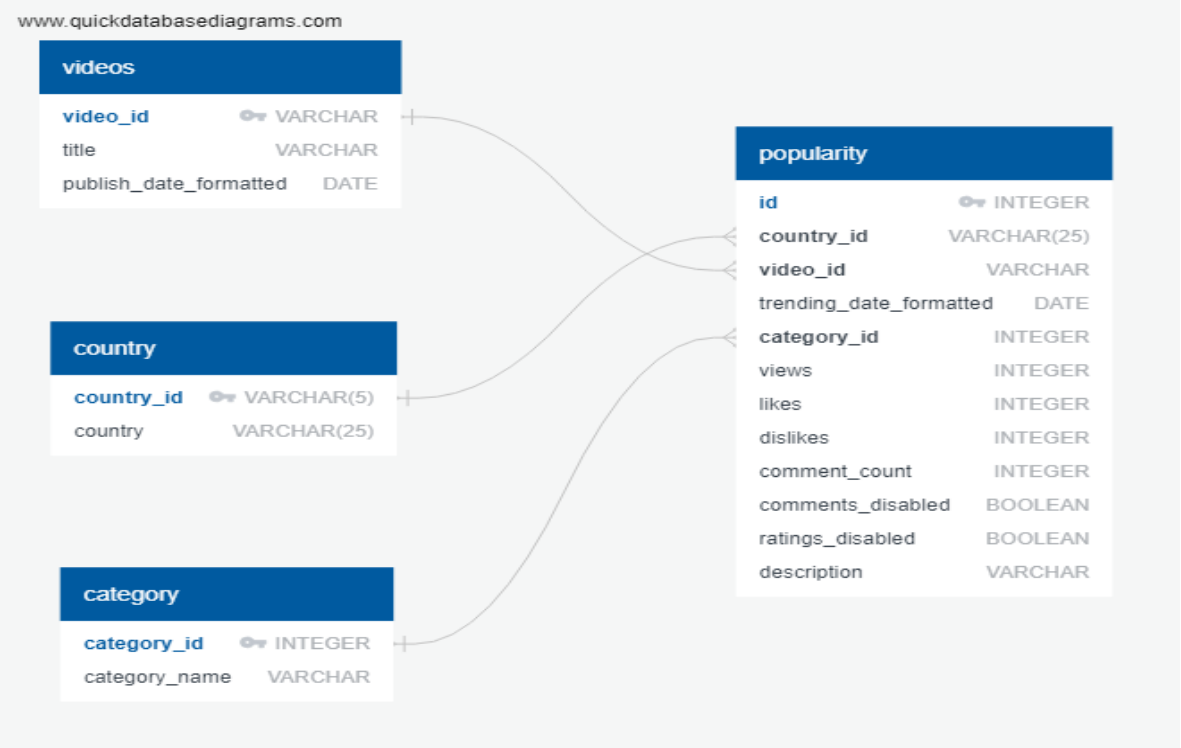
Table Schema | ERD

Using quickdatabasediagrams.com, we designed the below entity relationship diagram and derived the schema to setup the tables in the youtube_db.

Schema SQL:


QuickDBD-YoutubeVideos_NorthAmerica_S

Link to schema: https://app.quickdatabasediagrams.com/#/d/LHP3ZS



2 • EXTRACT

Extraction from CSV

After retrieving the csv datasets from Kaggle.com, reading the file using python and jupyter notebook was the next step.

We used the *read_csv()* function for this.

```
#Read Canada data from csv
csv_file1 = "../Resources/CAvideos.csv"
videos_ca = pd.read_csv(csv_file1)
```

```
#Read US data from CSV
csv_file3 = "../Resources/USvideos.csv"
videos_us = pd.read_csv(csv_file3)
```

```
#Read Mexico data from CSV
csv_file2 = "../Resources/MXvideos.csv"
videos_mx = pd.read_csv(csv_file2, encoding='ISO-8859-1')
```

Extraction from json

After retrieving the json datasets from Kaggle.com, next step was to read the json file using python and jupyter notebook. The json files for all the three countries were the same (YouTube categories are the same for all countries), so we used only the US_category_id json file to extract all the categories.

```
#dependencies
import json
import os
```

```
# Load JSON for US categories
filepath = os.path.join("../Resources", "US_category_id.json")
with open(filepath) as jsonfile:
    us_video_json = json.load(jsonfile)
```

3 • TRANSFORM

Transformation

The transformation aspect dealt with reviewing the dataframes created by reading the csv files and the json file. The steps we executed to transform out datasets into dataframes which were ready to load into the tables are below.

1. **Concatenation**

    1. Joined the three dataframes (US+CA+MX) into one (North America) and added a new column with country_id to distinguish between each country.
    2. Used the function *concat()* to join the three datasets into one

2. **Retaining required columns alone**

    1. Dropped all unnecessary columns and retained only the required ones.
    2. Retained columns:
    "country_id","country","video_id","category_id","trending_date","title","publish_time","likes","dislikes","views", "comment_count", "comments_disabled","ratings_disabled","description"

3. **Removal of blank records**

    1. There were many records without valid video_id which were dropped

4. **Formatting**

    1. Formatted date column data to standardize the date fields.
    2. Used to_datetime() function to format the dates.

5. Dataframes

Created 4 different dataframes to align with our 4 tables

1. videos dataframe
    i. Removed the duplicates from the master dataset and picked video_id, title and published_date_formatted fields to create the videos dataframe.

Table 1 | videos (source: csv)

```
1 videos.head()
```

	video_id	title	publish_date_formatted
0	2kyS6SvSYSE	WE WANT TO TALK ABOUT OUR MARRIAGE	2017-11-13
1	1ZAPwfrtAFY	The Trump Presidency: Last Week Tonight with J...	2017-11-13
2	5qpjK5DgCt4	Racist Superman | Rudy Mancuso, King Bach & Le...	2017-11-12
3	puqaWrEC7tY	Nickelback Lyrics: Real or Fake?	2017-11-13
4	d380meD0W0M	I Dare You: GOING BALDI?	2017-11-12

2. country dataframe
    i. Created country dataframe with country_id and a new column country_name

Table 2 | country (source: csv)

```
1 country = videos_northamerica.drop_duplicates('country_id',keep='first')
2 country = country[["country_id","country"]]
3 country.head()
```

	country_id	country
0	US	United States
0	CA	Canada
0	MX	Mexico

3. popularity dataframe
    i. The master dataframe with all North America data was retained as the popularity dataframe.
    ii. This dataframe had the details like, trending_date, views, likes, dislikes, comment_count etc. apart from video_id, country_id and category_id which were retained to build foreign key dependencies in the database between the tables.

Table 3 | popularity (source: csv)

```
popularity = videos_northamerica[["country_id","video_id","category_id","trending_date_formatted","likes","dislikes","views"
                                  "comment_count","comments_disabled","ratings_disabled","description"]]
#popularity.loc[:,"id"] = np.arange(len(popularity))


popularity.insert(0, 'id', range(1, 1 + len(popularity)))
popularity.head(5)
```

	id	country_id	video_id	category_id	trending_date_formatted	likes	dislikes	views	comment_count	comments_disabled	ratings_disabled	
0	1	US	2kyS6SvSYSE	22	2017-11-14	57527	2966	748374	15954	False	False	Sh https
1	2	US	1ZAPwfrtAFY	24	2017-11-14	97185	6146	2418783	12703	False	False	pres
2	3	US	5qpjK5DgCt4	23	2017-11-14	146033	5339	3191434	8181	False	False	VID
3	4	US	puqaWrEC7tY	24	2017-11-14	10172	666	343168	2146	False	False	Toda
4	5	US	d380meD0W0M	24	2017-11-14	132235	1989	2095731	17518	False	False	

4. <u>category dataframe</u>
    i. This dataframe was created from the json file US_category_id and had all the YouTube video category information.
    ii. category_id and category_name were the only fields extracted to create the category dataframe.

Table 4 | category (source: json)

```
#dataframe with video categories for category table
category = us_video_json_normalized[["category_id","category_name"]]
category.head()
```

	category_id	category_name
0	1	Film & Animation
1	2	Autos & Vehicles
2	10	Music
3	15	Pets & Animals
4	17	Sports

Note:

The process of extraction and transformation was slightly different for the dataframes created by reading the csv files and the json file.

- **CSV** - Transforming the dataframes created by reading the csv file after cleanup was direct. It involved selection of the required columns for each table.

- **Json** - Transforming dataframes extracted from the json file involved flattening the nested json into the dataframe.
- *json_normalize ()* function was used to retrieve data from the nested json file.

DEPENDENCIES & SETUP

```
# Dependencies and Setup
import pandas as pd
from sqlalchemy import create_engine
import datetime
import numpy as np
import psycopg2
from pandas.io.json import json_normalize
```

Table 4 | category (source: json)

```
#flattening the nested json to dataframe
us_video_json_normalized = json_normalize(us_video_json['items'])

#rename column names (id to category_id) and (snippet.title to category name)
us_video_json_normalized.rename(columns={'id':'category_id',
                                         'snippet.title':'category_name'},
                                inplace=True)
us_video_json_normalized.head()
```

4 • LOAD

Preconditions:

- The youtube_db database was setup in Postgres relational database management system.
- The table schema was derived using the Entity Relationship diagram created using quickdatabasediagram.com website.
- set up the dependencies and relationships between the tables.
- Executed the schema sql and created the 4 tables in youtube_db with the respective columns matching the corresponding dataframes.
- More details are given on Page 2

Loading Process

Loading the tables involved two steps:

1. **Connecting to the local database (postgres)**
    - Installed Psycopg2 which is a DB API 2.0 compliant PostgreSQL driver. It is one of the most popular PostgreSQL database adapters for the Python programming language.
    - Created a connection string to pass the username and password for the postgres database.

Load the dataframes as tables into relational postgres database

```
#dependencies
!pip install psycopg2
```

```
Requirement already satisfied: psycopg2 in c:\users\rima\anaconda3\lib\site-packages (2.8.3)
```

```
#connect to local database
rds_connection_string = "<insert user name>:<insert password>@localhost:5432/youtube_db"

engine = create_engine(f'postgresql://{rds_connection_string}')
```

```
#check tables already created using the table schema sql generated from Quick database diagram
engine.table_names()
```

```
['country', 'popularity', 'videos', 'category']
```

2. **Loading the dataframe into corresponding tables.**
    - Used the to_sql() function to load the records from the dataframes into the tables.

Load Table 1 | video

```
#Use pandas to load csv converted DataFrame into database
videos.to_sql(name='videos', con=engine, if_exists='append', index=False)
```

Load Table 2 | country

```
#Use pandas to load csv converted DataFrame into database
country.to_sql(name='country', con=engine, if_exists='append', index=False)
```

Load Table 4 | category

Note: load Table 4 before Table 3 due to dependency (foreign key category_id)

```
category.to_sql(name='category', con=engine, if_exists='append', index=False)
```

Load Table 3 | popularity ¶

```
#Use pandas to load csv converted DataFrame into database
popularity.to_sql(name='popularity', con=engine, if_exists='append', index=False)
```

5 • CONCLUSION

We verified that the data is loaded correctly by querying the databases both in Postgres and using python. Our analysis led to the below interesting findings.

1. Top trending videos in North America
    - Childish Gambino - This Is America (Official Video)
        - Country: US
        - Published Date: 2018-05-06
        - Trending Date: 2018-05-19
        - Time taken to trend: 13 days.
        - Views: 225211923
    - YouTube Rewind: The Shape of 2017 | #YouTubeRewind
        - Country: US
        - Published Date: 2017-12-06
        - Trending Date: 2017-12-14
        - Time taken to trend: 8 days.
        - Views: 113876217
2. Top Categories of trending videos
    - Music
    - Entertainment

3. The popularity across the countries varied overtime
    - The video 'YouTube Rewind: The Shape of 2017 | #YouTubeRewind' was trending in all the countries in Dec 2017.

Top Trending video in North America

o Childish Gambino - This Is America (Official Video) was trending in 13 days in US with views over 2MII

```
]: 1  pd.read_sql_query("""SELECT ca.category_id, ca.category_name, v.video_id, v.title, p.trending_date_formatted as "Trending da
   2  v.publish_date_formatted as "Published date", p.trending_date_formatted-v.publish_date_formatted as "days to trend", p.views
   3  c.country
   4  FROM videos v
   5  JOIN popularity p
   6  ON v.video_id = p.video_id
   7  JOIN country c
   8  ON p.country_id = c.country_id
   9  JOIN category ca
  10  ON p.category_id = ca.category_id
  11  WHERE p.views>100000000
  12  ORDER BY views DESC;""", con=engine).head()
```

]:

	category_id	category_name	video_id	title	Trending date	Published date	days to trend	views	likes	country
0	10	Music	VYOjWnS4cMY	Childish Gambino - This Is America (Official V...	2018-06-02	2018-05-06	27	225211923	5023450	United States
1	10	Music	VYOjWnS4cMY	Childish Gambino - This Is America (Official V...	2018-06-01	2018-05-06	26	220490543	4962403	United States
2	10	Music	VYOjWnS4cMY	Childish Gambino - This Is America (Official V...	2018-05-31	2018-05-06	25	217750076	4934188	United States
3	10	Music	VYOjWnS4cMY	Childish Gambino - This Is America (Official V...	2018-05-30	2018-05-06	24	210338856	4836448	United States
4	10	Music	VYOjWnS4cMY	Childish Gambino - This Is America (Official V...	2018-05-29	2018-05-06	23	205643016	4776680	United States

Querying Table 1 | videos

```
1  pd.read_sql_query('select * from videos', con=engine).head()
```

	video_id	title	publish_date_formatted
0	2kyS6SvSYSE	WE WANT TO TALK ABOUT OUR MARRIAGE	2017-11-13
1	1ZAPwfrtAFY	The Trump Presidency: Last Week Tonight with J...	2017-11-13
2	5qpjK5DgCt4	Racist Superman | Rudy Mancuso, King Bach & Le...	2017-11-12
3	puqaWrEC7tY	Nickelback Lyrics: Real or Fake?	2017-11-13
4	d380meD0W0M	I Dare You: GOING BALD!?	2017-11-12

Querying Table 2 | country

```
1 pd.read_sql_query('select * from country', con=engine).head()
```

	country_id	country
0	US	United States
1	CA	Canada
2	MX	Mexico

Querying Table 3 | popularity

```
1 pd.read_sql_query('select * from popularity', con=engine).head()
```

	id	country_id	video_id	trending_date_formatted	category_id	views	likes	dislikes	comment_count	c
0	1	US	2kyS6SvSYSE	2017-11-14	22	748374	57527	2966	15954	
1	2	US	1ZAPwfrtAFY	2017-11-14	24	2418783	97185	6146	12703	
2	3	US	5qpjK5DgCt4	2017-11-14	23	3191434	146033	5339	8181	
3	4	US	puqaWrEC7tY	2017-11-14	24	343168	10172	666	2146	
4	5	US	d380meD0W0M	2017-11-14	24	2095731	132235	1989	17518	

Querying Table 4 | category

```
1 pd.read_sql_query('select * from category', con=engine).head()
```

	category_id	category_name
0	1	Film & Animation
1	2	Autos & Vehicles
2	10	Music
3	15	Pets & Animals
4	17	Sports

Querying all the tables using joins

In [6]:

```
pd.read_sql_query("""SELECT v.video_id, v.title, v.publish_date_formatted as "published date", p.trending_date_formatted as
            FROM videos v
            JOIN popularity p
            ON v.video_id = p.video_id
            JOIN country c
            ON p.country_id = c.country_id
            JOIN category ca
            ON p.category_id = ca.category_id
            WHERE c.country= 'Canada'
            AND ca.category_id=22
            AND publish_date_formatted>'2017-11-13';""", con=engine).head()
```

Out[6]:

	video_id	title	published date	trending date	views	likes	country	category_id	category_name
0	1Zp_x9BSVVA	IE 21 JANVIER 2018 EYINDI NA KINSHASA	2018-01-21	2018-01-22	77810	311	Canada	22	People & Blogs
1	u2Ba65YELoo	The Reaction of The Streets (I Wait-Day6 Edition)	2017-12-05	2017-12-06	88889	25599	Canada	22	People & Blogs
2	lddDvegXlRY	Wonder - Julian Gets In Trouble (HD)	2018-02-15	2018-02-18	1268346	10565	Canada	22	People & Blogs
3	5czmY2tK07Y	JRE MMA Show #5 with Stipe Miocic	2017-12-13	2017-12-14	342738	9735	Canada	22	People & Blogs
4	fbOFogaFF84	The View March 14, 2018 ; Lena Waithe	2018-03-14	2018-03-15	58534	368	Canada	22	People & Blogs